A Basis for Source Coding

W. Bastiaan Kleijn
KTH (Royal Institute of Technology)
100 44 Stockholm, Sweden

March 12, 2011

A Basis for Source Coding

Contents

1

Overview

1.1 Introduction

The emergence of new technologies is associated with an ever-increasing need for the transmission and storage of signals. Most of these signals are discrete-time (sampled) signals[1]. The abundance of signals requiring transmission makes it natural to attempt to reduce the bit rate required to represent a signal. **Source coding** refers to the encoding of a source signal with a reduced bit-rate representation, the **code**, and the decoding of this reduced bit-rate representation into an approximate or exact copy of the original signal.

Particularly since about 1980, source coding has become an integral part of every day life. Mobile phones and audio-playback devices use speech and audio-coding techniques. Audio, video, and image compression are used for the Internet, digital television, and video-playback devices. Without source coding, these technologies would be significantly more expensive. The cost savings result from a reduction in bit rate that is typically an order of magnitude for speech and audio coding and up to three orders of magnitude for video and image coding when compared to the "raw" signal.

It is clear that there must be a lower bound on the bit rate that is required to characterize a certain signal at a certain fidelity. Past work in source coding has brought us significantly closer to such bounds. Unfortunately, it is generally difficult to provide good estimates of the lower rate bounds since both the character of the signal (e.g., the human voice) and the character of the receiver (e.g., the human auditory system and the successive processing by the brain) are often not well understood. However, judging by the speed at which bit rates are being reduced, the lower bound required to represent many source signals is not yet close. It seems safe to state that much progress remains to be made in the future, and that a significant fraction of this progress will be linked to a better understanding of the generation of the signal and of the perception of the signal by the receiver.

In this introductory chapter, we provide a brief outline of the fundamental principles

[1] In the following, the term "signal" refers to a discrete-time random process except where stated otherwise.

that underlie the field of source coding. We end the chapter with an overview of the remainder of the book.

1.2 Why Source Coding Works

Many signals can be represented at a lower bit rate without destroying their essential character. Two fundamentally different reasons facilitate bit rate reduction:

1. The receiver tolerates **distortion** in the signal.
2. The original signal description contains **redundancy**.

The tolerance for distortion can be exploited by reducing the precision of the representation of a signal, and, thus, the number of bits required to describe it. Such coding is called **lossy** coding. To minimize the impact of lossy coding, it is important to know how distortion is interpreted by the final receiver of the signal. For many signals to which source coding is applied, the final receiver is either the human visual or auditory system. These sensory systems have varying sensitivity for different features of the signals. By decomposing the signal into a set of new signals describing different features of the original signals, and by distributing the distortion selectively over these new signals, it is often possible to have significant distortion (in a squared-error sense) without it being perceived by the receiver. In many cases, certain features of the original signal do not require transmission at all.

> **Example 1.1: Auditory masking**
> Audio signals that are audible if no other audio signals are present can become inaudible in the presence of a louder signal. For example, when the sound of a car engine is loud, a radio signal may be inaudible. The information transmitted to the car radio is then irrelevant. Similarly, a low-amplitude tone nearby in frequency to a tone of high amplitude is often not audible and the description of the low-amplitude tone is not required for perceptually accurate reconstruction.

Next, we discuss redundancy. Let us consider the encoding of the realization of a stationary sequence of random variables (a process) into a sequence of bits. Redundancy rate[2] then refers to the average excess of bits per variable over that needed with the most efficient code theoretically possible that allows perfect reconstruction. The redundancy rate is greater than zero (it cannot be negative) if the statistics of a sequence of random variables are not fully exploited by the code. For example, it can mean that the dependencies between variables (samples) are not properly taken advantage of. Information may be shared between variables that are encoded independently. The shared information is sent multiple times, which is redundant. From another vantage point, redundancy relating to dependencies originates from not accounting for the non-uniformity of the multi-variational probability distribution of the data. Not accounting for non-uniformity of the distribution is also the cause of redundancy in the case of scalar variables. Consider, for example, a discrete random variable that has a non-uniform probability distribution and all values are encoded with **codewords** (a codeword is the

[2]The term "redundancy" will be used to refer loosely to the excess bits.

sequence of bits used to represent a particular value from the set of values that the random variable can take) of equal length. The average bit rate can then usually be reduced by assigning shorter codewords to high probability values and longer codewords to low probability values. The operation of removing redundancy from a sequence of discrete symbols (such as a sequence of text symbols) is called **lossless** coding.

Example 1.2: Redundancy in a discrete-time speech signal
Redundancy occurs in a sampled (discrete-time) speech signal. A one-sample-at-a-time description of this signal ignores that the samples of the speech signal are statistically interdependent. These interdependencies imply that if we know something about one sample, then we know something about surrounding samples as well. This information is repeated if we encode all samples separately and this repetition corresponds to redundancy. In addition, the amplitude distribution of speech is not uniform and as the samples are commonly encoded using a fixed codeword length, which is a second cause of redundancy.

Example 1.3: Redundancy related to marginal distributions
Let us consider the transmission of written text, one letter at a time, by means of a sequence of binary codewords. We assume that there are 128 distinct text symbols. It seems then natural to use 7 bits to describe each of the text symbols. However, certain text symbols, such as the letters "a" and "e", are more likely than text symbols such as "x" and ";". It can be shown, with the methods discussed in chapter 2, that we can save bits by assigning shorter codewords to the more commonly occurring text symbols, and longer codewords to the less common text symbols.

1.3 Source Coding Strategies

In this section, we provide a brief overview of source coding strategies that are used for real-world signals such as video, images, speech, and audio. We assume the input signal is a sampled signal with analog (infinite precision) amplitudes. In the lossy-coding subsection, we discuss methods that exploit the tolerance of the receiver to distortion. In the redundancy-removal subsection, we discuss methods that reduce the redundancy rate.

1.3.1 Lossy Coding

Lossy coding methods decrease the coding rate at the expense of increased distortion. The art of lossy coding is to minimize the distortion for a given rate or vice versa. Theoretical results on the lower bound on the distortion for a given rate are discussed in chapter 6. Here, we focus on the practical aspects of lossy coding. We briefly describe scalar quantization, vector quantization, and the distortion criterion.

Scalar Quantization

The simplest form of lossy coding is **scalar quantization**. It is used to encode scalar, continuous random variables. Scalar quantization divides the real line, $\mathbb{R}$, into distinct regions called cells. Each cell has a reconstruction value, which normally lies within the cell. Scalar quantization is the non-invertible mapping that maps, for all cells, all points contained in the cell to the corresponding reconstruction value (the boundary points can map into any bordering cell). The mapping is illustrated in figure 1.1.

Scalar quantization is useful since we can assign an **index** to each cell. We can transmit or store this index and then look up the corresponding reconstruction value for the cell. Scalar quantization is discussed in detail in chapters 7 and 8.

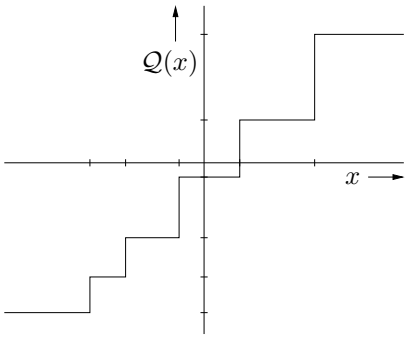

Figure 1.1: The mapping performed by a scalar quantizer from a value x to a (quantized) value $\mathcal{Q}(x)$. The cell boundaries are indicated on the horizontal axis and the reconstruction values on the vertical axis.

Vector Quantization

A generalization of scalar quantization is **vector quantization**. As the name suggests, a vector quantizer has a vector as input and the quantizer cells are multi-dimensional. An example of a two-dimensional vector quantizer is shown in figure 1.2. For a k-dimensional vector quantizer, the cells partition $\mathbb{R}^k$.

Assuming the joint statistics of the vector components are known, vector quantization is asymptotically optimal. That is, for a given mean distortion per component, the bit allocation per component (the rate) of an optimal vector quantizer converges asymptotically with increasing vector dimensionality k to the **rate-distortion bound**, which is the lower bound on the rate possible for the given distortion. This implies that vector quantizers can remove both irrelevancy and redundancy. As a result, vector quantizers have become an important tool in source coding.

To apply vector quantization to a discrete-time (sampled) signal, we can divide the signal into subsequent blocks (vectors), and quantize each block separately. If the signal is a stationary process, this application of vector quantization is asymptotically optimal.

The asymptotic optimality of vector quantization may lead one to believe that with vector quantization one should be able to solve most practical source-coding problems.

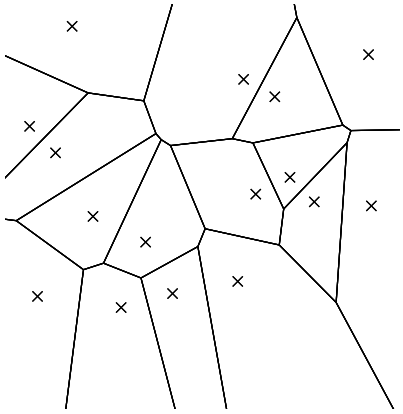

Figure 1.2: The mapping performed by a two-dimensional vector quantizer designed for two-dimensional vectors of uncorrelated normal distributed elements. The cells and the corresponding reconstruction points are shown.

Unfortunately, this is not the case. While vector quantizers are asymptotically optimal, their computational complexity often increases exponentially with their dimensionality (assuming a constant bit allocation per dimension). To prevent this problem, one can add structure (e.g., a lattice structure of the cells) to the vector quantizer, but then the asymptotic optimality may no longer be guaranteed. Structured and unstructured vector quantization are discussed in chapters 7 and 8.

Example 1.4: Codebook size for direct vector quantization of speech
According to some estimates, based on certain assumptions, perceptually transparent encoding of a continuous-time speech signal band-limited to about 3500 Hz (i.e., telephone bandwidth) requires a bit rate of about 8000 bits per second. For an 8000 Hz discrete-time speech signal, this corresponds to 1 bit per sample. We furthermore know from the nearly periodic nature of speech, that dependencies easily stretch over 10 ms, or 80 samples. To capture at least some of these dependencies, the vector dimension would have to be at least 80 samples, and the resulting codebook size would be $2^{80} \approx 10^{24}$. Such large codebooks can neither be trained nor searched without including structure.

The Distortion Criterion

Lossy coding leads to distortion and, as mentioned before, the objective is to minimize distortion given the rate (or vice versa). To make this optimization meaningful, the distortion criterion must be relevant to the receiver. In practical source coding applications such as image and audio coding, the distortion criteria are designed to form an approximation to human perception. Unfortunately, perceptually accurate distortion criteria are often computationally complex.

In theorical work, as well as in many applications, so-called **single-letter** distortion criteria are commonly used. A single-letter distortion criterion for a vector consists of a sum over the distortions for the vector components. As a result, single-letter distor-

tion criteria facilitate analytic manipulation and generally lead to a low computational complexity for quantization. Indeed, the usage of single-letter distortion criteria is often motivated by computational cost and not by relevance. Straightforward quantization based on perceptually accurate distortion criteria that depend simultaneously on multiple samples generally leads to computational problems.

It is sometimes possible to transform the signal from a domain where perceptual accuracy leads to computationally complex distortion criteria to a domain where a single-letter criterion is a good approximation. That is, the co-dependency of the distortion on signal samples is removed or reduced. This effect should not be confused with statistical independence of the vector components. For both visual and auditory perception, cosine and Fourier transforms are used for this purpose (in addition to the purpose of redundancy removal, which is discussed in section 1.3.2).

Example 1.5: Distortion criteria in audio coding
An example of a transformation simplifying the distortion criterion is the discrete cosine transform used in many audio coders. Usage of a single-letter weighted mean-squared error distortion criterion on the cosine transform coefficients performs significantly better than a similar criterion used directly on the audio-signal samples. In other words, for a given performance of the distortion criterion, the cosine transform coefficients allow a computationally simpler form of the distortion criterion.

1.3.2 Redundancy Removal

Redundancy can be removed in a number of ways. We have already mentioned that vector quantization removes redundancy. However, in vector quantizers, we also remove irrelevancy. Here, we will discuss methods that remove redundancy only: lossless coding and signal-processing procedures.

We start with methods that are commonly referred to as **lossless coding**. These methods generally operate on discrete random variables (variables that can take a countable set of values). We consider the case where each variable is encoded separately with a binary codeword that specifies the value. The entire set of codewords is called a code. The mean length of the codeword in bits determines the bit rate required to transmit a sequence of the variables. It is often possible to find a code with a shorter mean codeword length under the constraint that exact reconstruction is still possible, thus reducing the bit rate. The task of finding a shorter code is performed by a lossless coder. This type of lossless coding is commonly performed on quantizer indices in the context of lossy coding. An example of this application is the MPEG-2 AAC audio coder [1]. A more general type of lossless coding also considers dependencies between samples of a sequence. A simple manner to accomplish this is to group samples and code them together with a single codeword. Procedures that perform lossless coding are described in chapter 5.

Example 1.6: Lossless coding of a random variable
Consider a random variable X (a source) that maps into the values (has as alphabet) $\{a, b, c, d\}$. The probability distribution is $P(X = a) \equiv p_X(a) = 0.99$, $p_X(b) = p_X(c) = p_X(d) = 0.0033$. The simple way of encoding X would be with

the codewords $c(a) = 00$, $c(b) = 01$, $c(c) = 10$, and $c(c) = 11$. Clearly, encoding a sequence of source symbols requires an average rate of 2 bits per source symbol (the source sequence *aaacaa* is encoded by 000000100000). To make the encoding more efficient, we would like to encode the symbol a with just one bit, e.g., $c(a) = 0$, allowing us to approach a rate of 1 bit per symbol. To allow us to reconstruct the original source sequence, this means that the other codewords cannot start with a 0. We can use, for example $c(b) = 10$, but now we cannot use a 0 as second bit for c and d, so we use $c(c) = 110$ and $c(d) = 111$. It is easily verified that *aaacaa* is now encoded by 00011000 and that the source symbols can be reconstructed from this short bit sequence.

A major purpose of **signal processing** in coding is the removal of redundancy. Signal processing is commonly used in combination with lossy coding procedures, but it can also be used as part of a lossless coding procedure. In section 1.2, we noted that the samples of signals are interdependent and this implies that coding each sample independently from the others is inefficient. Doing so would mean that we transmit the same information multiple times. This argument can be turned around: coding samples or variables that are statistically independent with scalar quantizers (or vector quantizers of low dimensionality) is relatively efficient. Although it can be shown that scalar quantizers and vector quantizers of low dimensionality cannot quite reach the rate-distortion bound, they can come quite close if the variables to be quantized are statistically independent.

The above discussion suggests a commonly used strategy for source coding: use a signal-processing algorithm to transform the signal samples into a set of statistically independent variables (reducing redundancy). Assuming a simple distortion criterion that is not affected by the transform, the independent variables can then be coded efficiently with relatively straightforward, low-complexity quantizers, and upon decoding the inverse transformation is applied. This basic principle provides the motivation for employing transforms (including filter banks) and signal modeling (e.g., autoregressive modeling).

Using block transforms is one approach towards making the signal samples independent. In this method, the signal is divided into blocks, and each block is then subjected to an invertible transform aimed at reducing or removing the statistical dependencies. In practice, it is difficult to remove all dependencies from the signal, but if the statistics are known, it is relatively straightforward to remove correlations (decorrelate the signal samples). In the case of normal (Gaussian) distributed samples, this is sufficient for obtaining independent variables, and otherwise it can be seen as an approximation to this situation. Often, the decorrelating transform (which is called the Karhunen-Loève transform) is well approximated by the discrete cosine transform (DCT). An example of a coding system that uses the DCT at least in part for decorrelation is the JPEG image coder. More details about the procedure will be provided in chapter 9.

Another method to reduce dependencies between samples of a signal to be coded is linear prediction, where each sample is predicted as a linear combination of previous samples (the memory). The simplest example of a linear predictor is the subtraction of a previous sample from a present sample in a differential quantizer[3]. It can be shown that, for a stationary signal, the prediction error samples of a linear predictor converge to

[3]In a differential quantizer, the previous *quantized* sample value is subtracted from the current sample prior to encoding.

being uncorrelated with increasing memory length. As a result, encoding the prediction residual[4] rather than the original signal often results in a significant increase in coding efficiency. Linear prediction is described in chapter 10.

Example 1.7: Linear prediction in speech coding
Linear prediction based source coders are particularly common in speech coding, where it has two major advantages. First, a significant amount of the redundancy of the speech signal can be removed with predictors of low order. This suggest that an autoregressive model of the signal is relatively accurate. Second, linear prediction performs (partial) decorrelation without requiring delay. This is a major advantage of linear prediction over block-based transform methods. Having a low delay in a transmission system means that echos integrate perceptually with the original signal, and that, therefore, an echo canceler is not required. Low delays are also desirable in situations where a receiver may hear the signal both as transmitted through a network and through an acoustic path (e.g., flight-control rooms and hearing aids). Examples of standardized low-delay coders (cf. [2]) based on linear prediction are the adaptive differential pulse code modulation (ADPCM) algorithm (the ITU G.726 standard) and the low-delay CELP coder (ITU G.728). The linear-prediction based coders used in mobile telephone networks, generally operate on large (10-25 ms) signal blocks, and do not exploit the low delay advantage of linear prediction.

It is interesting to note that methods to remove redundancy can, in general, also be interpreted as being part of a structured vector-quantization procedure. Let us consider blockwise coding of a signal based on adaptive linear prediction, where the predictor is transmitted as side information. The information contained in a signal block is decomposed into the specification of an adaptive linear prediction-error filter, usually described by an index into a codebook of prediction coefficient vectors, and a specification of the excitation signal, usually described by an index into an excitation vector codebook. If we invert the prediction-error filter we obtain the the autoregressive filter structure. Given the past reconstructed signal, the combination of a particular index for the excitation-vector codebook, and a particular index for the codebook of linear prediction coefficient vectors, we obtain a particular reconstruction of the signal block. Thus, each combination of an excitation specification and an autoregressive model specification describes an entry in a vector codebook of signal segments. Because of the recursive structure of the autoregressive model, this vector codebook changes each coding block.

Signal processing is commonly used in source coding. Indeed, it is striking that much of the literature on practical source coding methods appears in journals on signal processing, and not in journals dedicated to information theory. This is a clear indication that signal processing plays an important role in practical source coding algorithms.

1.4 Outline of the Remainder of Book

The goal of this book is to provide an understanding of the main principles on which modern source coders are based. The contents of this book are applicable to any type

[4]It will be shown in chapter 10 that the standard mean squared-error distortion criterion is invariant with the prediction operation only if we use so-called *closed-loop* prediction.

of signal, with many of the examples drawn from audio coding.

We start with two chapters on information theory, which have as main purpose to define what information is. (Appendix A provides background information on probability theory.) For variables that can take a countable number of values, the definition implies a lower bound on the bit rate that can be obtained when using lossless coding (coding that allows perfect reconstruction).

The information-theory chapters are followed by a chapter on estimating probability distributions. Probability distributions are used extensively in information theory and for practical applications we have to know how to estimate these distributions.

Next is a chapter on lossless coding of sequences of discrete symbols. In this chapter the knowledge of information theory for discrete variables and probability distribution estimation is used for practical applications.

The focus then shifts to the background and techniques of lossy coding. Chapter 6 describes rate-distortion theory, which provides bounds on the performance of lossy source coders. While these bounds are obtained for conditions that are not commonly met in the physical world (for example stationarity, the squared error criterion), they provide useful insights for the design of practical lossy source coders. The next chapter discusses high-rate quantization theory. Though, strictly spoken, the high-rate conditions are not satisfied in most practical quantizers, the theory provides insight in how quantizers operate and, moreover, leads to practical design procedures that performe well in many applications, even at low rates. Chapter 8 provides practical procedures for the design of low-rate scalar and vector quantizers, which are commonly used for practical source coders.

The next two chapters focus on signal processing methods to remove redundancy. They describe the basics of the two signal processing procedures most commonly used in source coding: transforms (including filter banks) and linear prediction. In addition to providing the practical implementations for these procedures, the motivation for the usage of the procedures in source coding is given. Particularly in the case of transform coding, there are a number of different motivations for its usage in source coding, and, naturally, these different motivations lead to different flavors of the method.

In the final chapter, chapter 11, the encoding of a speech signal is used as an example application of the various methods described in the earlier chapters. It illustrates that the signal properties are important in the selection of the coding techniques.

1.5 Problems

1. A sampled signal has a bandwidth that is much less than half the sampling frequency. Give at least three alternative methods for coding that are more efficient than straight scalar quantization of the samples.

2. Obviously, it is efficient not to transmit redundancy and irrelevancy. Yet speech contains such redundancies at at least two levels.

   (a) Give a motivation for the existence of redundancy and irrelevancy in a sample sequence representing a speech sound.

(b) Give a motivation for the existence of redundancy and irrelevancy at a linguistic level (the redundancy in a sequence of symbols, each symbol representing a particular speech sound).

3. Using a computer, create a Gaussian signal with sample variance σ^2 and which has a low-pass character with a bandwidth of one quarter of the sampling rate. Solve the following problems using your computer.

   (a) Determine the approximate bit rate required (assuming uniform codeword length) for a uniform scalar quantizer with a maximum amplitude of $4\sigma^2$, and a signal-to-noise ratio of 10 dB.

   (b) Approximate the next sample as the previous quantized sample, multiplied by $\alpha < 1$, plus an error. For $\alpha = 0.9$ again determine the bit rate required for a signal-to-noise ratio of 10 dB.

   (c) Determine the optimal value of α if we can assume that the quantization error is zero. Again determine the bit rate required for a signal-to-noise ratio of 10 dB.

4. Figure 1.3 contains two illustrative sections of the reconstruction-point grids of a two-dimensional quantizer. We consider a mean squared error criterion and a uniform data density.

   (a) Plot the optimal quantizer cells in the grid.

   (b) Argue which quantizer is more efficient (do not consider the effect of the outer, dashed bounds).

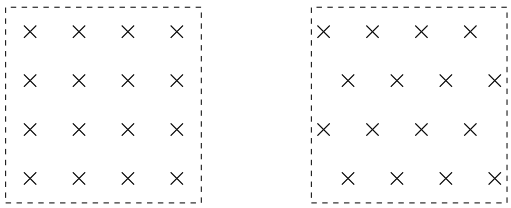

Figure 1.3: The two reconstruction-point grids for problem 4.

2

Introduction to Information Theory: Discrete Variables

2.1 Introduction

Information theory provides results relating to the average bit rate required for the encoding of a random process (a source). Typically, these results provide the lowest achievable average bit rate or a lower bound on the average bit rate. While information-theoretic results are often obtained under particular assumptions that are not always valid or cannot be validated (e.g., the stationarity assumption), they are of great value in the development of practical communication systems.

Information theory is a relatively young discipline. It is generally agreed upon that information theory was born in 1948 with the publication of Shannon's classic papers [3, 4]. Its birth coincides with the invention of the transistor[1]. The transistor made information theory relevant to society by providing us with the computational power to remove redundancy from audio and video signals. Much of the modern telecommunication infrastructure is based on insights gained from information theory.

The basic concepts of information theory have their roots in the ensemble theory of statistical physics, which was developed between 1870 and 1910 by, amongst others, Maxwell, Boltzmann, and Gibbs. This relationship is reflected in the fact that **entropy** plays a central role in both information theory and statistical physics. In the first section of this chapter, we discuss the meaning of entropy in physics in some more detail.

This chapter provides an introduction to the information theory of discrete random variables[2] from the perspective of source coding. Thus, the emphasis is on aspects of information theory that facilitate understanding of coding sampled signals.

[1] Interestingly, both information theory and the transistor were developed in the same building at AT&T Bell Laboratories in New Jersey.

[2] For a brief review of probability theory, see Appendix A.

2.2 Relation to Statistical Physics

We briefly discuss the meaning of entropy in physics before embarking on a discussion of information theory. Entropy was introduced by Clausius in 1850 as a macroscopic quantity useful to the definition of the second law of thermodynamics. Using entropy, this law can be stated as follows: in every process taking place in an isolated system, the entropy of the system either increases or remains constant (e.g., [5, 6]).

Shortly after the introduction of the concept of entropy, the groundwork for statistical physics was laid. The basic premise of statistical physics is that, for an isolated system, all accessible (micro-) states are equally likely. Macroscopic observables, such as pressure and concentration are simply the average values of these quantities over the system states. Boltzmann showed that entropy H and the number of accessible states, Ω, are related by

$$H = k \log(\Omega), \tag{2.1}$$

where k is the Boltzmann constant. In this light, entropy can be interpreted as a **measure of uncertainty** about the state of the system.

Example 2.1: The second law of thermodynamics
The second law states that the entropy of an isolated system cannot decrease. From a statistical-mechanics viewpoint, this means simply that states that correspond to a decrease in entropy are extremely unlikely to occur. Consider, for example, two containers, connected by a small opening, each containing two ideal classical gasses, one with red molecules and one with blue molecules. There are M molecules of each. What is the probability that the gasses spontaneously separate into the two containers? Separation means that for each molecule only half of the total space is available. The probability of having each molecule in a specified container is 2^{-2M}. In contrast, the probability that the molecules are precisely equally distributed over the container is a factor $\left(\frac{M!}{(M/2)!}(M/2)!\right)^2$ larger. In other words, the states corresponding to the mixed situation are much more likely than the states corresponding to separated gasses. Thus, if we start with the gasses separated in each container, and connect the container, then over the time the system will progress towards mixing, and higher entropy, simply because that is much more likely.

To predict physical properties, it is often advantageous to perform an analysis through an **ensemble** of a large number of identical systems, which together form an isolated system. In a canonical ensemble the overall energy of the ensemble is constrained and energy can be exchanged between the systems. In the grand-canonical ensemble the overall energy and the overall number of particles is constrained and energy and particles can be exchanged between the systems. The basic premise of statistical physics is that all possible ensemble configurations have equal probability. To be more precise: all distinguishable and jointly accessible configurations of states of the systems of the ensemble have equal probability.

Let the states of each system form a discrete set, $\mathcal{A} = \{1, 2, \cdots, |\mathcal{A}|\}$. N_1 systems of ensemble are in state 1, N_2 systems are in state 2, etc., with $\sum_{x\in\mathcal{A}} N_x = N$. The numbers N_i/N, $i \in \mathcal{A}$ describe the empirical distribution of the system states. The set of jointly accessible states (and, therefore, the empirical distribution) is constrained by the

ensemble (macroscopic) constraints (the total number of particles in the ensemble, the total energy). Let us evaluate the logarithm of the number of configurations accessible to the entire ensemble (by exchanging energy or energy and particles between the systems of the ensemble) for a given empirical distribution of the system states, normalized to a per system basis:

$$\begin{aligned} \frac{1}{N}\log(\Omega_{p_X}) &= \frac{1}{N}\log\left(\frac{N!}{N_1!\cdots N_{|\mathcal{A}|}!}\right) \\ &\approx \log(N) - \sum_{x\in\mathcal{A}}\frac{N_x}{N}\log(N_x) \\ &= -\sum_{x\in\mathcal{A}} p_X(x)\log(p_X(x)), \end{aligned} \tag{2.2}$$

where we used the Stirling approximation $\log(M!) \approx M\log(M) - M$ (valid only for large integer M), and where X is a random variable with the probability mass function $p_X(x) = N_x/N$. The distribution p_X corresponding to the largest number of configurations (largest entropy) is referred to as the **maximum-entropy distribution**. It can be shown that essentially all accessible ensemble configurations are associated with the maximum-entropy distribution p_X for very large ensembles (large N). As a consequence, the distribution of the system states is unambigious; although we set out with a notion of all ensemble configurations being equally likely, for essentially all observations the empirical probability mass function is the maximum-entropy distribution. Moreover, to predict any macroscopic variables describing a system, we can simply assume that the ensemble has a maximum-entropy distribution. The corresponding entropy is equivalent to the entropy as defined in thermodynamics.

The system states of statistical physics correspond to the symbols of an alphabet in information theory. The generic system used in the ensemble is the random variable. The specific system states correspond to realizations of the random variable. Thus, the ensemble of identical systems in statistical physics corresponds to a sequence of identically distributed random variables. The number of systems is the length of the message. The number of configurations of the ensemble corresponds to the number of possible messages. If the entropy of the system is small, then relatively few configurations of the states in the ensemble are possible. The uncertainties about a symbol and about the message are small. Little information is contained within such a message. If the entropy of the system is large, then the ensemble of systems has many possible configurations of its states. The uncertainty about a symbol and about a sequence of such symbols is then large. More information is contained in the message.

As we have seen above, statistical physics starts from the notion that all accessible configurations of the ensemble are equally likely and this leads to an unambiguous distribution of the system states. If we translate this to communication, this means that we assume all possible messages of a certain (long) length are equally likely and this in turns leads to an unambiguous probability distribution for the alphabet of symbols. In contrast, information theory generally starts from the definition of a random variable with a probability distribution and we do so in section 2.3. We show in section 2.8 that this starting point implies that almost all long sequences are equally likely. Thus, the two approaches are essentially equivalent, but start at opposite ends.

Example 2.2: Simple canonical ensemble
Consider a system consisting of one idealized atom that can take the discrete energy

levels $\{0, 1, 2, \cdots\}$. We create a canonical ensemble by simultaneously considering a very large set of of these atoms that can exchange energy. The macroscopic constraint is that the average energy of the systems in the ensemble is E. The most likely distribution of the energy over the systems in the ensemble is the distribution $p_X(x)$ that maximizes the entropy under the constraint that the mean energy is

$$E = \sum_{x=0}^{\infty} p_X(x)x,$$

where the x are the discrete energy levels. We also have the constraint that $p_X(x)$ is a probability distribution. That is

$$1 = \sum_{x=0}^{\infty} p_X(x)$$

and all $p_X(x)$ must be non-negative. For the derivation we can ignore the second constraint; it can be checked that our answer satisfies it. Using the method of Lagrange multipliers (see Appendix B), we obtain as extended criterion

$$\eta = \sum_{x=0}^{\infty} p_X(x)\log(p_X(x)) + \lambda_1(1 - \sum_{x=0}^{\infty} p_X(x)) + \lambda_2(E - \sum_{x=0}^{\infty} p_X(x)x),$$

where λ_1 and λ_2 are the Lagrange multipliers. Taking the derivative with respect to a particular $p_X(x)$ and setting the result to zero, we obtain

$$0 = \log(p_X(x)) + 1 + \lambda_1 + \lambda_2 x.$$

Since $p_X(x)$ is a probability distribution and must sum to unity, the distribution of the energy of an atom is of the form

$$p_X(x) = \frac{\mathrm{e}^{-\lambda_2 x}}{\sum_{x=0}^{\infty} \mathrm{e}^{-\lambda_2 x}}.$$

This implies that

$$\begin{aligned} E &= \sum_{x=0}^{\infty} p_X(x)\, x \\ &= \frac{\sum_{x=1}^{\infty} x\mathrm{e}^{-\lambda_2 x}}{\sum_{x=0}^{\infty} \mathrm{e}^{-\lambda_2 x}} \\ &= \mathrm{e}^{-\lambda_2} \frac{\sum_{x=1}^{\infty} x\mathrm{e}^{-\lambda_2 (x-1)}}{\sum_{x=0}^{\infty} \mathrm{e}^{-\lambda_2 x}} \\ &= \frac{\mathrm{e}^{-\lambda_2}}{1 - \mathrm{e}^{-\lambda_2}}, \end{aligned}$$

which shows that $\lambda_2 = -\log(\frac{E}{1+E})$. For sufficiently large $E >> 1$ it is reasonable to write $\lambda_2 = \frac{1}{E}$ and we then obtain a so-called canonical distribution:

$$p_X(x) = \frac{\mathrm{e}^{-\frac{x}{E}}}{\sum_{x=0}^{\infty} \mathrm{e}^{-\frac{x}{E}}}.$$

This energy distribution corresponds to the largest number of configurations for the canonical ensemble.

2.3 Information-Theoretic Definition of Entropy

Consider a random variable[3] X that can take any one of a set of values from a discrete alphabet (countable set) $\mathcal{A}$ with probability $p_X(x) \equiv P(X = x)$. The entropy of the random variable in bits is defined as

$$H(X) = -\sum_{x \in \mathcal{A}} p_X(x) \log_2(p_X(x)). \tag{2.3}$$

When using a base-2 logarithm, the entropy is specified in bits. It is often convenient for mathematical manipulation to use the natural logarithm instead. The unit of entropy is then called the **nat**. In this text, we sometimes use bits, and sometimes nats.

Entropy is never less than zero. This follows immediately from the fact that $p_X(x) \in [0, 1]$[4].

The entropy of a random variable can be interpreted as the uncertainty about the random variable prior to observation. Zero entropy implies no uncertainty and a high entropy implies strong uncertainty.

Example 2.3: Entropy of a constant
What is the entropy for a random variable, X, with an alphabet $\mathcal{A}$ containing only one entry, $p_X(x) = 1$? In this case the outcome of an observation is known a-priori, there is no uncertainty, and the entropy is zero (since $\log(p_X(x)) = 0$).

Example 2.4: Entropy of uniform distribution
What is the entropy of a random variable Y with an alphabet $\mathcal{B}$ with N entries and a uniform probability mass over these entries? In this case:

$$H(Y) = -\sum_{y \in \mathcal{B}} \frac{1}{N} \log_2(\frac{1}{N}) = \log_2(N). \tag{2.4}$$

As expected, the entropy, and thus the uncertainty, increases with an increase in the number of possible outcomes of the observations. We note also that if we index the elements of $\mathcal{B}$ by $1, \cdots, N$, and if N is a power of 2, then we can specify each index uniquely with a codeword of $\log_2(N)$ bits. As we will see below, this similarity of the required codeword length and the entropy is no coincidence.

Let us consider a random variable that has not yet been observed. Naturally, we can remove the uncertainty about the variable by specifying a description of its value. That is, the uncertainty about the variable can be removed by specifying its value in some agreed-upon format: a **code**. Such a code can be given in the form of bits and is then called a **binary code**. It would seem that there should be a relation between the entropy of the variable and the effort to specify the value of the variable. Indeed, as will be shown below, if we use a uniquely decodable code (which will be defined precisely later) to describe the variable, the entropy is a lower bound on the minimum required average codeword length. The most efficient codes can get within one bit of the entropy. If we code a sequence of variables in one codeword instead of one at a time, then we can

[3] Following common convention, we use capital letters to indicate random variables.

[4] *Differential entropy*, an information measure for continuous variables, which will be introduced in chapter 3, can be negative.

encode variables at a bit rate that is arbitrarily close to the entropy. Loosely speaking, entropy is the infimum average number of bits required to encode a random variable.

Example 2.5: American Standard Code for Information Interchange
The *American Standard Code for Information Interchange* (ASCII) code is a simple code used to describe keyboard entries. It uses 8 bits to represent the symbols. Only 7 of the 8 bits are actually used.

Since $p_X(X)$ is a random variable, the entropy can also be written as an expectation:

$$H(X) = -\mathrm{E}[\log(p_X(X))]. \tag{2.5}$$

The entity $-\log(p_X(x))$ can be interpreted as the information in nats provided by the observation of a **symbol** associated with an event $X = x$ for the random variable. Thus, events with low probability carry more information than events with high probability. This is consistent with intuition: in guessing someone's name, it is more valuable to be told that a name contains the (low probability) letter "x", than to be told it contains the (high probability) letter "a". The entropy of a random variable is then simply the expectation of the information in the variable.

It is instructive to determine the probability mass function for a random variable X with maximum entropy given the number, N, of possible events in the event set $\mathcal{A}$. In other words, X can be any one of N symbols. First, we find the extremum of the expression $-\sum_{x\in\mathcal{A}} p_X(x)\log(p_X(x))$ as a function of the entities $p_X(x)$ under the constraint that $\sum_{x\in\mathcal{A}} p_X(x) = 1$. (The condition that $p_X(x)$ is nonnegative should also be satisfied, but the solution does this without invoking the constraint.) Using the method of Lagrange multipliers, we obtain as criterion

$$\eta = -\sum_{x\in\mathcal{A}} p_X(x)\log(p_X(x)) - \lambda(1 - \sum_{x_\in\mathcal{A}} p_X(x)), \tag{2.6}$$

where λ is the Lagrange multiplier. Differentiating towards $p_X(x)$ (note that we consider the entity $p_X(x)$ as a variable) and equating the results to zero we obtain

$$\log(p_X(x)) = \lambda - 1 \tag{2.7}$$

for all $x \in \mathcal{A}$. This result is identical for all $p_X(x)$ and this means that all $p_X(x)$ are identical in value at the extremum. Furthermore, the constraint $\sum_{x_\in\mathcal{A}} p_X(x) = 1$, which sums over the N terms, shows that this value must be $1/N$. The corresponding entropy is $H_{ext}(X) = -\sum_{\mathcal{A}} 1/N\log(1/N) = \log(N)$. What remains to be shown is that this solution corresponds to a maximum. This can be done by a second differentiation or by using inequalities; we will not do so here. To summarize, we conclude that, for a given cardinality (number of elements) of the alphabet, the entropy of a random variable is highest for a uniform distribution.

2.4 Entropy and Optimal Codes

Probably the most important aspect of entropy is that it provides bounds on the mean code lengths required to encode a random variable. To be more precise: the entropy

provides bounds on the average length of a **uniquely decodable code**. We define a code as uniquely decodable if any finite concatenation of codewords maps into a unique sequence of source symbols.

Consider a random variable with alphabet $\mathcal{A}$ to be encoded with a uniquely decodable code with codewords of length $l(x)$. (To simplify notation, we will only consider bits in this section.) It is natural to search for the code with the shortest average codeword length, i.e., the code for which

$$L = \mathrm{E}[l(X)] = \sum_{x \in \mathcal{A}} p_X(x) l(x) \tag{2.8}$$

is minimized. The fact that a code is uniquely decodable introduces a constraint on the minimum expected code length. The mathematical formulation of this constraint is called the Kraft inequality, which was first proven in the context of uniquely decodable codes by McMillan [7]. We will show that, given the Kraft inequality, the entropy is a lower bound for L. In a separate proof we can then show that we always can get within one bit of this lower bound.

In the next subsection we describe and prove the Kraft inequality. In subsection 2.4.3 we then prove the inequality relations between entropy and L.

2.4.1 The Kraft Inequality

Our proof of the Kraft inequality follows Karush [8, 7]. As before, x is a realization of a random variable with alphabet $\mathcal{A}$. In the present context, x represents a source symbol. Then, let x^k denote a sequence of k source symbols x and let $\mathcal{A}^k$ denote the associated k-dimensional alphabet of source symbols (created from all possible concatenations of k source symbols x). The codeword length (we measure in bits) for a particular sequence of k symbols is denoted by $l(x^k)$. The codeword lengths of the individual symbols and those of the entire sequence of k concatenated symbols satisfy

$$(\sum_{x^1 \in \mathcal{A}} 2^{-l(x^1)})^k \quad = \quad \sum_{x^k \in \mathcal{A}^k} 2^{-l(x^k)}. \tag{2.9}$$

Equation 2.9 is simple to interpret. Let us denote the cardinality of $\mathcal{A}$ by $|\mathcal{A}|$. Then, there are $|\mathcal{A}|^k = |\mathcal{A}^k|$ terms on the right-hand side of equation 2.9, each corresponding to a particular sequence of k codewords from $\mathcal{A}$. The exponent of each term just adds (minus) the length of all k codewords corresponding to the sequence to render (minus) the overall length of the particular codeword sequence.

Example 2.6: Illustration of equation 2.9
Consider a random variable with an alphabet of cardinality three. We use a code with codewords 0, 10, 11. We then have

$$\sum_{x^1 \in \mathcal{A}} 2^{-l(x^1)} \quad = \quad 2^{-1} + 2 \cdot 2^{-2}.$$

Now consider the concatenation of two source symbols. We then have

$$\begin{aligned} (\sum_{x^1 \in \mathcal{A}} 2^{-l(x^1)})^2 \quad &= \quad (2^{-1} + 2 \cdot 2^{-2})(2^{-1} + 2 \cdot 2^{-2}) \\ &= \quad 2^{-2} + 4 \cdot 2^{-3} + 4 \cdot 2^{-4}, \end{aligned}$$

which corresponds to the codewords 00, 010, 011, 100, 1010, 1011, 110, 1110 1111. The terms gather codewords of identical length.

Defining $l_{\max} = \max_{x\in\mathcal{A}}(l(x^1))$, we can also write for a concatenation of k symbols

$$(\sum_{x^1\in\mathcal{A}} 2^{-l(x^1)})^k = \sum_{x^k\in\mathcal{A}^k} 2^{-l(x^k)} = \sum_{m=1}^{kl_{\max}} c(m)2^{-m}, \tag{2.10}$$

where the $c(m)$ are constants. Each $c(m)$ in equation 2.10 just represents the number of symbol sequences with concatenated codeword length m. However, by the definition of a uniquely decodable code, each concatenation of codewords of length l must be uniquely decodable. This implies that a sequence of m bits can represent no more than 2^m symbol sequences, i.e., $c(m) \leq 2^m$. Thus:

$$(\sum_{x^1\in\mathcal{A}} 2^{-l(x^1)})^k \leq \sum_{m=1}^{kl_{\max}} 2^m 2^{-m} = kl_{\max}. \tag{2.11}$$

Taking the k'th root of both sides of this equation provides the inequality

$$\sum_{x^1\in\mathcal{A}} 2^{-l(x^1)} \leq (kl_{\max})^{\frac{1}{k}}, \tag{2.12}$$

which is true for all k. The bound is tightest for the case where k approaches infinity, when the right hand side converges to unity. Thus, we have the following theorem:

Theorem 1 *Kraft inequality: for a uniquely decodable code, with codeword lengths $l(x)$*

$$\sum_{x\in\mathcal{A}} 2^{-l(x)} \leq 1. \tag{2.13}$$

2.4.2 How to Construct a Code Satisfying the Kraft Inequality

It is always possible to define a uniquely decodable code for a set of lengths $\{l(x)\}$ satisfying the Kraft inequality. This can be shown by construction of an actual code. We use a so-called **prefix** or **instantaneous code**. A prefix is simply an initial segment of a codeword. A prefix code is a code where a codeword cannot be a prefix to another codeword.

Example 2.7: A prefix code
Consider a code with three codewords 0, 10, 11. The shorter codeword 0 is not a prefix to the codewords 10 and 11, and the code is therefore a prefix code.

Consider a set of codeword lengths $\{l_i\} = l_1, \cdots, l_m$ that satisfy the Kraft inequality. We assume our code to be specified in bits and, without loss of generality, we also assume that the codeword lengths are ordered: $l_1 \leq l_2 \cdots \leq l_m$. We note that the Kraft inequality can be rewritten as

$$\sum_{x\in\mathcal{A}} 2^{l_m - l(x)} \leq 2^{l_m}. \tag{2.14}$$

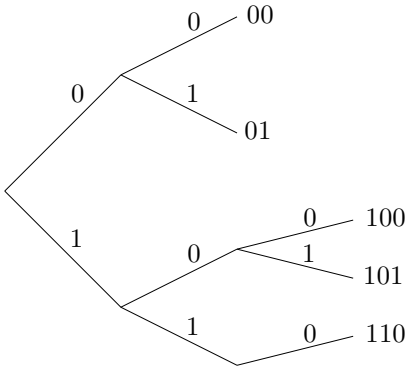

Figure 2.1: Binary tree and corresponding binary code.

Equation 2.14 facilitates the visualization of the code in a tree format. Consider a binary tree with each branching representing a bit and a codeword being the concatenation of the bits along the branches as shown in figure 2.1. Now consider a codeword of length $l(x)$. The term $2^{l_m - l(x)}$ represents the number of codewords of length l_m for which this codeword is a prefix. We will call these codewords of length l_m the descendants of the codeword of length $l(x)$. The Kraft inequality then states that the sum of the number of descendants of length l_m of all codewords in a uniquely-decodable code must be less than 2^{l_m}. A code satisfying this constraint is easily constructed with the binary tree structure. Figure 2.1 shows how to create a uniquely decodable code for the given codeword lengths $2, 2, 3, 3, 3$.

2.4.3 The Source-Coding Theorem

Using the Kraft inequality, we can now show that a uniquely decodable code can be no shorter in expectation than the entropy. We first define

$$p_L(x) \equiv \frac{2^{-l(x)}}{\sum_{x \in \mathcal{A}} 2^{-l(x)}} \tag{2.15}$$

and note that this quantity is a probability mass function since it sums to unity. In the derivation we will exploit two inequalities: $\log(x) \leq x - 1$ (or $\log_2(x) \leq (x-1)/\log(2)$)

and the Kraft inequality in the form $\log_2(\sum_{x\in\mathcal{A}} 2^{-l(x)}) \leq 0$. Then we have

$$\begin{aligned}
L &= \sum_{x\in\mathcal{A}} p_X(x)l(x) \\
&= -\sum_{x\in\mathcal{A}} p_X(x)\log_2(2^{-l(x)}) \\
&\geq -\sum_{x\in\mathcal{A}} p_X(x)\log_2(2^{-l(x)}) + \sum_{x\in\mathcal{A}} p_X(x)\log_2(\sum_{x\in\mathcal{A}} 2^{-l(x)}) \\
&= -\sum_{x\in\mathcal{A}} p_X(x)\log_2(p_L(x)) \\
&= H(X) - \sum_{x\in\mathcal{A}} p_X(x)\log_2(\frac{p_L(x)}{p_X(x)}) \\
&\geq H(X) - \sum_{x\in\mathcal{A}} p_X(x)\left(\frac{p_L(x)}{p_X(x)} - 1\right)/\log(2) \\
&= H(X) - \left(\sum_{x\in\mathcal{A}} p_L(x) - \sum_{x\in\mathcal{A}} p_X(x)\right)/\log(2) \\
&= H(X). \qquad (2.16)
\end{aligned}$$

We see thus that a uniquely decodable code cannot be shorter, on average, than the entropy.

Next, we show that a uniquely decodable code can, in fact, get within 1 bit of the entropy for an encoding in bits. To this purpose, we select a code with $l(x) = \lceil -\log_2(p_X(x))\rceil$, where $\lceil\cdot\rceil$ rounds to the next higher integer. Such a code satisfies the Kraft inequality:

$$\begin{aligned}
\sum_{x\in\mathcal{A}} 2^{-l(x)} &= \sum_{x\in\mathcal{A}} 2^{-\lceil -\log_2(p_X(x))\rceil} \\
&\leq \sum_{x\in\mathcal{A}} 2^{\log_2(p_X(x))} \\
&= \sum_{x\in\mathcal{A}} p_X(x) = 1, \qquad (2.17)
\end{aligned}$$

and can, therefore, be made to be a uniquely decodable code using the method provided in section 2.4.2. This type of uniquely-decodable code is called a **Shannon code** and will be discussed in more detail in section 5.2.1. For a Shannon code we have

$$\begin{aligned}
L &= \sum_{x\in\mathcal{A}} p_X(x)l(x) \\
&= \sum_{x\in\mathcal{A}} p_X(x)\lceil -\log(p_X(x))\rceil \\
&< -\sum_{x\in\mathcal{A}} p_X(x)\log(p_X(x)) + 1 \\
&= H(X) + 1. \qquad (2.18)
\end{aligned}$$

From this inequality and inequality 2.16 we obtain the **source coding theorem**:

Theorem 2 *The uniquely decodable code that minimizes the average codeword length, L, satisfies*

$$H(X) \leq L < H(X) + 1. \tag{2.19}$$

Note that the proof of this theorem was constructive and that Shannon codes can be used for practical applications (although there are better codes).

At first sight, the upper bound $H(X) + 1$ may not seem very tight for random variables with an entropy of, say, 1 bit or less. However, this problem is easily mitigated simply by combining symbol sequences into new symbols of higher entropy.

2.5 Other Entropy Measures

2.5.1 Joint Entropy

The definition of the entropy of a random variable is valid independently of the dimensionality of the variable. Thus, the joint entropy of a set of k random variables, $\{X_i\}_{0<i\leq k}$, is implicitly defined by our first entropy definition. However, for practical purposes, it is convenient to define a notation for the **joint entropy** of a set of random variables:

$$\begin{aligned} H(X_1, \cdots, X_k) &= -\mathrm{E}[\log(p_{X_1 \cdots X_k}(X_1, \cdots, X_k))] \\ &= -\sum_{x_1 \in \mathcal{A}_1} \cdots \sum_{x_k \in \mathcal{A}_k} p_{X_1 \cdots X_k}(x_1, \cdots, x_k) \log(p_{X_1 \cdots X_k}(x_1, \cdots, x_k)), \end{aligned}$$

where we used the alphabets $\mathcal{A}_i$ and where we note that the subscripts label the different variables.

Next, we prove the following theorem:

Theorem 3 *The joint entropy of a set of variables can never be more than the sum of the entropies of the individual variables:*

$$H(X_1, \cdots, X_k) \leq H(X_1) + \cdots + H(X_k). \tag{2.20}$$

We will first prove the theorem for two variables. Using nats as units and the fact that

$\log(x) \leq x - 1$, we have:

$$\begin{aligned}
H(X,Y) &= -\sum_{x\in\mathcal{A}}\sum_{y\in\mathcal{B}} p_{XY}(x,y)\log(p_{XY}(x,y)) \\
&= -\sum_{x\in\mathcal{A}} p_X(x)\log(p_X(x)) - \sum_{y\in\mathcal{B}} p_Y(y)\log(p_Y(y)) \\
&\qquad - \sum_{x\in\mathcal{A}}\sum_{y\in\mathcal{B}} p_{XY}(x,y)\log(\frac{p_{XY}(x,y)}{p_X(x)p_Y(y)}) \\
&= H(X) + H(Y) + \sum_{x\in\mathcal{A}}\sum_{y\in\mathcal{B}} p_{XY}(x,y)\log(\frac{p_X(x)p_Y(y)}{p_{XY}(x,y)}) \\
&\leq H(X) + H(Y) + \sum_{x\in\mathcal{A}}\sum_{y\in\mathcal{B}} p_{XY}(x,y)\left(\frac{p_X(x)p_Y(y)}{p_{XY}(x,y)} - 1\right) \\
&= H(X) + H(Y) + \sum_{x\in\mathcal{A}}\sum_{y\in\mathcal{B}} p_X(x)p_Y(y) - \sum_{x\in\mathcal{A}}\sum_{y\in\mathcal{B}} p_{XY}(x,y) \\
&= H(X) + H(Y). \qquad (2.21)
\end{aligned}$$

By recursive application of this method, we can prove the theorem for any number of variables.

Note that inequality 2.21 becomes an equality if $p_{XY}(x,y) = p_X(x)p_Y(y)$, i.e., if the variables X and Y are independent. This suggests that, if coding is to be performed individually on a set of variables, it is useful to search for a transform that results in variables that are independent. The inverse transform can then be performed upon decoding. (Transform coding will be the topic of chapter 9.)

Example 2.8: Joint entropy of dependent variables
Let us compute the joint entropies $H(X_1, X_2)$ and $H(X_1, Y = X_1 + X_2)$ and compare them to the entropy sums. X_1 and X_2 are independent with probability mass functions $p_{X_1}(0) = p_{X_1}(1) = 0.5$ and $p_{X_2}(0) = p_{X_2}(1) = 0.5$. From example 2.4, we know that $H(X_1) = H(X_2) = 1$ bit. The vector $[X_1, X_2]$, which has four states of equal probability, has entropy $H([X_1, X_2]) = H(X_1, X_2) = 2$ bits and this equals the sum of the entropies of the individual variables.

The vector Y takes the values $0, 1, 2$ with the probability mass function $p_Y(0) = 0.25, p_Y(1) = 0.5, p_Y(2) = 0.25$. Thus $H(Y) = 0.25 * 2 + 0.5 * 1 + 0.25 * 2 = 1.5$ bits. The probability mass function of the vector $[X_1, Y]$ is

$$\begin{aligned}
p_{X_1Y}(0,0) &= p_{X_1X_2}(0,0) = 0.25 \\
p_{X_1Y}(0,1) &= p_{X_1X_2}(0,1) = 0.25 \\
p_{X_1Y}(0,2) &= 0 \\
p_{X_1Y}(1,0) &= 0 \\
p_{X_1Y}(1,1) &= p_{X_1X_2}(1,0) = 0.25 \\
p_{X_1Y}(1,2) &= p_{X_1X_2}(1,1) = 0.25
\end{aligned}$$

which leads to the joint entropy $H([X_1, Y]) = H(X_1, Y) = 2$ bits. This is significantly less than $H(X_1) + H(Y) = 2.5$ bits.

The results illustrate theorem 3. The sum of the entropies of independent variables equals their joint entropy. For dependent variables the joint entropy is less than the sum of the entropies of the variables.

2.5.2 Conditional Entropy

The **conditional entropy** is defined as

$$H(Y|X) \equiv -\mathrm{E}[\log(p_{Y|X}(Y|X))] = -\sum_{x\in\mathcal{A}}\sum_{y\in\mathcal{B}} p_{XY}(x,y)\log(p_{Y|X}(y|x)). \tag{2.22}$$

A first interpretation of the conditional entropy can be obtained from writing it in the following form:

$$\begin{aligned} H(Y|X) &= -\sum_{x\in\mathcal{A}}\sum_{y\in\mathcal{B}} p_{XY}(x,y)\log(p_{Y|X}(y|x)) \\ &= -\sum_{x\in\mathcal{A}} p_X(x)\sum_{y\in\mathcal{B}} p_{Y|X}(y|x)\log(p_{Y|X}(y|x)) \\ &= \sum_{x\in\mathcal{A}} p_X(x)H(Y|X=x). \end{aligned} \tag{2.23}$$

Thus, the conditional entropy $H(Y|X)$ corresponds to the entropy of the random variable Y given that X is $X = x$, averaged over the alphabet $\mathcal{A}$ of X. More informally, the conditional entropy $H(Y|X)$ is the average uncertainty in Y after observation of X. This viewpoint suggests that the conditional entropy $H(Y|X)$ equals the entropy $H(Y)$ for the case that Y and X are independent.

From the definitions of joint and conditional entropy we note that

$$\begin{aligned} H(X,Y) &= -\mathrm{E}[\log(p_{XY}(X,Y))] \\ &= -\mathrm{E}[\log(p_{Y|X}(Y|X)) + \log(p_X(X))] \\ &= -\mathrm{E}[\log(p_{Y|X}(Y|X))] - \mathrm{E}[\log(p_X(X))] \\ &= H(Y|X) + H(X). \end{aligned} \tag{2.24}$$

Equation 2.24 allows a second interpretation of the conditional entropy, since $H(Y|X) = H(X,Y) - H(X)$. In words: the conditional entropy $H(Y|X)$ is the remainder uncertainty after the uncertainty of the random variable X has been subtracted from the joint uncertainty in the random variables X and Y.

It follows immediately from theorem 3 and equation 2.24 that conditioning never increases the entropy, an intuitive result:

Theorem 4 *Conditional entropy is bound from above by the corresponding unconditional entropy:*

$$H(Y|X) \leq H(Y). \tag{2.25}$$

The inequality in theorem 4 becomes an equality when X and Y are independent.

Example 2.9: Conditioning
What is the conditional entropy of a random parameter given itself? We note that $H(X,X) = H(X)$. Thus, we have that $H(X|X) = 0$.

The so-called **chain rule** is particularly useful for the study of random processes. By recursive application of equation 2.24 we see that, for a sample block $X_1, X_2, \cdots, X_k$ we have

$$\begin{aligned} H(X_1, X_2, \cdots, X_k) &= H(X_1) + H(X_2|X_1) + \sum_{i=3}^{k} H(X_i|X_1, \cdots, X_{i-1}) \\ &= \sum_{i=1}^{k} H(X_i|X_1, \cdots, X_{i-1}). \end{aligned} \tag{2.26}$$

If the samples are independent, then the conditionals can be dropped, and the joint entropy is just the sum of the entropies of the samples.

The addition of independent random variables is common and where such additions occur, a simple relation between entropy and conditional entropy is often helpful. Let us consider the random variables X, Y, and Z with alphabets $\mathcal{A}$, $\mathcal{B}$, and $\mathcal{C}$ respectively. Z is the addition of the independent variables X and Y: $Z = X + Y$. The following relation then holds:

$$\begin{aligned} H(X+Y|Y) = H(Z|Y) &= -\sum_{y\in\mathcal{B}}\sum_{z\in\mathcal{C}} p_{ZY}(z,y)\log(p_{Z|Y}(z|y)) \\ &= -\sum_{y\in\mathcal{B}} p_Y(y)\sum_{z\in\mathcal{C}} p_{Z|Y}(z|y)\log(p_{Z|Y}(z|y)) \\ &= -\sum_{y\in\mathcal{B}} p_Y(y)\sum_{x\in\mathcal{A}} p_{Z|Y}(x+y|y)\log(p_{Z|Y}(x+y|y)) \\ &= -\sum_{y\in\mathcal{B}} p_Y(y)\sum_{x\in\mathcal{A}} p_X(x)\log(p_X(x)) \\ &= \left(\sum_{y\in\mathcal{B}} p_Y(y)\right) H(X) = H(X). \end{aligned} \tag{2.27}$$

2.5.3 Entropy Rate and Redundancy Rate

In the coding of random processes, we deal with sequences of random variables (the samples). We can use the concept of joint entropy to study the properties of these sequences. The theory is usually limited to (strict-sense) stationary processes and we will also make this assumption. For a process it is often useful to consider not only the entropy of an individual sample, which is referred to as the **first-order entropy** of the process, $H_1(X_i) \equiv H(X_i)$, but also the entropy of blocks of samples. The **order-k entropy** is defined as

$$H_k(X_i) \equiv \frac{1}{k} H(X_{i+1}, \cdots, X_{i+k}) = \frac{1}{k} H(X_1, \cdots, X_k), \tag{2.28}$$

where we showed explicitly that the block selection is immaterial because of the stationarity assumption. The limiting case of this sequence of entropies is called the **entropy rate**:

$$H_\infty(X_i) \equiv \lim_{k\to\infty} \frac{1}{k} H(X_1, \cdots, X_k). \tag{2.29}$$

Next, we show that the order-k entropy is a monotonically nonincreasing function of k. Since the conditioned entropy of a random variable can not be more than the unconditioned entropy of a random variable, we see that for a stationary process X_i we have

$$H(X_k|X_1,\cdots,X_{k-1}) \leq H(X_i|X_1,\cdots,X_{i-1}), \quad i \leq k, \tag{2.30}$$

and $H(X_k|X_1,\cdots,X_{k-1}) \leq H(X_1)$. Using these properties, we can prove that the per-sample average entropy is a nonincreasing function of the order (to simplify notation, conditioning is not active for reverse-time order of the conditioning variables, i.e., $H(X_1|X_1,X_0) = H(X_1)$),

$$\begin{aligned} H_k(X_i) &= \frac{1}{k}\sum_{j=1}^{k} H(X_j|X_1,\cdots,X_{j-1}) \\ &= \frac{1}{k-1}\sum_{j=1}^{k-1} H(X_j|X_1,\cdots,X_{j-1}) \\ &\quad -\frac{1}{k}\left(\frac{1}{k-1}\sum_{j=1}^{k-1} H(X_j|X_1,\cdots,X_{j-1}) - H(X_k|X_1,\cdots,X_{k-1})\right) \\ &= H_{k-1}(X_i) - \frac{1}{k}\frac{1}{k-1}\sum_{j=1}^{k-1}\left(H(X_j|X_1,\cdots,X_{j-1}) - H(X_k|X_1,\cdots,X_{k-1})\right) \\ &\leq H_{k-1}(X_i), \end{aligned} \tag{2.31}$$

since the terms in the last summation are all nonnegative. Thus, the order-k entropy for samples of a stationary signal generally decreases with increasing block size. This is intuitively reasonable since the samples may share information, i.e., they may be statistically dependent.

Inequality 2.31 also shows that the order-k entropies form a decreasing sequence of nonnegative numbers, which must have a limit. It then follows from theorem 2 that the average codeword length normalized to a per sample basis can be made to be arbitrarily close to the entropy rate by increasing the block size, k, sufficiently. Coding at a higher average codeword length per sample (average bit rate) implies redundancy.

We have now seen that the order-k entropy is nonincreasing with increasing order (inequality 2.31) and we saw earlier in theorem 4 that the entropy is nonincreasing with increasing conditioning. It is useful to connect the bounds of the two sequences for a stationary process. For ease of notation let us write $H_c \equiv H(X_i|X_{-\infty},\cdots,X_{i-1})$. Then we can write (again, to simplify notation conditioning is not active for reverse-time order

of the conditioning variables)

$$\begin{aligned} H_k(X_i) &= \frac{1}{k}\sum_{i=1}^{k} H(X_i|X_1,\cdots,X_{i-1}) \\ &= \frac{1}{k}\sum_{i=1}^{k} H(X_1|X_{-i+2},\cdots,X_0) \\ &= \frac{1}{k}\sum_{i=1}^{N} H(X_1|X_{-i+2},\cdots,X_0) + \frac{1}{k}\sum_{i=N+1}^{k} H(X_1|X_{-i+2},\cdots,X_0) \\ &= \frac{1}{k}\sum_{i=1}^{N} H(X_1|X_{-i+2},\cdots,X_0) + \frac{k-N}{k}H_c + \\ &\quad \frac{1}{k}\sum_{i=N+1}^{k} (H(X_1|X_{-i+2},\cdots,X_0) - H_c) \end{aligned} \tag{2.32}$$

Now let us consider the individual terms on the right-hand side of equation 2.32 for the case that $k \to \infty$ and N is finite. The term $\frac{1}{k}\sum_{i=1}^{N} H(X_1|X_{-i+2},\cdots,X_0)$ vanishes because of the $1/k$ factor. The term $\frac{k-N}{k}H_c$ becomes H_c when k approaches infinity. Finally, since the terms $H(X_1|X_{-i+2},\cdots,X_0)$ approach H_c with increasing i, we can always find an N that makes the term $\frac{1}{k}\sum_{i=N+1}^{k}(H(X_1|X_{-i+2},\cdots,X_0) - H_c)$ smaller than any arbitrary small number. We write the result as a theorem.

Theorem 5 *For a stationary sequence the entropy rate equals the sample entropy conditioned by the infinite past.*

$$H_\infty(X_i) = H(X_1|X_{-\infty},\cdots,X_0). \tag{2.33}$$

Having defined a lower bound for the bit rate required for encoding a stationary signal, it is straightforward to define an excess bit rate required when the dependencies between samples are not accounted for. The **redundancy rate** (the redundancy per sample) of a stationary signal is defined as the difference between the entropy and the entropy rate:

$$\rho(X_i) = H_1(X_i) - H_\infty(X_i). \tag{2.34}$$

Example 2.10: Entropy and redundancy rate of iid process
It is interesting to consider the relation between $H_\infty(X_i)$ and $H_1(X_i)$ for a signal consisting of independent, identically distributed (iid) samples with a discrete alphabet $\mathcal{A}$. Theorem 5 immediately shows that $H_\infty(X_i) = H(X_1)$. In this case, this is also easily obtained directly from the defining equation for entropy rate

(equation 2.29):

$$\begin{aligned}
H_\infty(X_i) &\equiv \lim_{k\to\infty}\frac{1}{k}H(X_1,\cdots,X_k) \\
&= -\lim_{k\to\infty}\frac{1}{k}\sum_{x_1\in\mathcal{A}}\cdots\sum_{x_k\in\mathcal{A}} p_{X_1\cdots X_k}(x_1,\cdots,x_k)\log(p_{X_1\cdots X_k}(x_1,\cdots,x_k)) \\
&= -\lim_{k\to\infty}\frac{1}{k}\sum_{x_1\in\mathcal{A}}\cdots\sum_{x_k\in\mathcal{A}} p_{X_1}(x_1)\cdots p_{X_k}(x_k)\log(p_{X_1}(x_1)\cdots p_{X_k}(x_k)) \\
&= -\lim_{k\to\infty}\frac{1}{k}\sum_{i=1}^{k}\sum_{x_i\in\mathcal{A}} p_{X_i}(x_i)\log(p_{X_i}(x_i)) \\
&= \lim_{k\to\infty}\frac{1}{k}\sum_{i=1}^{k}H_1(X_i) \\
&= H_1(X_i).
\end{aligned}$$

Thus, if the samples are iid distributed, then the first-order entropy is equal to the entropy rate, and the redundancy rate is zero. In fact, all higher-order entropies are equal to the first-order entropy in this case.

Example 2.11: Entropies of Markov chain process
In a first-order Markov process (denoted here by X_i), the system state at time i

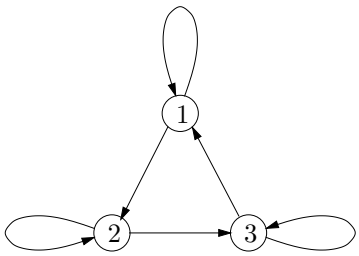

Figure 2.2: The three-state Markov process of example 2.11.

depends only on the system state at time $i-1$, i.e.,

$$P(X_i = x_i|X_{i-1} = x_{i-1}, X_{i-2} = x_{i-2},\cdots) = P(X_i = x_i|X_{i-1} = x_{i-1}).$$

Let us consider a stationary three-state Markov process illustrated in figure 2.2. The transition probabilities between the states are specified by the state transition matrix,

$$T = \begin{bmatrix} 0.5 & 0.0 & 0.5 \\ 0.5 & 0.5 & 0.0 \\ 0.0 & 0.5 & 0.5 \end{bmatrix},$$

where $T_{nm} = P(X_i = n|X_{i-1} = m)$.

We want to compute the $H(X_i)$, $H_2(X_i)$, and $H_\infty(X_i)$ of the three-state Markov process. We first conclude from symmetry of the transition matrix that the problem is symmetric in the states, and that, therefore, all states must have equal

probability. Using the basic formula for entropy with probability 1/3 for all states gives $H(X_i) = \log_2(3)$. Furthermore, we note

$$\begin{aligned} H_k(X_i) &= \frac{1}{k}\sum_{i=1}^{k} H(X_i|X_1,\cdots,X_{i-1}) \\ &= \frac{1}{k}H(X_1) + \frac{1}{k}\sum_{i=2}^{k} H(X_i|X_{i-1}) \\ &= \frac{1}{k}H(X_1) + \frac{k-1}{k}H(X_i|X_{i-1}). \end{aligned}$$

The conditional entropy is easy to evaluate:

$$\begin{aligned} H(X_i|X_{i-1}) &= -\sum_{x_i\in\mathcal{A},x_{i-1}\in\mathcal{A}} p_{X_iX_{i-1}}(x_i,x_{i-1})\log(p_{X_i|X_{i-1}}(x_i|x_{i-1})) \\ &= -\sum_{x_i\in\mathcal{A},x_{i-1}\in\mathcal{A}} p_{X_{i-1}}(x_{i-1})p_{X_i|X_{i-1}}(x_i|x_{i-1})\log(p_{X_i|X_{i-1}}(x_i|x_{i-1})) \\ &= -\sum_{x_i\in\mathcal{A}} p_{X_i|X_{i-1}}(x_i|x_{i-1}=1)\log(p_{X_i|X_{i-1}}(x_i|x_{i-1}=1)) \\ &= -\log_2(0.5) = 1, \end{aligned}$$

where we denoted the set of states by $\mathcal{A}$ and used the state symmetry extensively. Combining this result with equation 2.33 we obtain

$$\begin{aligned} H_2(X_i) &= \frac{1}{2}H(X_1) + \frac{1}{2}H(X_i|X_{i-1}) = \frac{1}{2}\log_2(3) + \frac{1}{2}, \\ H_\infty(X_i) &= H(X_i|X_{i-1}) = 1. \end{aligned}$$

2.6 Mutual Information

So-far, we have defined entropy, joint entropy, and conditional entropy. Next we define **mutual information**, a quantity that is central to information theory and particularly to rate-distortion theory (to be discussed in chapter 6). The mutual information between two random parameters X and Y is defined as

$$\begin{aligned} I(X;Y) &\equiv H(X) + H(Y) - H(X,Y) \\ &= H(X) - H(X|Y) \\ &= H(Y) - H(Y|X). \end{aligned} \tag{2.35}$$

Recall that $H(Y|X)$ is the expectation of the uncertainty in Y after observation of X. Thus, the mutual information $I(X;Y)$ is the expectation of the reduction in the uncertainty of Y after observation of X or vice versa.

Using the definitions of entropy and mutual information, it is possible to express mutual

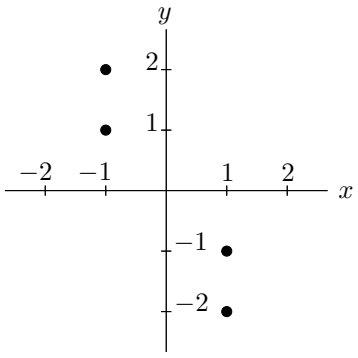

Figure 2.3: Probability mass function of example 2.12. Each point has probability 1/4.

information in terms of the probability mass functions:

$$\begin{aligned} I(X;Y) &\equiv H(X) + H(Y) - H(X,Y) \\ &= -\sum_{x\in\mathcal{A}} p_X(x)\log(p_X(x)) - \sum_{y\in\mathcal{B}} p_Y(y)\log(p_Y(y)) + \\ &\quad \sum_{x\in\mathcal{A}}\sum_{y\in\mathcal{B}} p_{XY}(x,y)\log(p_{XY}(x,y)) \\ &= \sum_{x\in\mathcal{A}}\sum_{y\in\mathcal{B}} p_{XY}(x,y)\log(\frac{p_{XY}(x,y)}{p_X(x)p_Y(y)}). \end{aligned} \tag{2.36}$$

Example 2.12: Mutual information of a discrete distribution
We compute the mutual information of the probability mass function in figure 2.3:

$$\begin{aligned} I(X;Y) &= \sum_{x=1}^{2}\sum_{y=1}^{2} p_{XY}(x,y)\log_2(\frac{p_{XY}(x,y)}{p_X(x)p_Y(y)}) \\ &= 4\frac{1}{4}\log_2(\frac{\frac{1}{4}}{\frac{1}{2}\frac{1}{4}}) \\ &= 1. \end{aligned}$$

This mutual information is 1 bit. This is intuitive since, if we know X we only know if Y is positive or negative, but no more. Similarly, if we know Y, we know whether X is -1 or +1, which again is 1 bit.

We can also find the mutual information by computing

$$\begin{aligned} H(X) &= \log_2(2) = 1 \\ H(Y) &= \log_2(4) = 2 \\ H(X,Y) &= \log_2(4) = 2 \end{aligned}$$

and using these results to obtain

$$\begin{aligned} I(X;Y) &= H(X) + H(Y) - H(X,Y) \\ &= 1 + 2 - 2 = 1. \end{aligned}$$

Similarly to entropy, mutual information is always nonnegative and this property is often used. This result immediately follows from the definition of mutual information in equation 2.36 and theorem 3. To emphasize its importance, we write this result as a theorem:

Theorem 6 *The mutual information of two random variables is always nonnegative,*

$$I(X;Y) \geq 0. \tag{2.37}$$

For clarity, we note that the nonnegativity of mutual information does not carry over to mutual information of more variables (using the standard definition). A direct proof of the nonnegativity of mutual information using probability mass functions is simple:

$$\begin{aligned} I(X;Y) &= \sum_{x\in\mathcal{A}}\sum_{y\in\mathcal{B}} p_{XY}(x,y)\log(\frac{p_{XY}(x,y)}{p_X(x)p_Y(y)}) \\ &= -\sum_{x\in\mathcal{A}}\sum_{y\in\mathcal{B}} p_{XY}(x,y)\log(\frac{p_X(x)p_Y(y)}{p_{XY}(x,y)}) \\ &\geq -\sum_{x\in\mathcal{A}}\sum_{y\in\mathcal{B}} p_{XY}(x,y)(\frac{p_X(x)p_Y(y)}{p_{XY}(x,y)} - 1) \\ &= -\sum_{x\in\mathcal{A}}\sum_{y\in\mathcal{B}} p_X(x)p_Y(y) + \sum_{x\in\mathcal{A}}\sum_{y\in\mathcal{B}} p_{XY}(x,y) \\ &= -1 + 1 = 0, \end{aligned} \tag{2.38}$$

where we used $\log(x) \leq x - 1$.

The derivation in 2.38 shows that the inequality becomes an equality if the variables are independent, i.e., when $p_{XY}(x,y) = p_X(x)p_Y(y)$. This is quite intuitive; independent variables do not share mutual information. The fact that mutual information is nonnegative forms a convenient basis for proving many other properties, including some of the properties we derived earlier.

For completeness, we mention that it is sometimes useful to define **conditional mutual information**:

$$I(X;Y|W) \equiv H(X|W) + H(Y|W) - H(X,Y|W). \tag{2.39}$$

The understanding of the relation between the various information-theoretic measures is facilitated by the bubble diagram shown in figure 2.4. It is important to realize that the bubble diagram illustration is correct only for two variables.

2.7 Relative Entropy

We now introduce the definition of the **relative entropy** (also known as **Kullback-Leibler distance**). The relative entropy is a measure of the dissimilarity of two probability mass functions, say p_X and p_Y:

$$H(p_X||p_Y) \equiv \sum_{x\in\mathcal{A}} p_X(x)\log(\frac{p_X(x)}{p_Y(x)}). \tag{2.40}$$

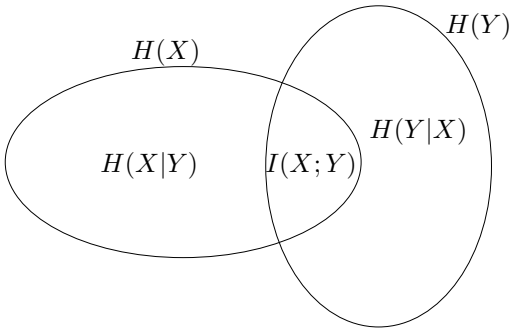

Figure 2.4: Bubble diagram showing various measures of information theory. The left bubble represents $H(X)$, the right bubble $H(Y)$; their union is the joint entropy $H(X,Y)$, and their intersection is the mutual information $I(X;Y)$. *Note: this diagram does not generalize to more variables.*

Note that the name Kullback-Leibler *distance* is misleading: in general $H(p_X||p_Y)$ does not equal $H(p_Y||p_X)$. Comparing equations 2.36 and 2.40, it is seen that mutual information can be written as a relative entropy:

$$I(X;Y) = H(p_{XY}||p_X p_Y). \tag{2.41}$$

Let us consider a given probability mass function $p_X(x)$; which probability mass function $p_Y(x)$ minimizes the relative entropy? We will provide two solutions, the first using regular calculus. Using the calculus-based approach, we aim to find the discrete function $p_Y(x)$ that makes $\sum_{x\in\mathcal{A}} p_X(x)\log(\frac{p_X(x)}{p_Y(x)})$ an extremum, under the constraint that

$$\sum_{x\in\mathcal{A}} p_Y(x) = 1. \tag{2.42}$$

Using Lagrange multipliers, we want to find the extremum of the extended criterion

$$\eta = \sum_{x\in\mathcal{A}} \left[p_X(x)\log(\frac{p_X(x)}{p_Y(x)}) + \lambda p_Y(x) \right], \tag{2.43}$$

where we opted to leave out the constant term. Differentiating equation 2.43 towards $p_Y(x)$ results in

$$\frac{p_X(x)}{p_Y(x)} = \lambda. \tag{2.44}$$

Combining this with the constraint, we see that the relative entropy takes an extreme value for $p_Y(x) = p_X(x)$. We note that the relative entropy takes the value zero at this extremum. To show that this is a minimum one can use inequalities, and this leads to an altogether more elegant approach for finding the minimum, where we can omit the calculus based approach.

We now show the strict nonnegativity of the relative entropy with inequalities. We use

natural logarithms and the familiar property $\log(x) \leq x - 1$:

$$\begin{aligned} H(p_X||p_Y) &= \sum_{x\in\mathcal{A}} p_X(x)\log(\frac{p_X(x)}{p_Y(x)}) \\ &= -\sum_{x\in\mathcal{A}} p_X(x)\log(\frac{p_Y(x)}{p_X(x)}) \\ &\geq -\sum_{x\in\mathcal{A}} p_X(x)(\frac{p_Y(x)}{p_X(x)} - 1) \\ &= -\sum_{x\in\mathcal{A}} (p_Y(x) - p_X(x)) \\ &= 0, \end{aligned} \tag{2.45}$$

and it is immediately seen that the minimum is at $p_Y(x) = p_X(x)$. We have now proven the following theorem:

Theorem 7 *The relative entropy of two probability mass functions is nonnegative.*

Example 2.13: The marginal density of Y minimizes $H(p_{XY}\|p_X q_Y)$

We want to find the $q_Y(y)$ that minimizes $H(p_{XY}\|p_X q_Y)$. We constrain $q_Y(y)$ to be a probability mass function (to sum to unity; we ignore the nonnegativity constraint and check that it holds at the end). The expression to be minimized can be written as

$$\begin{aligned} H(p_{XY}||p_X q_Y) &= \sum_{x,y} p_{XY}(x,y)\log(\frac{p_{XY}(x,y)}{p_X(x)q_Y(y)}) \\ &= \sum_{x,y} p_{XY}(x,y)\log(\frac{1}{q_Y(y)}) + \sum_{x,y} p_{XY}(x,y)\log(p_{Y|X}(y|x)) \\ &= \sum_{y} p_Y(y)\log(\frac{1}{q_Y(y)}) + \sum_{x,y} p_{XY}(x,y)\log(p_{Y|X}(y|x)). \end{aligned}$$

Naturally, the result of the minimization is not affected by adding and subtracting anything that does not involve $q_Y(y)$. To get a nice expression, we subtract the term $\sum_{x,y} p_{XY}(x,y)\log(p_{Y|X}(y|x))$ and add the term $\sum_y p_Y(y)\log(p_Y(y))$. We now search for the $q_Y(y)$ that minimizes

$$\begin{aligned} \sum_{y} p_Y(y)\log(\frac{1}{q_Y(y)}) + \sum_{y} p_Y(y)\log(p_Y(y)) &= \sum_{y} p_Y(y)\log(\frac{p_Y(y)}{q_Y(y)}) \\ &= H(p_Y||q_Y), \end{aligned}$$

which, under the constraint that $q_Y(y)$ is probability mass function, is minimized by $q_Y(y) = p_Y(y)$ (which implies that $q_Y(y)$ is indeed nonnegative). Thus, we obtain the result that the marginal density of Y minimizes the relative entropy $H(p_{XY}(x,y)||p_X(x)q_Y(y))$. This result will be used in the derivation of the Blahut algorithm in section 6.5.2.

2.8 Asymptotic Equipartition and Typical Sets

In section 2.2 we saw that statistical physics starts from the notion that all configurations of an ensemble (corresponding to all sequences of iid variables with a given large length) are equally likely and this naturally leads to the definition of a probability distribution. In contrast, information theory starts with the assumption of the existence of a random variable with a probability distribution. In this section, we will show that, for large sequence lengths, this leads to almost all sequences having identical probability, and we have come to a full circle.

In section 2.4, it was shown that the entropy forms a lower bound on the bit allocation required for a uniquely decodable code and that we could make that bound arbitrarily tight by coding sequences of such variables with a single codeword. In this section, we will elaborate on that statement. We start with a random variable X. We consider iid sequences of such random variables, denoted as $[X_1, \cdots, X_k]$. We divide the realizations of the random sequences into sequences $[x_1, \cdots, x_k]$ that fall in the so-called typical set and sequences $[x_1, \cdots, x_k]$ that are atypical and thus do not fall in the typical set. We will then show that the probability of observing an atypical sequence vanishes with increasing sequence length k, and that the probability of all typical sequences converges toward being the same (equipartition). If the probabilities are the same, we must encode the sequences with equal-length codewords. We will see that the typical sequences require a fixed codeword length of $kH(X) + 1$ bits. In other words, we can always find a k such that all sequences can be encoded at a rate arbitrarily close to $H(X_i)$ at an arbitrarily small probability of error.

We start with describing the laws of large numbers. Here, we will focus on the simpler results on sequences that are derived from the **weak law of large numbers** (also called **Bernoulli's theorem**). The weak law of large numbers is formulated in the following theorem:

Theorem 8 *For a sequence of independent random variables X_i, each with the distribution of the random variable X, and for a given δ and ϵ there is always an integer k_0 such that for $k \geq k_0$*

$$P(|\mathrm{E}[X] - \frac{1}{k}\sum_{i=1}^{k} X_i| < \epsilon) \geq 1 - \delta. \tag{2.46}$$

The proof of the weak law of large numbers is based on the Chebychev inequality. For notational clarity, let us consider a continuous random variable Y rather than a discrete random variable; the proof is identical for discrete random variables. We have

$$\begin{aligned}
\sigma_Y^2 &= \mathrm{E}[(Y - \mathrm{E}[Y])^2] \\
&= \int_{-\infty}^{\infty} p_Y(y)(y - \mathrm{E}[Y])^2 dy \\
&\geq \int_{-\infty}^{E[Y]-\epsilon} p_Y(y)(y - \mathrm{E}[Y])^2 dy + \int_{E[Y]+\epsilon}^{\infty} p_Y(y)(y - \mathrm{E}[Y])^2 dy \\
&\geq \epsilon^2 \left(\int_{-\infty}^{E[Y]-\epsilon} p_Y(y) dy + \int_{E[Y]+\epsilon}^{\infty} p_Y(y) dy\right),
\end{aligned} \tag{2.47}$$

where ϵ is arbitrary. This result immediately implies the Chebychev inequality:

$$P(|Y - \mathrm{E}[Y]| \geq \epsilon) \leq \frac{\sigma_Y^2}{\epsilon^2}. \tag{2.48}$$

We can then also write

$$P(|Y - \mathrm{E}[Y]| < \epsilon) \geq 1 - \frac{\sigma_Y^2}{\epsilon^2}. \tag{2.49}$$

Selecting $Y = \frac{1}{k}\sum_{i=1}^{k} X_i$, we obtain

$$P(|\mathrm{E}[X] - \frac{1}{k}\sum_{i=1}^{k} X_i| < \epsilon) \geq 1 - \frac{\sigma_X^2}{k\epsilon^2}, \tag{2.50}$$

which completes the proof.

The convergence of $\frac{1}{k}\sum_{i=1}^{k} X_i$ towards $\mathrm{E}[X]$ in the weak law of large numbers is called **convergence in probability**[5].

Let us write $X^k = [X_1, \cdots, X_k]$ and $x^k = [x_1, \cdots, x_k]$. We define a **weakly typical set**, or simply **typical set**, $\mathcal{A}_\epsilon^k$, as the set of length-k sequences of the form

$$\begin{aligned}
\mathcal{A}_\epsilon^k &\equiv \{x^k : |\frac{1}{k}\log(p_{X^k}(x^k)) + H(X)| < \epsilon\} \\
&= \{x^k : |\frac{1}{k}\sum_{i=1}^{k}\log(p_X(x_i)) + H(X)| < \epsilon\} \\
&= \{x^k : |\frac{1}{k}\sum_{i=1}^{k}\log(p_X(x_i)) - \mathrm{E}[\log p_X(X)]| < \epsilon\},
\end{aligned} \tag{2.51}$$

where we remember that X has the same distribution as X_i. From the weak law of large numbers, setting $\delta = \epsilon$, we see immediately that we can choose a sufficiently large k such that we have

$$P(X^k \in \mathcal{A}_\epsilon^k) > 1 - \epsilon. \tag{2.52}$$

In other words, for sufficiently large k almost all sequences are in the typical set.

From the definition of the typical set it immediately follows that for the sequences in the typical set

$$2^{-kH(X)-k\epsilon} < p_{X^k}(x^k) < 2^{-kH(X)+k\epsilon}. \tag{2.53}$$

We can now bound the cardinality (the number of elements) of the typical set, denoted as $|\mathcal{A}_\epsilon^k|$, as follows. An upper bound can be derived from

$$\begin{aligned}
1 - \epsilon &< \sum_{x \in \mathcal{A}_\epsilon^k} p_{X^k}(x^k) \\
&< |\mathcal{A}_\epsilon^k| 2^{-kH(X)+k\epsilon}.
\end{aligned} \tag{2.54}$$

[5] For completeness, we also list the strong law of large numbers. It can be written as $P(\mathrm{E}[X] = \lim_{k\to\infty}\frac{1}{k}\sum_{i=1}^{k} X_i) = 1$. This type of convergence is called "convergence with probability 1" or "convergence almost everywhere". It can be used to define strongly typical sets.

A lower bound can be found from

$$\begin{aligned} 1 &\geq \sum_{x \in \mathcal{A}_\epsilon^k} p_{X^k}(x^k) \\ &> |\mathcal{A}_\epsilon^k| 2^{-kH(X)-k\epsilon}. \end{aligned} \tag{2.55}$$

We conclude that the cardinality of the typical set is bounded by

$$(1-\epsilon)2^{kH(X)-k\epsilon} < |\mathcal{A}_\epsilon^k| < 2^{kH(X)+k\epsilon}. \tag{2.56}$$

By selecting ϵ such that $H(X) >> \epsilon$ we see that the cardinality of the typical set is approximately $2^{kH(X)}$ for large k.

Let us now code the sequences with a fixed codeword length. We create a table for the typical set. Since the cardinality of the typical set is less than $2^{kH(X)+k\epsilon}$, we can code all its entries with a fixed codeword length $\lceil \log_2(2^{kH(X)+k\epsilon}) \rceil = \lceil k(H(X)+\epsilon) \rceil$, where we rounded up to get an integer codeword length. When an atypical sequence appears, we can use different strategies. We can simply make an error and use a codeword for a sequence from the typical set. With this strategy, we can always find a k such that we can code any set at a rate less than $kH(X) + k\epsilon + 1$ with a probability of error δ, for any arbitrary ϵ and δ. Alternatively, we can code the entries not in the typical set, at a slight increase in rate. We add one bit to the codewords describing the typical set to indicate that this codeword describes an sequence in the typical set. Codewords not in the typical set can then be coded with $\lceil k \log_2(N) \rceil + 1$ bits, where N is the cardinality of the random variable X. Since the probability of all sequences not in the typical set decreases as $1/k$ (cf. equation 2.50), the average codeword contribution of these sequences is independent of k, which means their contribution per sample vanishes asymptotically. We have thus created an independent proof of the source coding theorem (theorem 2).

Example 2.14: The typical set for a binary distribution
Let us consider the case where $p_X(0) = 0.2$ and $p_X(1) = 0.8$, which has a (first-order) entropy $H(X) = 0.722$ bit. For a sequence, x^k, of length k with $k-n$ zeros and n ones, we define the normalized (negative) log probability,

$$G_X(n,k) \equiv -\frac{k-n}{k}\log(p_X(0)) - \frac{n}{k}\log(p_X(1)).$$

The value of $G_X(n,k)$ ranges between $\log_2(p_X(0)) = 2.32$ bit and $\log_2(p_X(1)) = 0.32$ bit. The typical set is defined as all sequences x^k for which $|-G_X(n,k) + H(X)| < \epsilon$.

To visualize the typical set, we select a particular k. We can then compute $G_X(n,k)$ for all $n \in \{0, \cdots, k\}$. For a given n and k there are $\frac{k!}{n!(k-n)!}$ sequences, each with probability $p_X(0)^{k-n}p_X(1)^n$. The probability mass function for $G_X(n,k)$ is, therefore, a binomial distribution:

$$P_G(G_X) = \frac{k!}{n!(k-n)!}p_X(0)^{k-n}p_X(1)^n.$$

In figure 2.5 we plot the smoothed and normalized probability mass function for several values of k. It is seen that almost all sequences fall in the typical set for sequences of length 1000 with $\epsilon = 0.1$ bit.

From the figure, we conclude that the typical set provides us with a theoretical understanding, but not with a practical coding procedure. To get good results

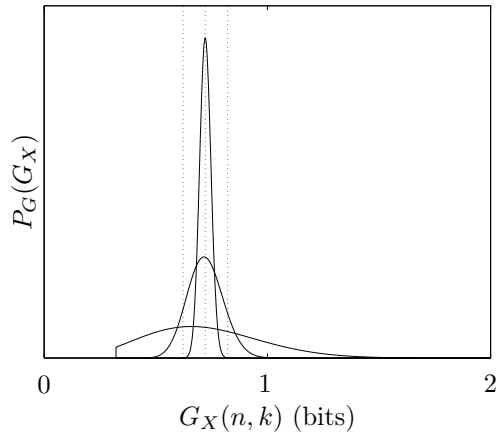

Figure 2.5: The probability mass functions of $G_X(n,k)$ of example 2.14 for $k = 10$, $k = 100$, and $k = 1000$ (increasing sharpness). The values for $H(X) - \epsilon$, $H(X)$, and $H(X) + \epsilon$ are shown as vertical dotted lines, with ϵ set to 0.1 bit.

using typical sets, we need for our example sequences that are significantly longer than 1000 source symbols, and such long sequences require a very large codebook, rendering the method computationally impractical.

Similarly to the typical set, we can also define a **jointly typical set** for elements with a joint probability density $p_{X^k,Y^k}(x^k, y^k) = \prod_{i=1}^{k} p_{X,Y}(x_i, y_i)$. The jointly typical set has as definition

$$\begin{aligned} \mathcal{A}^k_{J,\epsilon} \equiv \{(x^k, y^k) \quad : \quad & |\frac{1}{k}\log p_{X^k}(x^k) + H(X)| < \epsilon, \\ & |\frac{1}{k}\log p_{Y^k}(y^k) + H(Y)| < \epsilon, \\ & |\frac{1}{k}\log p_{X^kY^k}(x^k, y^k) + H(X,Y)| < \epsilon\}, \end{aligned} \tag{2.57}$$

where the subscript j in $\mathcal{A}^k_{J,\epsilon}$ denotes "joint". We note from equation 2.46 that it follows from the weak law of large numbers that there must be a k that is sufficiently large such that simultaneously

$$P(|\frac{1}{k}\log p_{X^k}(X^k) + H(X)| > \epsilon) \quad < \quad \epsilon/3, \tag{2.58}$$

$$P(|\frac{1}{k}\log p_{Y^k}(Y^k) + H(Y)| > \epsilon) \quad < \quad \epsilon/3, \tag{2.59}$$

$$P(|\frac{1}{k}\log p_{X^kY^k}(X^k, Y^k) + H(X,Y)| > \epsilon) \quad < \quad \epsilon/3, \tag{2.60}$$

where we picked $\epsilon/3$ to make the next expression look nicer. It follows that

$$P([X^k, Y^k] \in A^k_{J,\epsilon}) \geq 1 - \epsilon. \tag{2.61}$$

This means that there is a k for which the probability that a sequence pair $[X^k, Y^k]$, drawn from the density $f_{X^k,Y^k}(x^k, y^k) = \prod_{i=1}^{k} f_{X,Y}(x_i, y_i)$ is in the jointly typical set, is $1 - \epsilon$, where ϵ can be chosen arbitrarily close to zero.

From the definitions, it is easy to provide bounds on the probabilities for the sequences in the jointly typical set:

$$2^{-kH(X)-k\epsilon} < p_{X^k}(x^k) < 2^{-kH(X)+k\epsilon}, \quad (2.62)$$

$$2^{-kH(Y)-k\epsilon} < p_{Y^k}(y^k) < 2^{-kH(Y)+k\epsilon}, \quad (2.63)$$

$$2^{-kH(X,Y)-k\epsilon} < p_{X^kY^k}(x^k, y^k) < 2^{-kH(X,Y)+k\epsilon}. \quad (2.64)$$

An interesting result on the conditional probability density $p_{X^k|Y^k}(x^k|y^k)$ follows from the relations 2.62 through 2.64. We first recall that

$$p_{X^k|Y^k}(x^k|y^k) = p_{X^k}(x^k)\frac{p_{X^kY^k}(x^k, y^k)}{p_{Y^k}(y^k)p_{X^k}(x^k)}. \quad (2.65)$$

We furthermore recall that $I(X;Y) = H(X) + H(Y) - H(X,Y)$. Then, combining equation 2.65 with the inequalities 2.62 through 2.64 results in:

$$p_{X^k}(x^k)\, 2^{kI(X;Y)-3k\epsilon} \leq p_{X^k|Y^k}(x^k|y^k) \leq p_{X^k}(x^k)\, 2^{kI(X;Y)+3k\epsilon}. \quad (2.66)$$

This result will be used in the proof of the rate-distortion theorem in section 6.2.6.

2.9 Problems

1. Consider a random variable X that can take the values 0 and 1 with probability mass function $p_X(0) = p$ and $p_X(1) = 1 - p$. Plot $H(X)$ as a function of p.
2. Consider an alphabet $\mathcal{A} = \{a, b, c, d\}$. In this problem we encode sequences of variables from this alphabet with either a binary or a ternary code, using a separate codeword for each variable.
   (a) Outline the complete proof that we can obtain $L < H(X) + 1$ for the ternary code, where L is the average code length per (input) symbol and where $H(X)$ is expressed in ternary units.
   (b) Define a reasonable conversion factor for codeword length from bits to ternary units so that you can compare them.
   (c) Find a probability distribution for $\mathcal{A}$ where the ternary code is more efficient than the binary code.
   (d) Find a probability distribution for $\mathcal{A}$ where the binary code is more efficient than the ternary code.
   (e) Argue that, in general, a binary code is more efficient than a ternary code.
3. Prove that $H(X + bY|Y) = H(X|Y)$, where b is a constant (X and Y are not independent).
4. Prove that $\log(x) \leq x - 1$ for $x \in [0, \infty)$.
5. Show, using probability mass functions, that $H(X) \geq H(X|Y)$.
6. Consider a symbol set with probabilities $0.05, 0.05, 0.1, 0.8$.
   (a) Construct a Shannon prefix code for this symbol set.

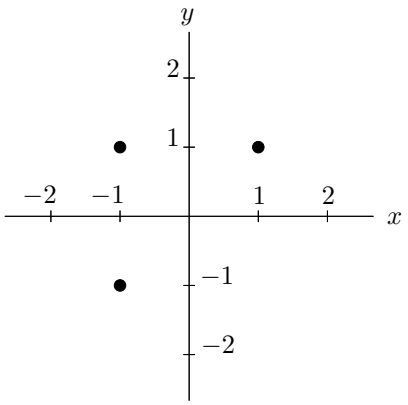

Figure 2.6: Probability mass function for problem 10. The three points each have probability 1/3.

(b) Evaluate the advantage of this prefix code over using equal codeword lengths for all symbols.

7. Consider two random variables, $X \in \{a, b\}$ and $Y \in \{c, d\}$. Let
$p_{XY}(a, c) = 0.25$,
$p_{XY}(a, d) = 0.25$,
$p_{XY}(b, c) = 0.50$,
$p_{XY}(b, d) = 0.0$.

   (a) Compute $H(X)$ and $H(Y)$.

   (b) Compute $H(X, Y)$, $H(X|Y)$, and $H(Y|X)$.

   (c) Compute $I(X; Y)$.

   (d) You have to transmit pairs of X and Y. Construct a Shannon prefix code, for transmitting the pair X, Y.

8. Express the conditional mutual information in terms of probability mass functions.

9. Construct a binary prefix code with the following codeword lengths: 2,2,3,3,4,4,4.

10. Consider figure 2.6.

   (a) Compute $H(X)$, $H(Y)$, and $H(X, Y)$.

   (b) Compute the mutual information between X and Y.

11. In figure 2.7, all four points in the X, Y plane have equal probability 1/4.

   (a) Compute $H(X)$, $H(Y)$, $H(X, Y)$, $H(X|Y)$, $H(Y|X)$, and $I(X; Y)$.

   (b) Given that the sample space consists of four separate discrete points in the x, y plane, each with probability 1/4, describe configurations of these points that result in maximum and minimum mutual information. Compute the numerical values for the mutual information in both cases.

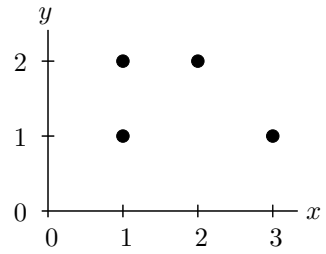

Figure 2.7: Probability mass function for problem 11.

12. You were to compute the entropy rate in a sequence. You approximated the probability mass densities and found the order-k entropies $H_1(X_i)$, $H_2(X_i)$, and $H_3(X_i)$ for the sequence. Unfortunately, you don't remember which is which. The numerical values are 4.0, 3.0, and 3.5 bits.

    (a) Provide a qualitative reasoning for the ordering of the order-k entropies and use this to assign the numbers to the correct order-k entropy.

    (b) Express the conditional entropies $H(X_i|X_{i-1})$ and $H(X_i|X_{i-1}, X_{i-2})$ in terms of the known order-k entropies.

    (c) Each of the conditional entropies and order-k entropies discussed above forms an upper bound on the entropy rates. Which of them provides the lowest upper bound for the entropy rate?

13. Provide an example of a random variable for which $L = H(X)$ and an example of a random variable for which the optimal code satisfies $L > H(X) + 1 - \epsilon$, where ϵ is an arbitrarily small positive number.

14. Compute $H(X_i)$, $H_2(X_i)$, $H_\infty(X_i)$, $I(X_i; X_{i-1})$, and $I(X_i; X_{i-2})$ for a stationary Markov process with two states and state-transition matrix,

$$T = \begin{bmatrix} 0.75 & 0.5 \\ 0.25 & 0.5 \end{bmatrix},$$

    where $T_{ij} = P(s(n) = i|s(n-1) = j)$, $s(n)$ being the state at time n.

15. Describe a coding procedure that, by selecting a sufficiently long sequence length, codes iid sequences at a rate arbitrarily close to the entropy rate of the sequence. The code should use codewords of only two lengths: one for the typical set and one for the remaining sequences.

16. Consider the normalized probability of example 2.14. In the following problems exploit the Stirling approximation $\log(n!) = n\log(n) - n$.

    (a) Prove $P_G(G_X(x^k))$, has an asymptotic maximum at $H(X)$.

    (b) Derive an equation for the variance $(G_X(X^k) - H(X))^2$.

    (c) For the numbers in the example, find the sequence length for which a fixed-length codeword can operate within 0.01 bit of the entropy, with small error probability.

    (d) Find the probability of error when operating within 0.01 bit of the entropy.

17. Consider a random variable with alphabet $\{0, 1, 2\}$ and $p_X(0) = 0.2$, $p_X(0) = 0.3$, $p_X(0) = 0.4$.

    (a) For a sequence of length k, what is the probability of the most probable sequence and of the most improbable sequence? Do these sequences usually fall in the typical set?

    (b) Plot the probability density for $\frac{1}{k}\log(p_{X^k}(x^k))$ for $k = 10, 100, 1000$.

    (c) To have a typical set with $\epsilon = 0.01$ (in equation 2.51), how large should k be?

18. Consider an iid sequence with two source symbols, 0 and 1.

    (a) Show that for the case $p_X(0) = p_X(1) = 0.5$ all sequences are in the typical set.

    (b) For the case $p_X(0) = 0.4$ and $p_X(1) = 0.6$, compute the entropy. Estimate (by approximation or computer) the fraction of sequences not in the typical set, using $\epsilon = 0.1$ bit, for a sequence length of 100, 1000, and 10000.

19. Conditioning affects mutual information differently from entropy. Provide an example where the mutual information increases and an example where the mutual information decreases as a result of conditioning.

20. The objective of a television game is for panel members to determine a person's profession. The person can only answer yes or no and the panel is assumed to ask "optimal" questions. If it takes on average 15 questions to determine the person's profession, what can you say about the probability distribution of the professions?

21. For many codes satisfying the Kraft inequality, the inequality is strict.

    (a) Prove that for $\{0, 10, 1100, 1101, 1110\}$ the inequality is strict.

    (b) Provide an example of an undecodable sequence for this code.

    (c) Relate the strict inequality to the fact that there exist sequences of code symbols that cannot be decoded.

22. Consider a discrete variable X with alphabet $\{-1, 0, 1\}$.

    (a) Write down all constraints that are imposed on the probability-mass function that satisfies $\mathrm{E}[X^2] = \frac{1}{2}$.

    (b) Find the maximum-entropy probability-mass function if $\mathrm{E}[X^2] = \frac{1}{2}$.

    (c) Find the *minimum*-entropy probability-mass function if $\mathrm{E}[X^2] = \frac{1}{2}$.

    (d) Find the maximum *and* minimum entropy of the constrained variable X.

    (e) Find the maximum *and* minimum entropy of the variable if the constraint is removed.

23. Consider a two-state Markov process X_i with states A and B state-transition matrix,

$$T = \begin{bmatrix} 1-\alpha & \beta \\ \alpha & 1-\beta \end{bmatrix},$$

    where $T_{nm} = P(x_i = n | x_{i-1} = m)$, x_i being the state at time i.

(a) Find the probabilities $P(x_i = A)$ and $P(x_i = B)$ for the stationary solution in terms of α and β. Hint: realize that the net flow to each state should balance (alternatively solve the eigenvalue problem).

(b) Express the entropy rate of the two-state Markov process in terms of α and β.

(c) Find the values for α and β that *maximize* the entropy rate. Then find the values for α and β that *minimize* the entropy rate. Give both entropy rates.

(d) Assume you have a sequence of observed states $[x_0, x_1, \cdots, x_{k-1}]$. Explain a sound procedure to find estimates of α and β.

24. We define $I(X;Y;W) = H(X)+H(Y)+H(W)-H(X,Y)-H(X,W)-H(Y,W)+H(X,Y,W)$.

(a) Show that $I(X;Y;W) = I(X;Y) - I(X;Y|W)$.

(b) Show that, in contrast to $I(X;Y)$, $I(X;Y;W)$ can be negative.

3

Continuous-Alphabet Variables

3.1 Introduction

In chapter 2, information theory was discussed in the context of discrete alphabets. This provided us with bounds on the mean rate required to encode a sequence of discrete alphabet variables. In practice, we often want to encode signals that are analog in nature. Again, we would like to determine bounds for the rates at which such signals must be encoded. It is, however, obvious that, in general, the exact description of a random variable that can take any value within a continuous alphabet is not possible with a finite number of bits.

We could take a practical approach to measuring information in an analog random variable by rounding the realizations of the random variable to a set of uniformly spaced values. In other words, we quantize the signal with a uniform scalar quantizer prior to measuring its information content. This type of quantization operation is what is done in an analog-to-digital converter. We can now compute the entropy of the discrete-alphabet variable, which, unfortunately, depends on the quantizer step size. It seems that we gained practicality at the expense of generality. However, there is a saving grace to this approach: it turns out that, in the limit of high resolution, the dependency of the entropy on the quantization step size resides in an additive term. Thus, by simply subtracting this term we obtain an intrinsic information measure of the random variable that has practical value and is independent of the quantization step size. This measure is called the **differential entropy**. We will return to this argument at the end of section 3.2.

In the above description, differential entropy was described as a measure of information for a random variable with a continuous alphabet. This information measure has obvious practical significance if we use a high-resolution uniform scalar quantizer as first encoding step. However, its use is more general. A quantizer inherently leads to distortion, and this leads to the more general problem of finding the best trade-off between distortion and bit rate. This question is addressed by rate-distortion theory and high-

rate theory, which are discussed in later chapters. We will find that differential entropy plays a central role in the determination of bounds on the best trade-off between distortion and rate. We will also find that, in the high-resolution limit, uniform quantizers are often optimal. The discussion of differential entropy in this chapter forms, therefore, a prerequisite for chapters 6 and 7.

3.2 Differential Entropy

Entropy is not defined for variables with continuous alphabets. By simply replacing the summation in the definition of entropy with an integral, we define a new measure, differential entropy, for a random variable X with probability density $f_X(x)$:

$$\begin{aligned} h(X) &= -\int f_X(x)\log(f_X(x))dx \\ &= -\mathrm{E}[\log(f_X(x)], \end{aligned} \tag{3.1}$$

where we assume that $f_X(x)\log(f_X(x))$ is integrable. While the definition of differential entropy is motivated by that of entropy, its properties differ significantly. For example, while entropy is guaranteed to be nonnegative, it is easy to construct a random variable with a negative differential entropy, as will be illustrated in examples 3.1 and 3.2.

Although differential entropy has properties different from those of entropy, it is common to omit the specifier "differential" when there is no confusion and the precise description is cumbersome. We will follow this convention and omit the specifier "differential" occasionally.

Example 3.1: Differential entropy for variable with rectangular density
Consider a density

$$f_X(x) = \begin{cases} 1/a, & x \in [0,a), \\ 0 & \text{elsewhere.} \end{cases}$$

Its differential entropy is

$$h(X) = -\int_0^a \frac{1}{a}\log(\frac{1}{a})dx = \log(a). \tag{3.2}$$

The differential entropy increases with a, which is consistent with the uncertainty about X increasing with the size of the interval $[0,a)$. However, somewhat less intuitive is that $h(X)$ is negative for $a < 1$.

Example 3.2: Random variable with Laplace density
In this example we compute the differential entropy of a random variable with Laplace density, i.e., with $f_X(x) = \frac{a}{2}e^{-a|x|}$. The differential entropy is

$$\begin{aligned} h(X) &= -\int_{-\infty}^{\infty} f_X(x)\log(\frac{a}{2}e^{-a|x|})dx \\ &= -\log(\frac{a}{2}) + \frac{1}{2}\int_{-\infty}^{\infty} a^2|x|e^{-a|x|}dx \\ &= \log(\frac{2}{a}) + a^2\int_0^{\infty} xe^{-ax}dx \\ &= \log(\frac{2}{a}) + 1 \\ &= \log(\frac{2\mathrm{e}}{a}). \end{aligned}$$

It is seen that the differential entropy of the random variable decreases with increasing a. This is consistent with the decreased uncertainty about the value the random variable will take when a is larger. We observe that the differential entropy becomes negative for large values of a.

In both examples 3.1 and 3.2, a scaling by a positive factor α changes the differential entropy by $\log_2(\alpha)$ bits. We now show that this is a general result. Let us consider a density $f_X(x)$ and let us note that $f_{\alpha X}(\alpha x) = \frac{1}{\alpha} f_X(x)$. Then we have that

$$\begin{aligned} h(\alpha X) &= -\int f_{\alpha X}(y) \log(f_{\alpha X}(y)) dy \\ &= -\alpha \int f_{\alpha X}(\alpha x) \log(f_{\alpha X}(\alpha x)) dx \\ &= -\alpha \int \frac{1}{\alpha} f_X(x) \log(\frac{1}{\alpha} f_X(x)) dx \\ &= h(X) + \log(\alpha). \end{aligned} \tag{3.3}$$

The effect of scaling on differential entropy is included in the more general theorem 9, which will be discussed in section 3.3.

While scaling of a continuous-alphabet random variable affects its differential entropy, translation does not. This is easily verified:

$$\begin{aligned} h(X+b) &= -\int f_{X+b}(y) \log(f_{X+b}(y)) dy \\ &= -\int f_{X+b}(x+b) \log(f_{X+b}(x+b)) d(x+b) \\ &= -\int f_X(x) \log(f_X(x)) dx \\ &= h(X). \end{aligned} \tag{3.4}$$

So-far in this section, we have defined differential entropy, discussed some of its properties, and computed its value for some examples. However, we have not shown directly that the measure is relevant for encoding a signal. Thus, we now return to the motivation for differential entropy presented in the introduction of this chapter. We consider a continuous random variable, X, with a density $f_X(x)$, which we assume to be Riemann integrable[1]. We define a set, $\mathcal{A}$, of numbers on the real line spaced by Δ and also define the probability mass function $p_\Xi(\xi) \equiv \int_\xi^{\xi+\Delta} f_X(x) dx, \xi \in \mathcal{A}$. We quantize the random variable X to obtain a discrete random variable, Ξ, with the discrete-alphabet $\mathcal{A}$. We assume that Δ is sufficiently small that the relation between the density and the probability-mass function can be approximated by $f_X(\xi)\Delta \approx p_\Xi(\xi)$ for $\xi \in \mathcal{A}$. The entropy of the discrete variable can then be written as

$$\begin{aligned} H(\Xi) &= -\sum_{\mathcal{A}} p_\Xi(\xi) \log(p_\Xi(\xi)) \\ &\approx -\sum_{\mathcal{A}} p_\Xi(\xi) \log(f_X(\xi)\Delta) \\ &\approx -\sum_{\mathcal{A}} f_X(\xi) \log(f_X(\xi))\Delta - \log(\Delta). \end{aligned} \tag{3.5}$$

[1] A function is Riemann integrable if it is continuous almost everywhere.

In the limit of vanishing Δ, the term $\sum_{\mathcal{A}} f_X(\xi)\log(f_X(\xi))\Delta$ defines a Riemann integral. We thus obtain for small Δ:

$$\begin{aligned} H(\Xi) &\approx -\int f_X(x)\log(f_X(x))dx - \log(\Delta) \\ &= h(X) - \log(\Delta). \end{aligned} \tag{3.6}$$

It follows from equation 3.6 that the differential entropy describes the part of the entropy of the discrete alphabet that is independent of the quantization step size. It is also seen that, for small Δ, the entropy approaches infinity while the differential entropy remains finite.

For the approximate equality 3.6 to hold, we only need to have that the function $f_X(x)$ is smooth with respect to the step size Δ. It is sometimes convenient to select $\Delta = 1$; then the entropy of the discrete-amplitude signal equals the differential entropy of the continuous-amplitude signal. This situation is often representative of the output of A/D converters. In example 3.3, we discuss under what conditions the differential entropy (approximately) equals the entropy for the case of the Laplace density.

Example 3.3: Uniform quantization, entropy, and differential entropy
We consider a random variable with the Laplace density of example 3.2. We want to know for what range of the Laplace-density parameter a we can use the unity step-size quantizer for which the entropy and differential entropy are approximately equal. We will use rather crude methods in our estimates.

Let $f_X(\xi)^*$ and $f_X(\xi)_*$ denote the supremum and infimum values of $f_X(x)$ for $x \in [\xi, \xi+\Delta]$. Then we have that

$$f_X(\xi)_* \log(f_X(\xi)_*) \leq f_X(\xi)\log(f_X(\xi)) \leq f_X(\xi)^* \log(f_X(\xi)^*)$$

and, similarly, for all $x \in [\xi, \xi+\Delta]$

$$f_X(\xi)_* \log(f_X(\xi)_*) \leq f_X(x)\log(f_X(x)) \leq f_X(\xi)^* \log(f_X(\xi)^*).$$

This implies the following bound:

$$|f_X(x)\log(f_X(x)) - f_X(\xi)\log(f_X(\xi))| \leq f_X(\xi)^*\log(f_X(\xi)^*) - f_X(\xi)_*\log(f_X(\xi)_*).$$

We want a quantizer for which the approximate equality 3.6 is valid with a fractional error of less than η. To be below this error, we require that for each interval

$$\frac{f_X(\xi)^*\log(f_X(\xi)^*) - f_X(\xi)_*\log(f_X(\xi)_*)}{f_X(\xi)\log(f_X(\xi))} \leq \eta.$$

We assume that the logarithmic factors vary slowly and can be omitted. For the Laplace density and $x \geq 0$ (similar results are obtained for negative x) we obtain then

$$\frac{e^{-a\xi} - e^{-a(\xi+\Delta)}}{e^{-a\xi}} = 1 - e^{-a\Delta} \leq \eta.$$

This implies that

$$\begin{aligned} e^{-a\Delta} &\geq 1-\eta \\ -a\Delta &\geq \log(1-\eta) \\ \Delta &\leq -\frac{1}{a}\log(1-\eta). \end{aligned}$$

From the inequality $\log(1+\eta) \leq \eta$ for small η, we obtain finally

$$\Delta \leq \frac{\eta}{a}.$$

If we select $\Delta = 1$, then the entropy equals the differential entropy. For the Laplace density, this is the case when $\frac{\eta}{a} \geq 1$, i.e., when $a \leq \eta$ is sufficiently small. In particular, if we assume an accuracy of 1/1000 is sufficient then we set $\eta = 1/1000$ and we must have $a \leq 1/1000$. Such values can always be achieved by simply scaling the signal appropriately.

3.3 More Information Measures

Measures corresponding to those derived from entropy in chapter 2 can also be defined for continuous variables. **Joint differential entropy, higher-order differential entropy, differential entropy rate, redundancy rate, conditional differential entropy, mutual information**, and **relative entropy** (as we will see later, there is no reason to add "differential" to redundancy rate, mutual information, and relative entropy) are all defined similarly as for the discrete alphabet case:

$$\begin{aligned}
h(X_1, \cdots, X_k) &\equiv -\mathrm{E}[\log(f_{X_1 \cdots X_k}(X_1, \cdots, X_k))], && (3.7)\\
h_k(X_i) &\equiv \frac{1}{k} h(X_1, \cdots, X_k), && (3.8)\\
h_\infty(X_i) &\equiv \lim_{k\to\infty} \frac{1}{k} h(X_1, \cdots, X_k), && (3.9)\\
\rho(X_i) &\equiv h_1(X_i) - h_\infty(X_i), && (3.10)\\
h(Y|X) &\equiv h(X,Y) - h(X), && (3.11)\\
I(X;Y) &\equiv h(X) + h(Y) - h(X,Y), && (3.12)\\
h(f_X || f_Y) &\equiv \int f_X(x) \log(\frac{f_X(x)}{f_Y(x)}) dx. && (3.13)
\end{aligned}$$

It is useful to relate the various measures for the discrete variables to those for the continuous variables. The joint entropy, the conditional entropy, and the entropy rate have similar relations to the corresponding continuous measures as the relation between entropy and differential entropy given in equation 3.6 (the relations will be worked out in problem 1). However, the situation is different for redundancy rate, relative entropy and mutual information.

The case of redundancy rate is very straightforward. As is seen in equation 3.10, redundancy rate is defined as a difference between differential entropies. Thus the term $-\log(\Delta)$ in equation 3.6 cancels, and the redundancy rate of a stationary discrete-alphabet discrete process converges to that of the corresponding continuous alphabet case with decreasing step size.

Next, we consider the relation between the discrete-case and the continuous-case for relative entropy and mutual information. Since mutual information can be written as a relative entropy we discuss only the latter in detail. We consider two densities, $f_X(x)$ and $f_Y(x)$, both Riemann integrable. We define a set, $\mathcal{A}$, of numbers on the real line spaced by Δ and also define the probability mass functions $p_\Xi(\xi) \equiv \int_\xi^{\xi+\Delta} f_X(x)dx, \xi \in \mathcal{A}$ and

$p_\Upsilon(\xi) \equiv \int_\xi^{\xi+\Delta} f_Y(x)dx, \xi \in \mathcal{A}$. We quantize the random variables X and Y to obtain a discrete random variable, Ξ and Υ, both with a discrete-alphabet $\mathcal{A}$. We assume that Δ is sufficiently small that the relations $f_X(\xi)\Delta \approx p_X(\xi)$ with $\xi \in \mathcal{A}$ and $f_Y(\xi)\Delta \approx p_Y(\xi)$ with $\xi \in \mathcal{A}$ are sufficiently accurate. We can now write for the relative entropy

$$\begin{aligned} H(P_\Xi||P_\Upsilon) &= \sum_{\xi\in\mathcal{A}} p_\Xi(\xi)\log(\frac{p_\Xi(\xi)}{p_\Upsilon(\xi)}) \\ &\approx \sum_{\xi\in\mathcal{A}} f_X(\xi)\Delta\log(\frac{f_X(\xi)}{f_Y(\xi)}) \\ &\approx \int f_X(x)\log(\frac{f_X(x)}{f_Y(x)})dx \\ &= h(f_X||f_Y). \end{aligned} \tag{3.14}$$

Thus, with decreasing Δ the relative entropy of the variables Ξ and Υ converges to the corresponding relative differential entropy of the variables X and Y.

It is useful to generalize the result on the effect of scaling (equation 3.3) to the vector case. Let us consider the linear transformation

$$Y^k = AX^k, \tag{3.15}$$

where A is a matrix. From elementary probability theory we have that

$$f_{Y^k}(y^k) = \frac{1}{|\det(A)|} f_{X^k}(x^k)$$

and that

$$dy^k = |\det(A)|dx^k.$$

This implies that

$$\begin{aligned} h(Y^k) &= -\int f_{Y^k}(y^k)\log(f_{Y^k}(y^k))dy^k \\ &= -\int f_{X^k}(x^k)\log(\frac{f_{X^k}(x^k)}{|\det(A)|})dx^k \\ &= h(X^k) + \log(|\det(A)|), \end{aligned} \tag{3.16}$$

which we capture in a theorem:

Theorem 9 *Consider a matrix multiplication $Y^k = AX^k$. The differential entropies of Y^k and X^k are related by $h(Y^k) = h(X^k) + \log(|\det(A)|)$.*

Example 3.4: Relative entropy of two-level and uniform distribution
We will confirm that the relative entropy between distributions with two probability levels and with one level is the same for the (high-resolution) discrete and the continuous cases for a particular example. We discuss the discrete case first. We assume the random variables Ξ and Υ have an alphabet $\mathcal{A} = \{1, \cdots, |\mathcal{A}|\}$ with even cardinality $|A|$. We consider the probability mass functions

$$\begin{aligned} p_\Xi(\xi) &= 1/|\mathcal{A}|, \ \xi \in \mathcal{A} \\ p_\Upsilon(\xi) &= \begin{cases} (1+\alpha)/|\mathcal{A}|, & \xi \in \{1, \cdots, |\mathcal{A}|/2\} \\ (1-\alpha)/|\mathcal{A}|, & \xi \in \{|\mathcal{A}|/2+1, \cdots, |\mathcal{A}|\} \end{cases} \end{aligned}$$

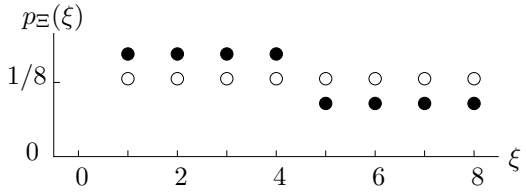

Figure 3.1: Probability mass functions $P_\Xi(\xi)$ and $P_\Upsilon(\xi)$ of example 3.4 for the case $|\mathcal{A}| = 8$.

where $\alpha \in [0, 1]$.

The relative entropy $H(\Xi||\Upsilon)$ is

$$\begin{aligned} H(P_\Xi||P_\Upsilon) &= \sum_\xi p_\Xi(\xi) \log(\frac{p_\Xi(\xi)}{p_\Upsilon(\xi)}) \\ &= \sum_{\xi=1}^{|\mathcal{A}|/2} \frac{1}{|\mathcal{A}|} \log(\frac{1/|\mathcal{A}|}{(1+\alpha)/|\mathcal{A}|}) + \sum_{\xi=|\mathcal{A}|/2+1}^{|\mathcal{A}|} \frac{1}{|\mathcal{A}|} \log(\frac{1/|\mathcal{A}|}{(1-\alpha)/|\mathcal{A}|}) \\ &= -\frac{1}{2}\log(1+\alpha) - \frac{1}{2}\log(1-\alpha) \\ &= -\frac{1}{2}\log(1-\alpha^2) \end{aligned}$$

The relative entropy vanishes when $\alpha = 0$, i.e., when the probability mass functions are equal.

In this particular case, the relative entropy is not dependent on the cardinality of the alphabet. This already suggests that we must get the same answer for the continuous case. To confirm this, we first define uniform and stepped density functions on $[0, a)$:

$$\begin{aligned} f_X(x) &= 1/a, \;\; x \in [0, a) \\ f_Y(x) &= \begin{cases} (1+\alpha)/a, & x \in [0, a/2) \\ (1-\alpha)/a, & x \in [a/2, a) \end{cases} \end{aligned}$$

The relative entropy $h(f_X||f_Y)$ is easily computed by splitting the integral into

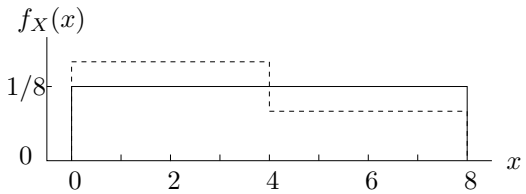

Figure 3.2: Probability density functions $f_X(x)$ and $f_Y(x)$ of example 3.4 for the case $a = 8$.

two segments:

$$\begin{aligned} h(f_X||f_Y) &= \int f_X(x)\log(\frac{f_X(x)}{f_Y(x)})dx \\ &= \int_0^{a/2}\frac{1}{a}\log(\frac{1/a}{(1+\alpha)/a})dx + \int_{a/2}^{a}\frac{1}{a}\log(\frac{1/a}{(1-\alpha)/a})dx \\ &= -\frac{1}{2}\log(1+\alpha) - \frac{1}{2}\log(1-\alpha) \\ &= -\frac{1}{2}\log(1-\alpha^2) \end{aligned}$$

which, as expected, equals the result for the discrete case.

Example 3.5: Mutual information for a simple joint density
We compute the mutual information between the random variables X and Y whose density is shown in figure 3.3. The bar shape has length $\sqrt{2}$ and width b. We assume that b is small, so that we can avoid having to deal carefully with the end regions. In its region of support, the density $f_{XY}(x,y) = \frac{1}{\sqrt{2}b}$. Similarly, in their regions of support, the marginal densities are $f_Y(y) = f_X(x) = 1$. The joint entropy is

$$\begin{aligned} h(X,Y) &= -\int\int f_{XY}(x,y)\log f_{XY}(x,y)dxdy \\ &= \log(\sqrt{2}b) \\ &= \frac{1}{2}\log(2b^2). \end{aligned}$$

The marginal differential entropy $h(X)$ is

$$\begin{aligned} h(X) &= -\int_{-\frac{1}{2}}^{\frac{1}{2}} f_X(x)\log(f_X(x))dx \\ &= -\int_{-\frac{1}{2}}^{\frac{1}{2}} 1\log(1)dx \\ &= 0 \end{aligned}$$

and furthermore $h(Y) = h(X)$. The mutual information is now

$$\begin{aligned} I(X;Y) &= h(X) + h(Y) - h(X,Y) \\ &= -\frac{1}{2}\log(2b^2). \end{aligned}$$

The example illustrates several issues. First, since $b << 1$, the differential entropy $h(X,Y)$ is negative, but the mutual information is positive (as it should be). Second, the mutual information between X and Y is finite whenever b is finite, i.e., whenever we have some remaining uncertainty about Y if we know X (and vice versa). The mutual information increases when this uncertainty decreases. This is natural: with decreasing b more information is shared between the variables. In the limit of vanishing b the mutual information becomes infinite since then $X = Y$ and the entropy of the continuous variables X and Y is infinite.

Since the relative differential entropy and relative entropy are closely related, we expect that the nonnegativity property carries over. This is easily confirmed, using the same

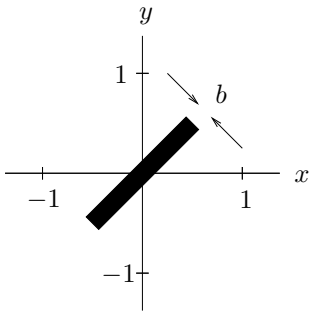

Figure 3.3: Joint density function for example 3.5. The bar has a length of $\sqrt{2}$.

procedure as that used in equation 2.45 for the discrete case:

$$\begin{aligned} h(f_X||f_Y) &= \int f_X(x)\log(\frac{f_X(x)}{f_Y(x)})dx \\ &= -\int f_X(x)\log(\frac{f_Y(x)}{f_X(x)})dx \\ &\geq -\int f_X(x)(\frac{f_Y(x)}{f_X(x)}-1)dx \\ &= -\int (f_Y(x)-f_X(x))\,dx \\ &= 0. \end{aligned} \tag{3.17}$$

We write this result as a theorem:

Theorem 10 *The relative (differential) entropy of two probability densities is nonnegative.*

A corollary of this theorem is that the nonnegativity of mutual information (theorem 6) is also valid for continuous variables.

It is also of practical interest to consider the effect of scaling on mutual information. Since mutual information of two variables cannot be negative, we know that scaling cannot affect mutual information like it does differential entropy. Let X and Y be continuous-alphabet random variables and α a scaling factor. Then we have

$$\begin{aligned} I(X;\alpha Y) &= h(\alpha Y) - h(\alpha Y|X) \\ &= h(Y) + \log(\alpha) - (h(Y|X) + \log(\alpha)) \\ &= h(Y) - h(Y|X) = I(X;Y). \end{aligned} \tag{3.18}$$

Since mutual information can be written as a difference of the unconditioned and conditioned differential entropy of a variable, it is not affected by scaling.

Many practical problems have continuous random variables that themselves are sums of two continuous variables. The following simple theorem is useful in this type of situation:

Theorem 11 *The differential entropy of the variable $X + \alpha Y$ conditioned on Y equals the differential entropy of X conditioned on Y:*

$$h(X + \alpha Y|Y) = h(X|Y). \tag{3.19}$$

The discrete case was treated in problem 3 of chapter 2. The theorem is easily proven:

$$\begin{aligned} h(X + \alpha Y|Y) &= \int f_Y(y) h(X + \alpha Y|Y = y) dy \\ &= \int f_Y(y) h(X|Y = y) dy \\ &= h(X|Y). \end{aligned} \tag{3.20}$$

If X and Y are independent continuous-alphabet random variables, then the theorem reduces to $h(X + \alpha Y|Y) = h(X)$.

3.4 Gaussian Densities

Processes and variables with a Gaussian (also called **normal**) distribution play an important role in source coding. Many processes have a distribution that is nearly Gaussian, a fact that can be explained from the central-limit theorem[2]. More-over, in source-coding theory, the Gaussian distribution often leads to convenient upper bounds on achievable rates. For example, as we will see below, for a given variance the Gaussian distribution is the continuous distribution that leads to the highest differential entropy. Obviously, the combination of practical applicability and ease of analysis makes studying the properties of Gaussian densities very worthwhile.

The normal, or Gaussian, density for a random variable X is

$$f_X(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \mathrm{e}^{-\frac{(x-\mu)^2}{2\sigma^2}}, \tag{3.21}$$

where μ and σ^2 are the mean and variance of X. The differential entropy of a Gaussian variable is easy to compute if we use nats as unit (i.e., we use the natural logarithm):

$$\begin{aligned} h(X) &= -\int f_X(x) \log(f_X(x)) dx \\ &= \int f_X(x) \log(\sqrt{2\pi\sigma^2}) dx + \int f_X(x) \frac{x^2}{2\sigma^2} dx \\ &= \frac{1}{2} \log(2\pi\sigma^2) + \frac{1}{2} \\ &= \frac{1}{2} \log(2\pi \mathrm{e} \sigma^2). \end{aligned} \tag{3.22}$$

Example 3.6: A simple finite-impulse-response filter
We want to compute the first-order entropy, the entropy rate, and the mutual information between samples for the process

$$X_i = E_i + E_{i-1},$$

[2]The central-limit theorem states that, under certain common conditions, the density of the summation of independent random variables with arbitrary distributions tends to a normal distribution as the number of variables approaches infinity (e.g., [9]).

where the process E_i is iid normal with unit variance. We first determine the first-order differential entropy of X_i. A standard result of probability theory is that the summation of two Gaussian variables is a Gaussian variable with a variance that is the sum of the variances of the two original variables. Indeed, the marginal density of X_i is:

$$\begin{aligned}
f_{X_i}(x_i) &= \int f_{X_i|E_{i-1}}(x_i|e_{i-1})\, f_{E_{i-1}}(e_{i-1})\, de_{i-1} \\
&= \int f_{E_i|E_{i-1}}(x_i - e_{i-1}|e_{i-1})\, f_{E_{i-1}}(e_{i-1})\, de_{i-1} \\
&= \int f_{E_i}(x_i - e_{i-1})\, f_{E_{i-1}}(e_{i-1})\, de_{i-1} \\
&= \frac{1}{2\pi} \int \mathrm{e}^{-\frac{(x_i - e_{i-1})^2}{2}}\, \mathrm{e}^{-\frac{e_{i-1}^2}{2}}\, de_{i-1} \\
&= \frac{1}{2\pi} \int \mathrm{e}^{-\frac{(x_i^2 - 2x_i e_{i-1} + 2e_{i-1}^2)}{2}}\, de_{i-1} \\
&= \frac{1}{2\pi}\, \mathrm{e}^{-\frac{x_i^2}{4}} \int \mathrm{e}^{-\frac{(e_{i-1} - \frac{1}{2}x_i)^2}{1}}\, de_{i-1} \\
&= \frac{1}{\sqrt{4\pi}}\, \mathrm{e}^{-\frac{x_i^2}{4}},
\end{aligned}$$

which confirms that X_i has a Gaussian density with variance 2. From equations 3.21 and 3.22 it then follows that the first-order differential entropy of the signal is $h_1(X_i) = \frac{1}{2}\log(4\pi\mathrm{e})$.

Next, we determine the differential entropy rate. We do this by first considering a sequence of k samples. Assuming a particular choice at the sequence endpoints, the relation between the vector e^k and the vector x^k can be written as $x^k = Ae^k$ where

$$A = \begin{bmatrix}
1 & 0 & 0 & \cdots & & & \\
1 & 1 & 0 & \cdots & & & \\
0 & 1 & 1 & \cdots & & & \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\
 & & & \cdots & 1 & 1 & 0 \\
 & & & \cdots & 0 & 1 & 1
\end{bmatrix}.$$

We note that the determinant of A is unity, $\det(A) = 1$. Then, using theorem 9, we see that

$$\begin{aligned}
\frac{1}{k}h(X^k) &= h(E) + \frac{1}{k}\log(\det(A)) \\
&= h(E) \\
&= \frac{1}{2}\log(2\pi\mathrm{e}).
\end{aligned}$$

When k approaches infinity, the vectors approach the stationary process X_i and we have, therefore, shown that $h_\infty(X_i) = \frac{1}{2}\log(2\pi\mathrm{e})$.

Next, we derive the differential entropy for the multi-variate normal distribution. This differential entropy is important as such, but it also forms the basis for obtaining the differential entropy of Gaussian processes. Since translation does not affect the differential entropy, (see equation 3.4) we assume the multi-variate distribution to be zero-mean

without loss of generality. The k**-dimensional multi-variate normal density** is then defined as

$$f_{X^k}(x^k) = \frac{1}{\sqrt{(2\pi)^k \det(R)}} \mathrm{e}^{-\frac{1}{2} x^{kT} R^{-1} x^k}, \tag{3.23}$$

where R is the k-dimensional covariance matrix[3] of the random vector and $\det(\cdot)$ indicates determinant. The evaluation of the differential entropy (in nats) is straightforward. We note that R is symmetric and, in general, full rank. Thus, an invertible matrix B exists such that $B^T B = R$. The vector $y^k = B^{-1} x^k$ has a diagonal covariance matrix and

$$\begin{aligned}
h(X^k) &= -\int f_{X^k}(x^k) \log(f_{X^k}(x^k)) dx^k \\
&= -\log(\frac{1}{\sqrt{(2\pi)^k \det(R)}}) + \frac{1}{2} \int f_{X^k}(x^k) x^{kT} R^{-1} x^k dx^k \\
&= -\log(\frac{1}{\sqrt{(2\pi)^k \det(R)}}) + \frac{1}{2} \int f_{Y^k}(y^k) y^{kT} B^T R^{-1} B y^k dy^k \\
&= -\log(\frac{1}{\sqrt{(2\pi)^k \det(R)}}) + \frac{1}{2} \int f_{Y^k}(y^k) y^{kT} y^k dy^k \\
&= \frac{1}{2} \log((2\pi)^k \det(R)) + \frac{1}{2} k \\
&= \frac{1}{2} \log((2\pi \mathrm{e})^k \det(R)),
\end{aligned} \tag{3.24}$$

where we used that $dx^k = |\det(B)| dy^k$ and $f_{X^k}(x^k) = f_{Y^k}(y^k)/|\det(B)|$. We write this as a theorem:

Theorem 12 *The differential entropy of a Gaussian random vector X^k with covariance matrix R is*

$$h(X^k) = \frac{1}{2} \log((2\pi \mathrm{e})^k \det(R)). \tag{3.25}$$

Since the determinant is invariant under unitary transforms [10], we can also write equation 3.25 as

$$h(X^k) = \frac{1}{2} \log \left((2\pi \mathrm{e})^k \prod_{i=1}^{k} \lambda_i \right), \tag{3.26}$$

where the λ_i are the eigenvalues of the k-dimensional covariance matrix R. On a per-dimension basis, we can write this as

$$\frac{1}{k} h(X^k) = \frac{1}{2} \log(2\pi \mathrm{e}) + \frac{1}{2k} \sum_{i=1}^{k} \log(\lambda_i). \tag{3.27}$$

Example 3.7: Mutual information of components of bi-variate Gaussian
We consider a two-dimensional vector $X^2 = [X_1, X_2]^T$ with a covariance matrix

$$R = \begin{bmatrix} \sigma_1^2 & \rho_{12}\sigma_1\sigma_2 \\ \rho_{12}\sigma_1\sigma_2 & \sigma_2^2 \end{bmatrix},$$

[3] The covariance matrix of X^k is defined as $R_{ij} = \mathrm{E}[(X_i - \mathrm{E}[X_i])(X_j - \mathrm{E}[X_j])]$. Thus, a covariance matrix R must be positive semidefinite, i.e., $y^{kH} R y^k \geq 0$ for all $y^k \in \mathbb{R}^k$.

where ρ_{12} is a correlation coefficient. The mutual information between the components of the Gaussian vector is then

$$\begin{aligned} I(X_1;X_2) &= h(X_1)+h(X_2)-h(X_1,X_2) \\ &= \frac{1}{2}\log(2\pi\mathrm{e}\sigma_1^2)+\frac{1}{2}\log(2\pi\mathrm{e}\sigma_2^2)-\frac{1}{2}\log((2\pi\mathrm{e})^2\det(R)) \\ &= \frac{1}{2}\log(2\pi\mathrm{e}\sigma_1^2)+\frac{1}{2}\log(2\pi\mathrm{e}\sigma_2^2)-\frac{1}{2}\log((2\pi\mathrm{e})^2\sigma_1^2\sigma_2^2(1-\rho_{12}^2)) \\ &= -\frac{1}{2}\log(1-\rho_{12}^2). \end{aligned}$$

Example 3.8: A simple FIR filter, continued
Theorem 12 facilitates the computation of additional information measures for the output of the FIR filter of example 3.6. To compute the second-order entropy, we note that the covariance matrix of two sequential samples of the process X_i is

$$\begin{aligned} A &= \begin{bmatrix} \mathrm{E}[X_iX_i] & \mathrm{E}[X_{i-1}X_i] \\ \mathrm{E}[X_iX_{i-1}] & \mathrm{E}[X_{i-1}X_{i-1}] \end{bmatrix} \\ &= \begin{bmatrix} \mathrm{E}[E_iE_i]+\mathrm{E}[E_{i-1}E_{i-1}] & \mathrm{E}[E_{i-1}E_{i-1}] \\ \mathrm{E}[E_{i-1}E_{i-1}] & \mathrm{E}[E_{i-1}E_{i-1}]+\mathrm{E}[E_{i-2}E_{i-2}] \end{bmatrix}. \\ &= \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}, \end{aligned}$$

which has eigenvalues 3 and 1. It then follows from equation 3.27 that

$$h_2(X_i)=\frac{1}{2}h(X^2)=\frac{1}{2}\log(2\pi\mathrm{e}\sqrt{3}).$$

As it should be, $h_\infty(X_i)\le h_2(X_i)\le h_1(X_i)$. Furthermore, we see that

$$\begin{aligned} h(X_i|X_{i-1}) &= h(X_i,X_{i-1})-h(X_i) \\ &= \frac{1}{2}\log((2\pi\mathrm{e})^2 3)-\frac{1}{2}\log(4\pi\mathrm{e}). \\ &= \frac{1}{2}\log(2\pi\mathrm{e}\frac{3}{2}). \end{aligned}$$

Finally, we discuss the case of a discrete zero-mean Gaussian process, X_i, with memory[4]. To this purpose, we first consider some properties of discrete-time stationary processes in more detail. For a (wide-sense) stationary discrete-time process, we have that $R_{ij}=R_{i-j}$. We note that the equivalent of multiplication of a vector by the covariance matrix is now the convolution operation $\sum_{n\in\mathcal{Z}} R_{i-n}f_n$ where f_n is a discrete-time function. Eigenfunctions g_n of the time-invariant convolution operation must satisfy

$$\sum_{n\in\mathcal{Z}} R_{i-n}g_n=\lambda g_i \tag{3.28}$$

or, equivalently, by taking the discrete-time Fourier transform,

$$R(\mathrm{e}^{j\omega})g(\mathrm{e}^{j\omega})=\lambda g(\mathrm{e}^{j\omega}), \tag{3.29}$$

where $R(\mathrm{e}^{j\omega})$ is the power spectrum[5] of the stationary process X_i,

$$R(\mathrm{e}^{j\omega})=\sum_{n\in\mathcal{Z}} R_n\mathrm{e}^{-j\omega n}. \tag{3.30}$$

[4] A discrete-time process is said to have memory when its samples are not independent.
[5] Note that, with this notation, $R(z)$ is the z-transform of R_i.

For the general case that $R(e^{j\omega})$ is not constant, only functions that are nonzero at a single point can satisfy equation 3.29. Such functions integrate to zero and are of no practical use for our purpose. To eliminate this problem, we resort to generalized functions: the Dirac delta functions, which have the property that they integrate to one, but have vanishing support. The eigenfunctions of 3.28 are thus the complex exponentials $g_n = e^{j\omega n}$ and their eigenvalues are $R(e^{j\omega})$.

The above reasoning suggests how we can generalize equation 3.27 to the case of a Gaussian process with memory. For this we need a theorem that was first proven by Szegö [11, 12], and that is natural in light of our above, mathematically nonrigorous, discussion. A less general form of **Szegö's theorem** states:

Theorem 13 *Let $\{\lambda_i\}_{i\in\{1,\cdots,k\}}$ be the eigenvalues of a Hermitian Toeplitz matrix, with as first column $[R_0, \cdots, R_k]$, let $R(e^{j\omega})$ be the discrete-time Fourier transform of that first column, and let $R(e^{j\omega})$ be a Riemann integrable and bounded power spectrum on $[-\pi, \pi]$. Then, for any function $F(\cdot)$ continuous on the range of $R(e^{j\omega})$, we have that*

$$\lim_{k\to\infty} \frac{1}{k} \sum_{i=1}^{k} F(\lambda_i) = \frac{1}{2\pi} \int_{-\pi}^{\pi} F(R(e^{j\omega}))d\omega. \tag{3.31}$$

Theorem 13 states that the distribution of the discrete eigenvalues of the matrix R is asymptotically equal to the distribution of the power spectrum $R(e^{j\omega})$. That is, if a is the support for a certain range of power-spectrum values, then $a/2\pi$ is asymptotically identical to the fraction of eigenvalues of R that fall within that same range. This is illustrated in figure 3.4.

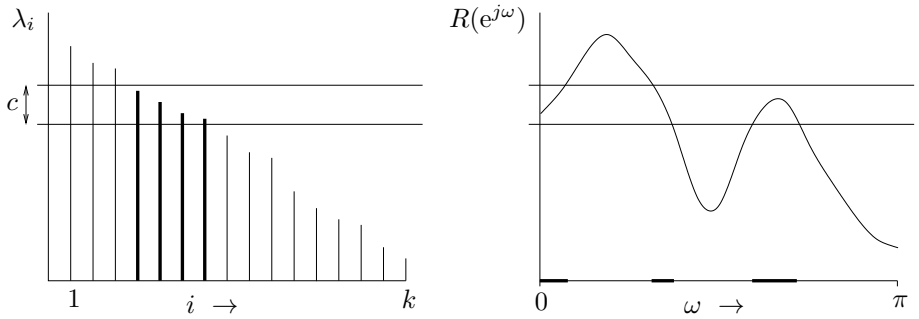

Figure 3.4: The essence of the Szegö's theorem, obtained by letting $F(\cdot)$ approximate a square window. On the left the (ordered) eigenvalues of R, on the right the power spectrum $R(e^{j\omega})$. The fraction of the eigenvalues having values in a range c (four out of a total of k=16) is asymptotically (that is, for $k \to \infty$) identical to the fraction of the support on the interval $[0, 2\pi]$ of the power-spectrum falling in the same range, indicated by the bold sections on the horizontal axis. Since the power spectrum is symmetric, we selected to display the spectrum over $[0, \pi]$, rather than $[0, 2\pi]$.

It is now straightforward to extend our earlier result. We start with equation 3.27 and take the limit $k \to \infty$ for both sides. Using Szegö's theorem, with the function $\log(\cdot)$ for $F(\cdot)$, we obtain the **differential entropy rate for a Gaussian process with**

memory:

$$h_\infty(X_i) = \frac{1}{2}\log(2\pi\mathrm{e}) + \frac{1}{4\pi}\int_{-\pi}^{\pi}\log(R(\mathrm{e}^{j\omega}))d\omega. \tag{3.32}$$

We will use this result in section 4.4, when we discuss spectral estimation.

Example 3.9: Entropy rate of iid Gaussian process
The autocorrelation function of an iid Gaussian process with variance σ^2 is

$$R_i = \begin{cases} \sigma^2, & i = 0 \\ 0, & i \in \mathcal{Z} - \{0\}. \end{cases}$$

We thus have

$$R(\mathrm{e}^{j\omega}) = \sigma^2$$

and equation 3.32 becomes

$$\begin{aligned} h_\infty(X_i) &= \frac{1}{2}\log(2\pi\mathrm{e}) + \frac{1}{2}\log(\sigma^2) \\ &= \frac{1}{2}\log(2\pi\mathrm{e}\sigma^2). \end{aligned}$$

Naturally, the same result is more easily obtained by realizing that for an iid process $h_1(X_i) = h_\infty(X_i)$ and that $h_1(X_i) = \frac{1}{2}\log(2\pi\mathrm{e}\sigma^2)$.

It is straightforward to obtain the redundancy rate of a Gaussian process:

$$\begin{aligned} \rho(X_i) &\equiv h_1(X_i) - h_\infty(X_i) \\ &= \frac{1}{2}\log(\sigma^2) - \frac{1}{4\pi}\int_{-\pi}^{\pi}\log(R(\mathrm{e}^{j\omega}))d\omega \\ &= \frac{1}{2}\log(\frac{1}{2\pi}\int_{-\pi}^{\pi}R(\mathrm{e}^{j\omega})d\omega) - \frac{1}{4\pi}\int_{-\pi}^{\pi}\log(R(\mathrm{e}^{j\omega}))d\omega. \end{aligned} \tag{3.33}$$

Equation 3.33 states that the redundancy rate is half the difference between the log of the mean of the power spectrum and the mean of the log of the power spectrum. It is easily seen that this measure is zero for a flat power spectrum and takes positive values for a colored power spectrum. Thus, for Gaussian signals a flat power spectrum corresponds to no redundancy rate, while a strongly colored power spectrum generally corresponds to a high redundancy rate. We return to this subject in section 10.2.9.

Example 3.10: Redundancy rates of vowel sounds

In table 3.1, the redundancy rate associated with the spectral envelope of vowel sounds in speech sampled at 8 kHz is shown, assuming that the signal is Gaussian. The spectral envelope is associated with the vocal-tract configuration. The spectral envelope was modeled based on resonance specifications given by Fant [13] (shown in table 3.2). In addition to these formant frequencies, a tilt of approximately -6 dB per octave (doubling of the frequency) was applied by adding a zero at 0.90. It is seen that there is typically a redundancy of about 2 bits per sample and that the "o" sound contains a particularly high level of redundancy. The redundancy rates should be compared to coding rates of between 1 and 2 bit per sample for typical mobile telephone coding systems and the coding rate of 8 bit per sample for the conventional wireline telephone network.

Table 3.1: Redundancy rates (in bits per sample) associated with the spectral envelopes of some vowel sounds assuming Gaussianity for 8 kHz sampling rate.

sound	"a"	"e"	"i"	"o"	"y"
$\rho(X_i)$	2.1	1.1	2.1	5.6	1.9

Table 3.2: Center frequencies, c(i), and bandwidths, bw(i), of the four resonances of vowel sounds [13].

sound	c(1)	bw(1)	c(2)	bw(2)	c(3)	bw(3)	c(4)	bw(4)
"a"	582	94	940	91	2480	107	3290	199
"e"	415	54	1979	101	2810	318	3450	50
"i"	256	43	2066	59	2960	388	3400	174
"o"	232	53	596	59	2395	388	3850	174
"y"	256	61	1928	57	2421	65	3300	43

3.5 Problems

1. Assuming Riemann integrability of all required densities, write a relation between the discrete and continuous measures for joint entropy, higher-order entropy, and entropy rate. Specify for each the conditions under which the relation is valid.

2. Consider the random vector $X = [X_1, X_2]$ with the density shown in figure 3.5 (dark areas have uniform density).

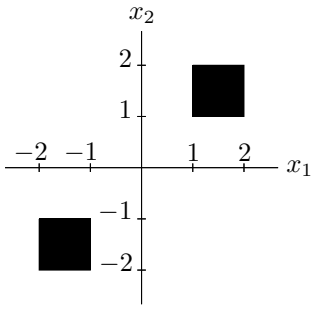

Figure 3.5: The density function of problem 2.

   (a) Compute the differential entropies $h(X_1)$ and $h(X_2)$.

   (b) Compute $h(X_1, X_2)$.

   (c) Compute $h(X_1|X_2)$ and $h(X_2|X_1)$.

   (d) Compute the mutual information $I(X_1; X_2)$. Give an explanation for your result.

3. Figure 3.6 shows the joint probability distribution for the inputs and outputs of two three-level quantizers. In the right figure the density is uniform in the dark areas.

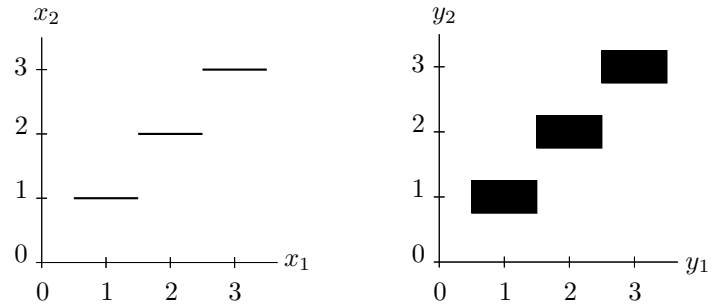

Figure 3.6: Three-level quantizer with noise-free and noisy reconstruction, respectively.

(a) Compute the entropy or differential entropy (select what is appropriate) for X_1, X_2, Y_1 and Y_2.

(b) Compute the conditional (differential) entropy of X_2 given X_1 and vice versa.

(c) Compute the conditional (differential) entropy of Y_2 given Y_1 and vice versa.

(d) Compute the mutual informations $I(X_1; X_2)$ and $I(Y_1; Y_2)$. Give an interpretation of the results.

4. Find the probability densities that maximize the differential entropy for the following cases:

   (a) The density can be nonzero only on the intervals $[2n, 2n+1]$, $n = 0, \cdots, N-1$.

   (b) The density is nonzero in the interval $[0, 1]$ and the expected value of the random variable is $1/4$.

5. Show examples with continuous distributions where $I(X; Y|W) > I(X; Y)$ and $I(X; Y|W) < I(X; Y)$.

6. By means of inequalities show that, given a fixed variance, the normal density has maximum differential entropy.

7. Consider the filter with transfer function $H(z) = \alpha_1 + \alpha_2 z^{-1}$. Let E_i be the input process and X_i the output process. Since one E_{i-2} and $H(X_i)$ are independent, we have $h(X_i) = h(X_i|E_{i-2})$. This leads one to think that $h(X_i|X_{i-1}) = h(X_i|X_{i-1}, E_{i-2})$ might hold and that this may be a simpler way to solve example 3.8. Prove that this equality is incorrect.

8. Consider a Markov chain where $p_{X_n, X_{n+1}}(x_{n+1}|x_n)$ is the transition probability from x_n to x_{n+1}. Let $p_X(x_n)$ be a stationary distribution for the Markov chain.

   (a) Prove that for any distribution $q_{X_n}(x_n)$ we have that

   $$H(q_{X_n}(x_n)||p_X(x_n)) \geq H(q_{X_{n+1}}(x_{n+1})||p_X(x_{n+1}))$$

   Hint: consider first the joint distributions, $H(q_{X_n, X_{n+1}}(x_n, x_{n+1})||p_{X_n, X_{n+1}}(x_n, x_{n+1}))$.

   (b) Give examples where the entropy of $q_{X_n}(x_n)$ increases and decreases with time.

   (c) Are your results consistent with an increasing entropy in the universe?

9. The μ-law compression characteristic used in North-America to compand the amplitude x of the samples of a discrete speech signals prior to coding is

$$c(x) = x_{\max} \frac{\log_e(1 + \mu|x|/x_{\max})}{\log_e(1 + \mu)} \text{sign}(x),$$

with $\mu = 255$. In the following, model the speech signal as a Gaussian sequence that repeats every 5 ms.

   (a) Create a set signals consisting of six signal levels of your periodic signal, increasing loudness in steps of 12 dB, the lowest being at your hearing threshold. Add white noise with a uniform amplitude distribution to model quantization noise and provide an estimate at what SNR (in dB) the noise becomes inaudible for each of the levels.

   (b) You want to estimate the bit rate required for a coding method which does not account for dependencies between signal samples. Using your previous audibility-threshold and assuming a Gaussian signal distribution, what is the minimum bit rate required for transparent quality at each of the signal levels with a uniform quantizer? Derive explicitly any equations you require and add your results to the table of SNR values from the previous question.

   (c) Estimate the minimum bit rate required to ensure transparency if all input levels occur with equal probability and you have to use a fixed encoder and decoder?

   (d) Explain how you can use the μ-law compander to reduce the rate if you don't know the incoming signal level. Without further experiments, estimate the rate required for transparent μ-law companded speech for the dynamic range of 60 dB you experimented with.

10. Consider a discrete stationary speech process X_i, a discrete stationary noise process N_i, and a contaminated signal $Y_i = X_i + N_i$. X_i and N_i are Gaussian and independent.

   (a) Derive an expression for the mutual information rate between the signals X_i and Y_i in terms of their spectral power densities $R_X(\mathrm{e}^{j\omega})$ and $R_N(\mathrm{e}^{j\omega})$.

   (b) The noise variance is σ_N^2. You can, however, control the shape of the noise power spectrum. Your objective is to maximize the mutual information rate between X_i and Y_i by a proper shaping of $R_N(\mathrm{e}^{j\omega})$. For your derivations you may assume that the signal-to-noise ratio is very high and your answer may be in the form of a simple algorithm.

11. This problem deals with entropy vs differential entropy and its relation to sampling and quantization. Consider a continuous-amplitude, discrete-time, Gaussian, white-noise (flat power spectrum) process, X_i, sampled at 8 kHz. The process has a variance $\sigma_X^2 = 10^6$.

   (a) Compute the first-order differential entropy and the differential entropy rate of the process X_i.

   (b) The samples are quantized with a uniform quantizer that has step size 2 (two). We denote the quantized discrete process as $Y_i = \mathcal{Q}(X_i)$. Compute (or find a good approximation) of the first-order entropy and the entropy rate of the process Y_i.

(c) The process X_i is upsampled to 16 kHz, then low-pass filtered with 4 kHz cut-off (assume an ideal filter) to render the process V_i. Provide the first-order differential entropy and the differential entropy rate of V_i.

12. Consider two real variables X, Y, both with rectangular distribution with unit support ($f_X(x) = 1$ over region of length 1).

(a) Plot two joint distributions for X and Y: one where the mutual information is minimized, and one where the mutual information is maximized.

(b) Compute both the minimum and maximum mutual information between X and Y.

(c) Consider now a third variable Z with alphabet $\{0, 1\}$. Both X and Y are dependent on Z. Without changing the unit support marginal distributions for X and Y, describe and plot a joint distribution for X, Y, and Z, where knowing Z *increases* the mutual information between X and Y.

(d) Under the same conditions, describe and plot a joint distribution for X, Y, and Z, where knowing Z *decreases* the mutual information between X and Y.

4

Estimation of Probability Distributions

4.1 Introduction

Probability distributions play a central role in source coding. Information-theoretical measures of a random variables, or sets of random variables, are fully determined by their probability distributions. As is seen in chapters 5 and 7, many practical coding methods require knowledge of the probability distribution. However, in practical applications, the probability distribution is generally not known a-priori. In this chapter we discuss methods to estimate the probability distribution from a set of data.

To permit a discussion on the estimation of distributions, we must first define their description. To facilitate practical estimation, it is convenient to specify the distribution with a countable set of parameters. As is common practice, we assume the random variables to be either discrete or continuous. For discrete random variables the description is straightforward: the parameter set is simply the countable set of probabilities of the alphabet. To describe probability density functions, we assume a model and then estimate the discrete set of model parameters. In this case, the error in the estimated distribution will be due to both the range of the model and the error in the estimated parameters.

Our first discussions of probability distribution estimation will be based on three methods. The first is the maximum likelihood method. This results in that distribution (within the range of modeled distributions) for which the observed data have the highest probability. The second method is the maximum-a-posteriori (MAP) estimation method, which determines the set of parameters that have the highest probability given the data. As is shown in this chapter, this is equivalent to an extension of the maximum likelihood method that accounts for prior knowledge about the probability distribution. If no prior knowledge exists, then the MAP method reduces to the maximum likelihood method. The third estimation method is a further generalization. In this third method, a penalty is assigned to the severity of the estimation error (of the probability distribution) and we then find the probability distribution that minimizes this error, given the

observed data.

Occam's razor effectively states that one should always select the simplest solution possible. In information theory this naturally leads to the maximum-entropy distribution (the distribution with the highest uncertainty about the random variable) for a given set of constraints on the probability distribution. The constraints are generally formulated in terms of known expectations of functions of the random variables. In practice, the expectations are often set equal to the corresponding observed averages over a data base.

The estimation of probabability distributions leads to so-called *plug-in* estimators of information-theoretical measures. Naturally, the quality of the resulting estimates depend on the quality of the probability distribution. While this chapter is dedicated to the estimation of probability distributions, it should be noted that competitive alternatives to plug-in estimators exist. For example, methods exist for the estimation of the differential entropy that directly relate differential entropy to nearest neighbor distances observed on data [14, 15].

In this chapter, we first discuss the estimation of probability mass functions in section 4.2. We then discuss the estimation of probability density functions in section 4.3, and conclude with a discussion of maximum-entropy distributions in section 4.4.

4.2 Probability-Mass Estimation

In this section, we describe methods for estimating the probability-mass function from the processed data. We discuss three methods for estimating the probability-mass function. We use the following notation. The symbols have alphabet $\mathcal{A}$. The random variable K_x counts the number of appearances of x in an independent identically distributed (iid) sequence of k symbols. For the observed sequence we have $K_x = k_x$. For a probability mass function of the random variable X we write $p_x \equiv p_X(x)$. In two of the methods, the probabilities themselves are random variables, which we indicate by using the notation P_x.

4.2.1 Maximum-Likelihood Estimation

In maximum-likelihood estimation, the probability mass distribution is interpreted as a deterministic set of values. We denote by $P(K_x = k_x|p_x)$ the probability of observing the value x for k_x observations from of a total of k observations, given the probability mass function p_x. The resulting probability distribution is the binomial distribution:

$$P(K_x = k_x|p_x) = C^k_{k_x}\ p_x^{k_x}(1-p_x)^{k-k_x}, \tag{4.1}$$

where

$$C^k_{k_x} = \frac{k!}{k_x!(k-k_x)!}. \tag{4.2}$$

The maximum-likelihood estimate for p_x is the value that maximizes the likelihood of the observations, $\underset{p_x\in[0,1]}{\text{argmax}}\ P(K_x = k_x|p_x)$. We first determine a stationary point for p_x

in $[0, 1]$. We first differentiate the binomial distribution with respect to p_x, and set the result equal to zero:

$$\begin{aligned} 0 &= k_x p_x^{k_x-1}(1-p_x)^{k-k_x} - (k-k_x)p_x^{k_x}(1-p_x)^{k-k_x-1} \\ &= \left(k_x(1-p_x) - (k-k_x)p_x\right) p_x^{k_x-1}(1-p_x)^{k-k_x-1}, \end{aligned} \tag{4.3}$$

where we omitted the factor $C_{k_x}^k$. The stationary point we are looking for is the solution of this equation. It is straightforward to show that this stationary point is a maximum, thus forming the maximum-likelihood estimate. It is

$$\hat{p}_x^{(ML)} = \frac{k_x}{k}. \tag{4.4}$$

The maximum-likelihood method justifies using straightforward empirical probability mass functions in adaptive codes. However, the estimate is not very useful for small k, where it varies strongly from one set of observed data to another.

4.2.2 Maximum A-Posteriori (MAP) Estimation

The maximum a-posteriori (MAP) estimate can be considered a generalization of the maximum-likelihood procedure. In the MAP estimate, each value of the probability distribution P_x is a random variable. The goal of the MAP procedure is to find the p_x that maximizes the a-posteriori probability $P(P_x = p_x|K_x = k_x)$. Using Bayes rule we have

$$\begin{aligned} \hat{p}_x^{(MAP)} &= \operatorname*{argmax}_{p_x\in[0,1]} P(P_x = p_x|K_x = k_x) \\ &= \operatorname*{argmax}_{p_x\in[0,1]} \frac{P(K_x = k_x|P_x = p_x)\, P(P_x = p_x)}{P(K_x = k_x)} \\ &= \operatorname*{argmax}_{p_x\in[0,1]} P(K_x = k_x|P_x = p_x)\, P(P_x = p_x), \end{aligned} \tag{4.5}$$

where we omitted the a-priori probability $P(K_x = k_x)$ since it does not depend on the specific value p_x. The probability $P(K_x = k_x|P_x = p_x)$ is identical to $P(K_x = k_x|p_x)$ of equation 4.1. However, we must determine an expression for $P(P_x = p_x)$ to find the estimate $\hat{p}_x^{(MAP)}$.

We now determine an a-priori density $P(P_x = p_x)$ appropriate for the case that we have no specific knowledge about the probability mass function. Probability mass functions are discrete functions that sum to unity and have a range $[0, 1]$. The set of probability mass functions of an alphabet $\mathcal{A}$ defines a set of points (region) $\mathcal{G}_\mathcal{A}$ in a subspace of $|\mathcal{A}| - 1$ dimensions of the space defined by $p_1, \cdots, p_{|\mathcal{A}|}$ (where we assume a trivial alphabet):

$$\mathcal{G}_\mathcal{A} = \{\{P_x\}_{x\in\mathcal{A}} \,:\, P_x \in [0, 1], \sum_{x\in\mathcal{A}} P_x = 1\}. \tag{4.6}$$

The constraints on the sum and the range of the probability mass function facilitate a simple geometric interpretation of $\mathcal{G}_\mathcal{A}$. Consider an $\mathcal{A}$-dimensional linear space spanned by $p_1, \cdots, p_{|\mathcal{A}|}$. The range constraint means that G_A lies in the hypercube that has the unit vectors as edges. The sum constraint implies that G_A lies in the dimension $|\mathcal{A}| - 1$ hyperplane that includes all unit vectors (i.e., $[1, 0, \cdots, 0]$, $[0, 1, \cdots, 0]$, lie in this plane).

$\mathcal{G}_\mathcal{A}$ is the intersection of the hyperplane and the cube. Figure 4.1 illustrates the set of admissabile probability mass functions for an alphabet with three symbols. Note that there are more probability mass functions with near-uniform probability distribution than with nonuniform probability distribution.

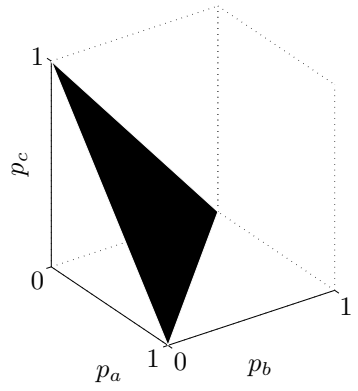

Figure 4.1: The set of admissable probability mass functions for the alphabet $\{a, b, c\}$.

Since we have no a-priori knowledge about the probability mass functions, we assume that the admissable probability mass functions (those in $G_\mathcal{A}$) are equally probable. This starting point is similar in philosophy as that used in ensemble theory discussed in 2.2.

At least in principle, we can compute the probability $P(P_x = p_x)$, by means of integrating the probability density of the probability mass functions over the remaining dimensions $\{p_y\}_{y\in\mathcal{A},y\neq x}$. We evaluate the integral using geometric considerations. We note that if symbol x has probability one, $P_x = 1$, then all other symbols have zero probability, $P_y = 0, y \neq x,\ y \in \mathcal{A}$. Thus, $P_x = 1$ corresponds to a single point in $G_\mathcal{A}$ and we must have $P(P_x = 1) = 0$. It is also easily seen that the unity-sum constraint implies that $P(P_x = 1 - \epsilon) \propto \epsilon^{|\mathcal{A}|-2}$. Thus, the marginal density $P(P_x = p_x)$ is

$$P(P_x = p_x) = (|\mathcal{A}| - 1)(1 - p_x)^{|\mathcal{A}|-2}, \tag{4.7}$$

where the normalization constant was determined using $\int_0^1 P(P_x = p_x)dp_x = 1$.

From equations 4.1 (which equals $P(K_x = k_x|P_x = p_x)$ for the random P_x case) and 4.7 we see that

$$P(K_x = k_x|P_x = p_x)\, P(P_x = p_x) = (|\mathcal{A}| - 1)C_{k_x}^k p_x^{k_x}(1 - p_x)^{k-k_x+|\mathcal{A}|-2}. \tag{4.8}$$

We note that this equation is of the same form as equation 4.1 and that we, thus, can find $\hat{p}_x^{(MAP)}$ in the same manner. The MAP estimate is

$$\hat{p}_x^{(MAP)} = \frac{k_x}{k + |\mathcal{A}| - 2}. \tag{4.9}$$

This simple MAP estimate is somewhat unsatisfactory for our purpose, since the estimated probability mass function does not sum to unity. It can, thus, not be used for coding without modifications. Unfortunately, simply adding the unity-sum constraint leads to a nonlinear problem.

4.2.3 Bayesian Inference

The MAP solution gives the maximum in the a-posteriori probability density $P(P_x = p_x|K_x = k_x)$. Alternatively, we can define a Bayes cost function and minimize the mean cost, which is referred to as **Bayes risk**. For the squared error $(\hat{p}_x - P_x)^2$ the Bayes risk is

$$\mathrm{E}[(\hat{p}_x - P_x)^2|K_x = k_x] = \int_0^1 (\hat{p}_x - p_x)^2 P(P_x = p_x|K_x = k_x)dp_x. \tag{4.10}$$

This criterion is minimized by the mean value of P_x,

$$\hat{p}_x^{(Laplace)} = \mathrm{E}[P_x|K_x = k_x] = \int_0^1 p_x P(P_x = p_x|K_x = k_x)dp_x. \tag{4.11}$$

We rewrite the probability in the argument of the integral with Bayes rule:

$$P(P_x = p_x|K_x = k_x) = \frac{P(K_x = k_x|P_x = p_x)\,P(P_x = p_x)}{P(K_x = k_x)}. \tag{4.12}$$

Assuming no specific prior knowledge, $P(K_x = k_x|P_x = p_x)$ and $P(P_x = p_x)$ are given by equations 4.1 and 4.7. We evaluate the a-priori probability $P(K_x = k_x)$:

$$\begin{aligned} P(K_x = k_x) &= \int_0^1 P(K_x = k_x|P_x = p_x)P(P_x = p_x)dp_x \\ &= (|\mathcal{A}| - 1)C_{k_x}^k \int_0^1 p_x^{k_x}(1 - p_x)^{k-k_x+|\mathcal{A}|-2}dp_x \\ &= (|\mathcal{A}| - 1)C_{k_x}^k \frac{\Gamma(k_x + 1)\,\Gamma(k - k_x + |\mathcal{A}| - 1)}{\Gamma(k + |\mathcal{A}|)} \\ &= (|\mathcal{A}| - 1)C_{k_x}^k \frac{k_x!\,(k - k_x + |\mathcal{A}| - 2)!}{(k + |\mathcal{A}| - 1)!}, \end{aligned} \tag{4.13}$$

where we used a table [16] to find the solution to the definite integral $\int_0^1 x^n(1 - x)^m dx$ (this is a form of the beta function, e.g., [16, 17]) and where we note that, for integers n we have that $\Gamma(n + 1) = n!$.

Substituting equations 4.1, 4.7, and 4.13 into equation 4.12, we obtain

$$P(P_x = p_x|K_x = k_x) = p_x^{k_x}(1 - p_x)^{k-k_x+|\mathcal{A}|-2}\frac{(k + |\mathcal{A}| - 1)!}{k_x!\,(k - k_x + |\mathcal{A}| - 2)!}. \tag{4.14}$$

Using equation 4.14 in equation 4.11 gives then

$$\begin{aligned} \hat{p}_x^{(Laplace)} &= \int_0^1 p_x P(P_x = p_x|K_x = k_x)dp_x \\ &= \frac{(k + |\mathcal{A}| - 1)!}{k_x!\,(k - k_x + |\mathcal{A}| - 2)!}\int_0^1 p_x^{k_x+1}(1 - p_x)^{k-k_x+|\mathcal{A}|-2}dp_x \\ &= \frac{(k + |\mathcal{A}| - 1)!}{k_x!\,(k - k_x + |\mathcal{A}| - 2)!}\frac{\Gamma(k_x + 2)\,\Gamma(k - k_x + |\mathcal{A}| - 1)}{\Gamma(k + |\mathcal{A}| + 1)} \\ &= \frac{(k + |\mathcal{A}| - 1)!}{k_x!\,(k - k_x + |\mathcal{A}| - 2)!}\frac{(k_x + 1)!\,(k - k_x + |\mathcal{A}| - 2)!}{(k + |\mathcal{A}|)!} \\ &= \frac{k_x + 1}{k + |\mathcal{A}|}. \end{aligned} \tag{4.15}$$

Equation 4.15 is called **Laplace's rule**. The Laplace-rule estimate is reasonable for small k (having equal probabilities for $k = 0$, which is consistent with our observations around figure 4.1) and also provides a proper probability mass function.

4.3 Probability Density Estimation

A probability density of a continuous random variable is usually characterized with a countable set of parameters. Thus, we can distinguish two aspects in the estimation of probability density function: *i*) the design of an appropriate model for the density, *ii*) the estimation of the model parameters using the training data. A commonly used model is the mixture model, which consists of the summation of a set of **kernels** $f_{X;\theta_i}(x)$, each specified by a set of parameters θ_i. The mixture model is

$$f_{X;\theta}(x) = \sum_{i \in \mathcal{I}} p_I(i) f_{X;\theta_i}(x), \tag{4.16}$$

where $\mathcal{I} = \{1, \cdots, |\mathcal{I}|\}$ is the set of mixture component indices, $p_I(i)$ is a probability-mass function, and $\theta = \{p_I(i), \theta_i\}_{i \in \mathcal{I}}$. We note that $p_I(i)$ can be interpreted as the probability of the component. Most commonly, the kernels $f_{X;\theta_i}$ are Gaussian densities, but other kernels can also be used.

The motivation for using a mixture model is that, for a sufficiently large cardinality of $\mathcal{I}$ and for a suitable kernel, the model can approximate a large range of densities. Furthermore, the mixture models of equation 4.16 are convenient when estimating moments of the random variable X:

$$\begin{aligned} \mathrm{E}[X^m] &= \int x^m f_{X,\theta}(x) dx \\ &= \sum_{i \in \mathcal{I}} p_I(i) \int x^m f_{X;\theta_i}(x) dx, \end{aligned} \tag{4.17}$$

where the superscript m indicates the power.

The estimation of the parameters θ from training data is often not straightforward. In section 4.3.1 we discuss a measure of accuracy and the sections there-after describe the estimation of the parameter set θ. The methods that we use are related to those used in section 4.2 to estimate discrete probability distributions from training data. We first discuss the basic maximum-likelihood and maximum a-posteriori (MAP) estimation methods and then the so-called EM algorithm, which is commonly used to maximize the likelihood when analytic solutions cannot be found for the maximum-likelihood estimate.

4.3.1 Density Models and Information Measures

While mixture models are commonly used, it is difficult to assess their performance. A good measure of performance would be the relative entropy between the original density $f_X(x)$ and the model density $f_{X;\theta}(x)$,

$$h(f_X \| f_{X;\theta}) = \int f_X(x) \log(\frac{f_X(x)}{f_{X;\theta}(x)}) dx. \tag{4.18}$$

This measure is generally not directly useful since the original density $f_X(x)$ is unknown. However, since the relative entropy is nonnegative, the quantity

$$h(X) + h(f_X \| f_{X;\theta}) = -\int f_X(x) \log(f_{X;\theta}(x)) dx \tag{4.19}$$

approaches the differential entropy of X from above with increasing quality of the density model $f_{X;\theta}(x)$. The expression is easily approximated by exploiting the weak law of the large numbers (theorem 8). Averaging over N independent observations $\{x_{(n)}\}_{n=1,\cdots,N}$ (the parentheses around the subscript indicate that it is an index into the set and not a vector element) results in:

$$\begin{aligned} -\int f_X(x) \log(f_{X;\theta}(x)) dx &\approx -\mathrm{E}[\log(f_{X;\theta}(X))] \\ &\approx -\frac{1}{N} \sum_{n=1}^{N} \log(f_{X;\theta}(x_{(n)})). \end{aligned} \tag{4.20}$$

Equation 4.20 is a **Monte Carlo integration**, which generally requires a large computational expense.

From equations 4.19 and 4.20 we note that (assuming the Monte Carlo integration is sufficiently precise) the estimated differential entropy never underestimates the true differential entropy:

$$h(X) \leq -\frac{1}{N} \sum_{n=1}^{N} \log(f_{X;\theta}(x_{(n)})). \tag{4.21}$$

This means that we can compare the usefulness of various model densities by evaluating equation 4.20. Importantly, the model density that provides the smallest value is best.

The right-hand side of equation 4.20 is minus the log likelihood of the data given the density model. It is seen that the likelihood is closely related to the relative entropy of the original and model densities. Furthermore, it is clear that the differential entropy defines an upper bound on the log likelihood of the model.

Equation 4.20 is also commonly interpreted as a particular plug-in estimate of the entropy. For the mixture model, this plug-in estimate can be written as

$$h(X) \approx -\frac{1}{N} \sum_{n=1}^{N} \log(\sum_{i \in \mathcal{I}} p_I(i) \hat{f}_{X;\theta_i}(x_{(n)})). \tag{4.22}$$

4.3.2 Maximum-Likelihood Estimation of θ

In maximum-likelihood estimation[1], the parameter set θ is considered to be deterministic. We try to determine the parameter set θ from a set of N independent observations denoted by $\breve{x} = \{x_{(n)}\}_{n=1,\cdots,N}$. The **likelihood** of the observations $\breve{x}$ for a certain θ is

[1] Whether the supremum of the likelihood is a maximum depends on the modelling assumptions. In practice, the assumptions are selected such that maxima exist. Hence the name maximum-likelihood estimation is natural.

simply $f_{\check{X};\theta}(\check{x})$ for given $\check{x}$. The maximum-likelihood estimate for θ is then

$$\begin{aligned}\theta^{\mathrm{ML}} &= \operatorname*{argmax}_{\theta} f_{\check{X};\theta}(\check{x}) \\ &= \operatorname*{argmax}_{\theta} \prod_{n=1}^{N} f_{X;\theta}(x_{(n)}),\end{aligned} \tag{4.23}$$

where we exploited that the $x_{(n)}$ are independent. It is often convenient to use the so-called **log likelihood**, $\log(f_{\check{X};\theta}(\check{x}))$. The θ that maximizes the log likelihood is

$$\theta^{\mathrm{ML}} = \operatorname*{argmax}_{\theta} \sum_{n=1}^{N} \log(f_{X;\theta}(x_{(n)})). \tag{4.24}$$

Example 4.1: Maximum likelihood estimation of a Gaussian density
We want to estimate $\theta = \{\sigma^2, \mu\}$ in

$$f_{X,\theta}(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \mathrm{e}^{-\frac{(x-\mu)^2}{2\sigma^2}}.$$

The maximum (log) likelihood estimate of the parameter set $\{\mu, \sigma^2\}$ is

$$\{\mu_{\mathrm{ML}}, \sigma^2_{\mathrm{ML}}\} = \operatorname*{argmax}_{\{\mu,\sigma^2\}} \left(-\frac{N}{2} \log(\sigma^2) - \sum_{n=1}^{N} \frac{(x_{(n)} - \mu)^2}{2\sigma^2} \right).$$

It is convenient to find the estimate for μ first, since it does not depend on σ^2:

$$\begin{aligned}\mu_{\mathrm{ML}} &= \operatorname*{argmax}_{\mu} \; -\sum_{n=1}^{N} (x_{(n)} - \mu)^2 \\ &= \operatorname*{argmin}_{\mu} \; N\mu^2 - 2\mu \sum_{n=1}^{N} x_{(n)} \\ &= \operatorname*{argmin}_{\mu} \; (N\mu - \sum_{n=1}^{N} x_{(n)})^2 \\ &= \frac{1}{N} \sum_{n=1}^{N} x_{(n)}.\end{aligned}$$

For σ^2_{ML} we obtain

$$\begin{aligned}\sigma^2_{\mathrm{ML}} &= \operatorname*{argmax}_{\sigma^2} \left(-\frac{N}{2} \log(\sigma^2) - \sum_{n=1}^{N} \frac{(x_{(n)} - \mu_{\mathrm{ML}})^2}{2\sigma^2} \right) \\ &= \operatorname*{argmin}_{\sigma^2} \left(\log(\sigma^2) + \frac{\frac{1}{N}\sum_{n=1}^{N} (x_{(n)} - \mu_{\mathrm{ML}})^2}{\sigma^2} \right) \\ &= \frac{1}{N} \sum_{n=1}^{N} (x_{(n)} - \mu_{\mathrm{ML}})^2,\end{aligned}$$

where we have to employ calculus to perform the last step.

Example 4.2: ML estimation of a multi-variate Gaussian density
Using the same approach as example 4.1, we now estimate the parameters $\theta =$

$\{R, \mu^k\}$ (R is the covariance matrix R and μ^k the mean) of the multi-variate Gaussian distribution,

$$f_{X^k;\theta}(x) = \frac{1}{\sqrt{(2\pi)^k \det(R)}} \mathrm{e}^{-\frac{1}{2}(x^k-\mu^k)^T R^{-1}(x^k-\mu^k)},$$

from a set of N independent k-dimensional observation vectors $\check{x}^k = \{x^k_{(n)}\}_{n=1,\cdots,N}$.

Assuming that the maximum (log) likelihood estimate of the mean does not depend on R, we can write

$$\begin{aligned}
\mu^k_{\mathrm{ML}} &= \underset{\mu^k}{\operatorname{argmin}} \sum_{n=1}^{N} (x^k_{(n)} - \mu^k)^T R^{-1} (x^k_{(n)} - \mu^k) \\
&= \underset{\mu^k}{\operatorname{argmin}} \left(N \mu^{kT} R^{-1} \mu^k - 2 \sum_{n=1}^{N} \mu^{kT} R^{-1} x^k_{(n)} \right) \\
&= \underset{\mu^k}{\operatorname{argmin}} \, (\sum_{n=1}^{N} x^k_{(n)} - N\mu^k)^T R^{-1} (\sum_{n=1}^{N} x^k_{(n)} - N\mu^k) \\
&= \frac{1}{N} \sum_{n=1}^{N} x^k_{(n)},
\end{aligned}$$

which is indeed independent of R.

Next we estimate the covariance matrix R. For this we need knowledge of the matrix calculus results

$$\frac{\delta \log(\det(A))}{\delta A} = A^{-1},$$
$$\frac{\delta x^{kT} A x^k}{\delta A} = x^k x^{kT}.$$

Equating R^{-1} with A, we see that differentiating the log likelihood

$$\log(f_{\check{X}^k;\theta}(\check{x}^k)) = -\frac{kN}{2} \log(2\pi) + \frac{N}{2} \log(\det(R^{-1})) - \frac{1}{2} \sum_{n=1}^{N} (x^k_{(n)} - \mu^k)^T R^{-1} (x^k_{(n)} - \mu^k)$$

towards R^{-1} and setting the result to zero results in

$$R_{\mathrm{ML}} = \frac{1}{N} \sum_{n=1}^{N} (x^k_{(n)} - \mu^k_{\mathrm{ML}})(x^k_{(n)} - \mu^k_{\mathrm{ML}})^T.$$

This completes the maximum-likelihood estimation of the multi-variate Gaussian.

Example 4.3: Maximum-likelihood estimate of Laplacian density
We want to estimate the parameters $\theta = \{a, \mu\}$ in

$$f_{X;\theta}(x) = \frac{a}{2} \mathrm{e}^{-a|x-\mu|}.$$

Ignoring a constant term, the corresponding log likelihood of N observations $x_{(n)}$ is

$$\log(f_{\check{X};\theta}(\check{x})) = N \log(a) - \sum_{n=1}^{N} a|x_{(n)} - \mu|.$$

Differentiating towards μ and setting the result to zero we obtain

$$\sum_{n=1}^{N} \operatorname{sign}(x_{(n)} - \mu) = 0,$$

which implies that μ_{ML} is the median of $\{x_{(n)}\}_{n=1,\cdots,N}$. Once μ_{ML} is known, the solution for a is trivial:

$$a = \frac{N}{\sum_{n=1}^{N} |x_{(n)} - \mu_{\mathrm{ML}}|}.$$

4.3.3 Maximum A-Posteriori Estimation of θ

As was mentioned in section 4.2, the maximum-likelihood estimation procedure can be interpreted as a special case of the maximum a-posteriori (MAP) estimation procedure. For the MAP method, the deterministic vector θ is replaced with a random vector, which we denote by Θ. This replacement is reflected in our notation of the densities where $f_{X;\theta}(x)$ is replaced by $f_{X|\Theta}(x|\theta)$. In MAP estimation, the goal is to find a realization of Θ that maximizes its density given the observations $\check{x}$. We can write

$$\begin{aligned}
\theta^{\mathrm{MAP}} &= \operatorname*{argmax}_{\theta} f_{\Theta|\check{X}}(\theta|\check{x}) \\
&= \operatorname*{argmax}_{\theta} f_{\check{X}|\Theta}(\check{x}|\theta) \frac{f_{\Theta}(\theta)}{f_{\check{X}}(\check{x})} \\
&= \operatorname*{argmax}_{\theta} f_{\check{X}|\Theta}(\check{x}|\theta) f_{\Theta}(\theta) \\
&= \operatorname*{argmax}_{\theta} \Big(\log(f_{\check{X}|\Theta}(\check{x}|\theta)) + \log(f_{\Theta}(\theta)) \Big), \qquad (4.25)
\end{aligned}$$

which is similar to the derivation in 4.5. Equation 4.25 shows that the MAP estimate differs from the maximum likelihood estimate in an additional term in the criterion. This term accounts for the a-priori probability distribution $f_{\Theta}(\theta)$.

4.3.4 The Expectation-Maximization Algorithm

It is often impossible to find a simple analytic solution for the θ that maximizes the log likelihood criterion. In such cases, an effective iterative procedure that reduces the likelihood criterion at each iteration step is useful. An iterative algorithm that is commonly used for this purpose is the **expectation-maximization algorithm** or **EM algorithm**, which we describe in more detail below.

In the derivation of the EM algorithm, we have as goal to obtain a double maximization of an expression. Alternating these maximizations will result in an iterative procedure. We start with rewriting the log likelihood of N independent observations $\check{x} = \{x_{(n)}\}_{n=1,\cdots,N}$ of the random variable X. We introduce an additional random vector, Y, and consider the parameter set θ to specify the joint distribution, $f_{XY;\theta}(x, y)$, of X and Y. Note that we have not yet specified anything about possible dependencies between X and Y. The likelihood of a given set of observations $\check{X} = \check{x}$ can be written as

$$\begin{aligned}
\log(f_{\check{X};\theta}(\check{x})) &= \log(f_{\check{X};\theta}(\check{x})) \int f_{\check{Y}|\check{X};\theta'}(\check{y}|\check{x}) d\check{y} \\
&= \int f_{\check{Y}|\check{X};\theta'}(\check{y}|\check{x}) \log(f_{\check{X};\theta}(\check{x})) d\check{y} \\
&= \int f_{\check{Y}|\check{X};\theta'}(\check{y}|\check{x}) \log(\frac{f_{\check{X}\check{Y};\theta}(\check{x}, \check{y})}{f_{\check{Y}|\check{X};\theta}(\check{y}|\check{x})}) d\check{y}, \qquad (4.26)
\end{aligned}$$

where the integral is over the sample space of $\check{Y}$. The right-hand side of equation 4.26 can be interpreted as an expectation of $\log(\frac{f_{\check{X}\check{Y};\theta}(\check{X},\check{Y})}{f_{\check{Y}|\check{X};\theta}(\check{Y}|\check{X})})$, which is specified by θ, over the probability distribution $f_{\check{Y}|\check{X};\theta'}$, which is specified by θ'.

Next, we restrict our choice of Y. We choose our random vector Y such that if y is known, then x is fully specified, but not necessarily vice versa. In other words, the mapping from y to x is, in general, a many-to-one mapping. This is often expressed by referring to Y as the **complete data**, whereas X describes the observed data.

The choice of a deterministic mapping from y to x leads to singularities in density functions that can be avoided by defining integrals over $\check{y}$ to be on manifolds. We gloss over these technical difficulties here. The many-to-one mapping leads to the identity $f_{\check{X}\check{Y};\theta} = f_{\check{Y};\theta}$ and we rewrite the likelihood of equation 4.26 as

$$\log(f_{\check{X};\theta}(\check{x})) \quad = \quad \int f_{\check{Y}|\check{X};\theta'}(\check{y}|\check{x}) \log(\frac{f_{\check{Y};\theta}(\check{y})}{f_{\check{Y}|\check{X};\theta}(\check{y}|\check{x})}) d\check{y}. \tag{4.27}$$

Finding the θ that maximizes the likelihood, is then equivalent to finding the θ that maximizes the right-hand-side of equation 4.27. Similarly to the left-hand side, the solution is often analytically untractable. However, we are now ready to introduce the second maximization such that we obtain an iterative procedure by alternating the maximizations. We do this by adding an expression that has as maximum over the parameter space θ' the value zero and maximizing this over θ' (that is, we really add zero!). We select for the term[2] that has as maximum zero $-h(f_{\check{Y}|\check{X};\theta'}(\cdot|\check{x})||f_{\check{Y}|\check{X};\theta}(\cdot|\check{x}))$. The maximum likelihood can then be expressed as

$$\begin{aligned}
\max_{\theta} f_{\check{X},\theta}(\check{x}) &= \max_{\theta} \log(f_{\check{X},\theta}(\check{x})) \\
&= \max_{\theta} \int f_{\check{Y}|\check{X};\theta'}(\check{y}|\check{x}) \log(\frac{f_{\check{Y};\theta}(\check{y})}{f_{\check{Y}|\check{X};\theta}(\check{y}|\check{x})}) d\check{y} \\
&= \max_{\theta} \max_{\theta'} \Bigg(\int f_{\check{Y}|\check{X};\theta'}(\check{y}|\check{x}) \log(\frac{f_{\check{Y};\theta}(\check{y})}{f_{\check{Y}|\check{X};\theta}(\check{y}|\check{x})}) d\check{y} - \\
&\qquad h(f_{\check{Y}|\check{X};\theta'}(\cdot|\check{x})||f_{\check{Y}|\check{X};\theta}(\cdot|\check{x})) \Bigg) \\
&= \min_{\theta} \min_{\theta'} h(f_{\check{Y}|\check{X};\theta'}(\cdot|\check{x})||f_{\check{Y};\theta}(\cdot)),
\end{aligned} \tag{4.28}$$

where we exploited that the log likelihood does not depend on θ' (see equation 4.26) and that $\min_{\theta'} h(f_{\check{Y}|\check{X};\theta'}(\cdot|\check{x})||f_{\check{Y}|\check{X};\theta}(\cdot|\check{x})) = 0$.

Alternating the minimizations in equation 4.28 results in a minimization of the relative entropy $h(f_{\check{Y}|\check{X};\theta'}(\cdot,\check{x})||f_{\check{Y};\theta})$ and, thus, a maximization of the likelihood of equation 4.26. To make this iterative algorithm practical, we need the explicit solutions for the optimal values for θ given θ' and for θ' given θ. First, we note from 4.28 that the minimization over θ' corresponds to minimization of $h(f_{\check{Y}|\check{X};\theta'}(\cdot|\check{x})||f_{\check{Y}|\check{X};\theta}(\cdot|\check{x}))$, and

[2] A note on notation of the relative entropy. Relative entropy is a functional, i.e., it depends on functions and *not* on just a particular value the function takes. This is the reason why we generally write $h(f_X \| f_Y)$. We can also write $h(f_X(\cdot)\| f_Y(\cdot))$, but *not* the meaningless $h(f_X(x)\| f_Y(y))$. For conditional distributions the values of the conditioning variables are important and we, therefore, must write, for example, $h(f_{X,Z}(\cdot|z)\| f_Y(\cdot))$.

Table 4.1: The EM algorithm for iteratively maximizing the likelihood of the parameter set θ that specifies $f_{X;\theta}$. The function $Q(\cdot,\cdot)$ is defined in equation 4.31.

1. initialize $\theta = \theta'$
2. compute $\theta_{\text{opt}} = \text{argmax}_\theta \, Q(\theta;\theta')$
3. reset $\theta' := \theta_{\text{opt}}$
4. check whether iterations have converged; go to 2 if they have not

that this simply corresponds to selecting

$$\theta'_{\text{opt}} = \theta. \tag{4.29}$$

Second, we note that optimizing for θ corresponds to an expectation maximization, which motivates the name of the algorithm. We can rewrite this step as:

$$\begin{aligned}
\theta_{\text{opt}} &= \underset{\theta}{\text{argmin}} \; h(f_{\check{Y}|\check{X};\theta'}(\cdot|\check{x})||f_{\check{Y};\theta}) \\
&= \underset{\theta}{\text{argmin}} \left(\int f_{\check{Y}|\check{X};\theta'}(\check{y}|\check{x}) \log(f_{\check{Y}|\check{X};\theta'}(\check{y}|\check{x}))d\check{y} - \int f_{\check{Y}|\check{X};\theta'}(\check{y}|\check{x}) \log(f_{\check{Y};\theta}(\check{y}))d\check{y} \right) \\
&= \underset{\theta}{\text{argmax}} \int f_{\check{Y}|\check{X};\theta'}(\check{y}|\check{x}) \log(f_{\check{Y};\theta}(\check{y}))d\check{y}, \\
&= \underset{\theta}{\text{argmax}} \; Q(\theta;\theta'),
\end{aligned} \tag{4.30}$$

where we defined the cost function

$$\begin{aligned}
Q(\theta;\theta') &= \int f_{\check{Y}|\check{X};\theta'}(\check{y}|\check{x}) \log(f_{\check{Y};\theta}(\check{y}))d\check{y} \\
&= \int \prod_{n=1}^{N} f_{Y|X;\theta'}(y_{(n)}|x_{(n)}) \log(\prod_{n'=1}^{N} f_{Y;\theta}(y_{(n')}))d\check{y} \\
&= \sum_{n=1}^{N} \int f_{Y|X;\theta'}(y|x_{(n)}) \log(f_{Y;\theta}(y))dy.
\end{aligned} \tag{4.31}$$

The resulting EM algorithm is summarized in table 4.1.

For a particular problem, with a particular form for $f_{Y;\theta}(y)$, an explicit expression for θ_{opt} can often be found. Here we are interested in the optimization of the parameters of a mixture model. A straightforward calculus approach towards solving equation 4.24 does not work because we obtain a set of nonlinear equations, and that motivates us to use the EM algorithm.

Let us now consider the EM for the mixture model. In the mixture model approach, the set of parameters specifying the model is

$$\theta = \{\theta_i, p_I(i)\}_{i\in\mathcal{I}}. \tag{4.32}$$

The idea behind using the EM approach for the mixture model is simple: we select the complete data vector such that the optimization problem of equation 4.30 essentially

reduces to that the individual maximum likelihood estimation of the kernel densities. Since we optimize the parameters by differentiation, that means the objective is to create a cost function $Q_{\text{mixture}}(\theta;\theta')$ that is additive in terms depending on the elements of the set $\{\theta_i, p_I(i)\}_{i\in\mathcal{I}}$. To this purpose, we use $Y = \{X, I\}$ where I is the random index of the mixture component. Accounting for the fact that I is a *discrete* random variable, which means the integral is replaced by a sum, the cost function of equation 4.31 can be written as

$$\begin{aligned} Q_{\text{mixture}}(\theta;\theta') &= \sum_{n=1}^{N}\sum_{i\in\mathcal{I}} f_{X,I|X;\theta'}(x_{(n)}, i|x_{(n)}) \log(f_{X,I;\theta}(x_{(n)}, i)) \\ &= \sum_{n=1}^{N}\sum_{i\in\mathcal{I}} p_{I|X;\theta'}(i|x_{(n)}) \log(f_{X,I;\theta}(x_{(n)}, i)) \\ &= \sum_{n=1}^{N}\sum_{i\in\mathcal{I}} p_{I|X;\theta'}(i|x_{(n)}) \left(\log(f_{X;\theta_i}(x_{(n)})) + \log(p_I(i))\right), \end{aligned} \tag{4.33}$$

which displays the desired additivity in terms depending on the elements of the set θ. The probability distributions $p_{I|X;\theta'}(i|x_{(n)})$ (which are constant for the optimization of the cost function with respect to θ) can be computed using Bayes rule:

$$\begin{aligned} p_{I|X;\theta'}(i|x) &= \frac{f_{X|I;\theta'}(x|i)p'_I(i)}{f_{X;\theta'}(x)} \\ &= \frac{f_{X|I;\theta'}(x|i)p'_I(i)}{\sum_{i\in\mathcal{I}} f_{X|I,\theta'}(x|i)p'_I(i)} \\ &= \frac{f_{X;\theta'_i}(x)p'_I(i)}{\sum_{i\in\mathcal{I}} f_{X;\theta'_i}(x)p'_I(i)}. \end{aligned} \tag{4.34}$$

Example 4.4 shows how the maximization of the cost function of equation 4.33 can be used to perform training for the Gaussian mixture model.

Example 4.4: EM algorithm for a Gaussian mixture model

In this example we find the equations that allow us to compute the EM estimate of a Gaussian mixture model on a computer. For clarity we consider explicitly the case of k-dimensional vectors.

Additive terms in the cost function of equation 4.33 that involve disjoint parameter sets can be optimized separately. We first optimize the individual θ_i; the relevant part of the criterion is

$$\eta_i^{(f)} = \sum_{n=1}^{N} p_{I|X^k;\theta'}(i|x^k_{(n)}) \log(f_{X^k;\theta_i}(x^k_{(n)})).$$

Straightforward optimization of this criterion is not difficult and similar to the optimization found in example 4.2. In fact, we can exploit the optimization of example 4.2 by recognizing that the only difference is that we have a weighting of the individual additive terms of the log likelihood by $p_{I|X^k;\theta'}(i|x^k_{(n)})$, which varies with each observation n rather than being constant. It is then easily seen that the

solution is

$$\begin{aligned}\mu_{i,\text{opt}}^k &= \frac{\sum_{n=1}^N p_{I|X^k;\theta'}(i|x_{(n)}^k)x_{(n)}^k}{\sum_{n=1}^N p_{I|X^k;\theta'}(i|x_{(n)}^k)},\\ R_{i,\text{opt}} &= \frac{\sum_{n=1}^N p_{I|X^k;\theta'}(i|x_{(n)}^k)(x_{(n)}^k-\mu_{i,\theta'}^k)(x_{(n)}^k-\mu_{i,\theta'}^k)^T}{\sum_{n=1}^N p_{I|X^k;\theta'}(i|x_{(n)}^k)},\end{aligned}$$

where we use equation 4.34 to compute the $p_{I|X^k;\theta'}(i|x_{(n)}^k)$.

To find the optimal solution for the mixture component probabilities $\{p_I(i)\}_{i\in\mathcal{I}}$ we consider the terms

$$\eta^{(p)} = \sum_{i\in\mathcal{I}}\sum_{n=1}^N p_{I|X^k;\theta'}(i|x_{(n)}^k)\log(p_I(i))$$

of the criterion $Q_{\text{mixture}}(\theta;\theta')$. The probabilities $\{p_I(i)\}_{i\in\mathcal{I}}$ must be nonnegative and should add to unity:

$$\sum_{i\in\mathcal{I}} p_I(i) = 1.$$

Using the method of Lagrange multipliers, we find as solution

$$p_{I,\text{opt}}(i) = \frac{1}{N}\sum_{n=1}^N p_{I|X^k;\theta'}(i|x_{(n)}^k).$$

The EM algorithm can be implemented by performing the following operations at each iteration:

1. Compute $p_{I|X^k;\theta'}(i|x_{(n)}^k)$ using equation 4.34.
2. Compute θ_{opt}. That is, compute for all i the parameters $p_{I;\text{opt}}(i)$, $\mu_{i,\text{opt}}^k$, and $R_{i,\text{opt}}$ using the equations provided in this example.
3. Replace θ' with θ_{opt}.

We continue these iterations until the criterion is deemed sufficiently small. The algorithm can be initialized with a random set of means and identical covariances for all components.

4.4 Maximum-Entropy Probability Distributions

In section 4.3, we discussed methods to estimate probability distributions from a known data sequence. In this section, we determine probability distributions for the case that we only know a set of constraints on the probability distribution. In addition to the constraints that the distribution has to integrate to one and be nonnegative additional constraints, such as a given set of autocorrelations values, may or may not be present. The probability distribution that we seek is the distribution that is the most likely empirical distribution under the assumption that all sequences that are possible, given the constraints, are equally likely. This distribution is called the maximum-entropy distribution. Maximum-entropy distributions are often useful and, thus, commonly computed.

As we will see in chapter 6, at a given distortion, high uncertainty leads to a higher bit rate. It is thus natural to associate a maximum-entropy distribution with an upper bound on rate given the constraints.

Maximum-entropy distributions are also useful outside the realm of source coding. A good example of that is the maximum-entropy spectral estimation method, which was originally introduced by Burg in 1967 (for a complete description see [18])[3].

We start this section with a more in-depth analysis of the meaning of the maximum-entropy criterion. We then show that the maximum-entropy criterion with autocorrelation constraints results in Gaussian densities and also discuss spectral estimation based on the maximum-entropy criterion.

4.4.1 The Interpretation of the Maximum-Entropy Criterion

The maximum-entropy method provides an estimate of the probability distribution of a random variable. Let us consider an iid sequence of k of these random variables, where k is very large. The random variable has a discrete alphabet. The basic tenet of the maximum-entropy method is that each unique sequence has the same probability of occurrence. This assumption is identical to the assumption that all accessible ensemble configurations in section 2.2 are equally likely. Under this assumption, what is a reasonable estimate of the probability distribution for the symbols in an arbitrary sequence? For large k, it is reasonable to use the empirical distribution that is most likely to be observed when selecting an arbitrary sequence.

We now show that the most likely empirical distribution corresponds to the maximum-entropy probability distribution. For notational convenience, let us consider the discrete alphabet $\mathcal{A} = \{1, 2, \cdots, |\mathcal{A}|\}$ and a k-symbol sequence where k_1 symbols are "1", k_2 symbols are "2", etc. We can use the reasoning used in equation 2.2 of section 2.1. The logarithm of the number of sequences normalized by the number of samples is

$$\begin{aligned} \frac{1}{k}\log(\Omega_{\text{seq}}) &= \frac{1}{k}\log\left(\frac{k!}{k_1!\cdots k_{|\mathcal{A}|}!}\right) \\ &\approx -\sum_{x\in\mathcal{A}} p_X(x)\log(p_X(x)) \\ &= H(X), \end{aligned} \tag{4.35}$$

where X is a random variable with the probability mass function $p_X(x) = k_x/k$. Thus, selecting the empirical probability mass function with the highest probability is the same as selecting the probability mass function with maximum entropy. Moreover, when the sequence is very long, almost all sequences have a probability mass function corresponding to the maximum-entropy probability mass function.

We assume that the above reasoning remains valid when we move from a discrete alphabet to a continuous alphabet. This implies that we use a fine-grained, regular quantization when presupposing an equal probability density for the symbol sequences. The maximum-entropy method then leads to the maximization of the differential entropy to obtain a probability density function.

[3] In spectral estimation, the appropriateness of the criterion is not without controversy (e.g., [19]).

In the above interpretation of the maximum-entropy criterion, we have used a starting point that is exactly the complement of the one used in section 2.8 to derive the equipartition principle. To find an interpretation of the maximum-entropy criterion in the present section, we started with the notion that all sequences are equally likely, and we then found that the empirical distribution that is likely to be observed is the maximum-entropy distribution. Conversely, in our derivation of the equipartition principle in section 2.8, we assumed that the probability distribution exists, and we then found that for large cardinality of $\mathcal{I}$, almost all sequences have equal probability.

4.4.2 Maximum-Entropy Distributions

In the following, we will find a formal description of maximum-entropy distributions for continuous alphabets. The discrete-alphabet case is very similar (see problem 6). In the maximum-entropy method we try to find the probability density $f_{X^k}(x^k)$ that maximizes

$$h(X^k) = -\int_{\mathcal{A}} f_{X^k}(x^k)\log(f_{X^k}(x^k))dx = -\mathrm{E}[\log(f_{X^k}(X^k))], \tag{4.36}$$

where $\mathcal{A}$ is the range of support of $f_{X^k}(x^k)$, with the constraint that $f_{X^k}(x^k)$ is a density, that is that

$$1 = \int_{\mathcal{A}} f_{X^k}(x^k)dx^k, \tag{4.37}$$

and that $f_{X^k}(x^k)$ is nonnegative everywhere. We can have additional set, $\mathcal{B}$, of constraints of the form

$$\mathrm{E}[g_i(x^k)] = G_i, \;\; i \in \mathcal{B}, \tag{4.38}$$

where $g_i(\cdot)$ is some function and G_i is a constant. The extended criterion to be optimized is thus

$$\eta = \int_{\mathcal{A}} f_{X^k}(x^k)\log(f_{X^k}(x^k))dx + \mu\int_{\mathcal{A}} f_{X^k}(x^k)dx^k + \sum_{i\in\mathcal{B}}\lambda_i\int_{\mathcal{A}} g_i(x^k)f_{X^k}(x^k)dx^k, \tag{4.39}$$

where μ and the λ_i are Lagrange multipliers. The Euler-Lagrange equation is

$$0 = \log(f_{X^k}(x^k)) + 1 + \mu + \sum_{i\in\mathcal{B}}\lambda_i g_i(x^k) \tag{4.40}$$

and we have as formal solution

$$f_{X^k}(x^k) = \beta \mathrm{e}^{-\sum_{i\in\mathcal{B}}\lambda_i g_i(x^k)}, \;\; x^k \in \mathcal{A}, \tag{4.41}$$

where β and λ_i are selected to satisfy the constraints.

A particular, important case is the maximum-entropy distribution for a zero-mean vector with known variance. The constraint $\mathrm{E}[g_1(x^k)] = G_1$ is then specified by $g_1(x^k) = \sum_i x_i^2$, which means that the formal solution takes the form of a k-dimensional multivariate Gaussian with diagonal covariance matrix. To emphasize the importance of this result, we write it as a theorem:

Theorem 14 *The Gaussian density*

$$f_{X^k}(x^k) = \frac{1}{\sqrt{(2\pi)^k \sigma^{2k}}} \mathrm{e}^{-\frac{1}{2\sigma^2}\sum_{i=1}^{k} x_i^2}$$

maximizes the differential entropy for a vector with a sum of component variances equal to $k\sigma^2$.

In many cases it is not possible to construct a proper probability density with the solution of the form given in equation 4.41. A trick that can be used to find a proper solution is to *i*) add additional constraints that renders equation 4.41 of a form that facilitates simple analytic descriptions of the probability densities, and *ii*) maximize the values of the additional constraints so as to maximize the (differential) entropy, for proper probability densities. We use this trick in section 4.4.3.

Example 4.5: Finite-range variable with maximum differential entropy
Consider the set of densities $f_X(x)$ that are constrained to be zero outside the interval $\mathcal{A} = [0, a)$. We want to find the density with maximum differential entropy. In this problem the only constraint is that $f_X(x)$ is a density. Thus, equation 4.41 reduces to

$$f_X(x) = \beta$$

in the range of support. Together with the constraint this gives as solution $f_X(x) = 1/a,\ x \in [0, a)$. Since $f_X(x)$ is nonnegative everywhere, $f_X(x)$ is a density. The solution is very similar to that obtained for the discrete case discussed in section 2.3. It is intuitive that the uncertainty about the value that X will take is maximized by this density. The corresponding differential entropy is given in example 3.1.

Example 4.6: Density with known absolute value expectation
We want to find the maximum-entropy density for the case where we know that $\mathrm{E}[|x|] = b$, where $b \in [0, \infty)$. Equation 4.41 tells us that

$$f_X(x) = \beta \mathrm{e}^{-\lambda_1 |x|}.$$

The constraints are

$$\begin{aligned} 1 &= \int_{-\infty}^{\infty} \beta \mathrm{e}^{-\lambda_1 |x|} dx = 2 \int_{0}^{\infty} \beta \mathrm{e}^{-\lambda_1 x} dx, \\ b &= \int_{-\infty}^{\infty} |x| \beta \mathrm{e}^{-\lambda_1 |x|} dx = 2 \int_{0}^{\infty} x \beta \mathrm{e}^{-\lambda_1 x} dx. \end{aligned}$$

These constraints are satisfied by

$$f_X(x) = \frac{1}{2b} e^{\frac{-|x|}{b}}.$$

which is a density, since $f_X(x)$ is nonnegative for all x on the real line.

4.4.3 Maximum-Entropy Spectral Estimation

In maximum-entropy spectral estimation, it is assumed that a set of moments of a signal is known[4]. Based on this set of constraints, we can find a joint probability

[4] In practical spectral estimation, the moments are estimated, rendering this assumption inaccurate.

density that maximizes the joint entropy of the samples. We will see that this joint probability density function is the multi-variate Gaussian density. When we extend this reasoning to the case of stationary processes, we find that, given a set of second-order moments (i.e., part of the covariance matrix) the maximum-entropy process must be auto-regressive.

To start, let us consider a vector X^k in $\mathbb{R}^k$ for which we know for a set $\mathcal{B}$ of indices i and j the numbers $\mathrm{E}[X_i X_j] = R_{ij}$. What is the maximum-entropy multi-variate density of X^k under these constraints? The solution method of section 4.4.2 readily generalizes to multiple dimensions. Thus, we see from equation 4.41 that the multi-variate density will be of the form

$$f_{X^k}(x^k) = \beta \mathrm{e}^{-\sum_{i,j\in\mathcal{B}} \lambda_{ij} x_i x_j}. \tag{4.42}$$

It is often not straightforward to find a solution of this form. Furthermore, the solution may often not be a proper density. Thus, we apply the trick mentioned after equation 4.41 and add "undetermined" constraints. We simply extend $\mathcal{B}$ to include all i and j combinations such that $i \in \{1, \cdots, k\}$ and $j \in \{1, \cdots, k\}$. The probability density can then be written as

$$f_{X^k}(x^k) = \beta \mathrm{e}^{-x^{kT} \Lambda x^k}, \tag{4.43}$$

where $x^k = [x_1, \cdots, x_k]^T$ and Λ is a matrix with elements λ_{ij}. Equation 4.43 is of the form of the multi-variate Gaussian distribution, so we immediately conclude that the probability density function is

$$f_{X^k}(x^k) = \frac{1}{\sqrt{(2\pi)^k \det(R)}} \mathrm{e}^{-\frac{1}{2} x^{kT} R^{-1} x^k}, \tag{4.44}$$

where R is the covariance matrix with elements R_{ij} and where $\det(R)$ is the determinant of this matrix. All that remains is to replace the unknown values of the matrix R_{ij} with those that maximize the entropy. From equation 3.25, we see that this is equivalent to finding the values that **maximize the determinant of the covariance matrix**.

Example 4.7: Two-dimensional vector with known variances
We have a two-dimensional random vector with known R_{11} and R_{22}. In this case, equation 4.42 gives a valid probability density. It is of the form

$$f_{X^2}(x^2) = \beta \exp(-\sum_{i=1}^{i=2} \lambda_i x_i^2),$$

which results in the solution

$$f_{X^2}(x^2) = \frac{1}{\sqrt{\prod_{i=1}^{i=2} (2\pi R_{ii})}} \exp(-\sum_{i=1}^{i=2} \frac{x_i^2}{2R_{ii}}).$$

We can also start from equation 4.44. The elements $R_{12} = R_{21}$ of R are not known. We want to find the value for R_{12} that maximizes the diffential entropy, or, by theorem 12, the determinant of the covariance matrix:

$$\begin{aligned} h(X^2) &= -\int f_{x^2}(x^2) \log(f_{X^2}(x^2)) dx^2 \\ &= \frac{1}{2} \log((2\pi \mathrm{e})^2 \det(R)) \\ &= \frac{1}{2} \log((2\pi \mathrm{e})^2 (R_{11} R_{22} - R_{12}^2)) \end{aligned}$$

where we used equation 3.25. In other words, we want to minimize R_{12}^2. For real random vectors this means that $R_{12} = 0$ and equation 4.44 reduces to the result obtained earlier in this example.

In the case of spectral estimation, the finite-dimensional vector is replaced with a stationary Gaussian process. We maintain the procedure, but replace the expression for the differential entropy according to Szegös theorem (theorem 13). Thus, we now maximize the differential entropy rate of a Gaussian process with memory

$$h_\infty(X_i) = \frac{1}{2}\log(2\pi\mathrm{e}) + \frac{1}{4\pi}\int_{-\pi}^{\pi}\log(R(\mathrm{e}^{j\omega}))d\omega \tag{4.45}$$

under constraints of the form

$$\begin{aligned} R_i &= \mathrm{E}[X_n X_{n+i}] \\ &= \frac{1}{2\pi}\int_{-\pi}^{\pi} R(\mathrm{e}^{j\omega})e^{j\omega i}d\omega, \end{aligned} \tag{4.46}$$

where we introduced the simplified notation $R_i \equiv R_{n,n+i}$. In the following, we will assume that we know R_i for $i \in \{-p, \cdots, 0, \cdots, p\}$ and, naturally, that $R_i = R_{-i}$. The extended criterion can be written as

$$\eta = \frac{1}{4\pi}\int_{-\pi}^{\pi}\log(R(\mathrm{e}^{j\omega}))d\omega - \sum_{i=-p}^{p}\lambda_i\frac{1}{2}\frac{1}{2\pi}\int_{-\pi}^{\pi}R(\mathrm{e}^{j\omega})e^{j\omega i}d\omega, \tag{4.47}$$

where we included an additional factor $-\frac{1}{2}$ to simplify notation in the following derivation. The Euler-Lagrange equations are

$$0 = \frac{1}{R(\mathrm{e}^{j\omega})} - \sum_{i=-p}^{p}\lambda_i e^{j\omega i}. \tag{4.48}$$

The maximum-entropy power density is then of the form

$$R(\mathrm{e}^{j\omega}) = \frac{1}{\sum_{i=-p}^{p}\lambda_i e^{j\omega i}}. \tag{4.49}$$

We note that a symmetry in i is required for λ_i if $R(\mathrm{e}^{j\omega})$ is to be real for all values of ω. Let us consider then the following factorization for the denominator of equation 4.49:

$$\sum_{i=-p}^{p}\lambda_i e^{j\omega i} = (\sum_{i=0}^{p}\mu_i e^{j\omega i})(\sum_{i=0}^{p}\mu_i e^{j\omega i})^* = |\sum_{i=0}^{p}\mu_i e^{j\omega i}|^2. \tag{4.50}$$

where * indicates complex conjugate. Thus, equation 4.49 can be written as

$$R(\mathrm{e}^{j\omega}) = \frac{1}{|\sum_{i=0}^{p}\mu_i e^{j\omega i}|^2}. \tag{4.51}$$

Equation 4.51 shows that the maximum-entropy spectral estimate is the all-pole spectrum with p poles with the p specified autocorrelations. This motivation is often used to justify spectral estimation with all-pole models. However, it should be remembered that in practical cases the moments are generally approximations.

Example 4.8: MA1 process has lower entropy rate than AR1 process
In this example, we directly compare the differential entropy rate of a stationary first-order moving average (MA1) process and that of a stationary first-order autoregressive (AR1) process, all for equal R_0 and R_1. These processes can be created by filtering. From the theoretical results above, we expect the MA1 process to have a lower entropy rate than the AR1 process, and we will confirm that. We first compute explicit expressions for the entropy rates of both processes in terms of R_0 and R_1.

We start with the MA1 process. We generate a signal X_i by filtering a stationary Gaussian iid process E_i with variance σ_E^2 with the MA1 filter $H(z) = 1 + az^{-1}$. The signal X_i is Gaussian (we can use the same derivation as in example 3.6 to prove this) with a variance

$$R_0 = \mathrm{E}[X_i^2] = \mathrm{E}[(E_i + aE_{i-1})(E_i + aE_{i-1})] = (1 + a^2)\sigma_E^2.$$

We also have

$$R_1 = \mathrm{E}[X_i X_{i-1}] = \mathrm{E}[(E_i + aE_{i-1})(E_{i-1} + aE_{i-2})] = a\sigma_E^2.$$

Solving for σ_E^2 in terms of R_0 and R_1 we obtain

$$\sigma_E^2 = \frac{1}{2}R_0 \pm \frac{1}{2}\sqrt{R_0^2 - 4R_1}.$$

Using the reasoning for obtaining the entropy rate used in example 3.6, we obtain for the present MA filter

$$\begin{aligned} h_\infty(X_i) &= h(E_i) \\ &= \frac{1}{2}\log(2\pi\mathrm{e}\sigma_E^2) \\ &= \frac{1}{2}\log(\pi\mathrm{e}(R_0 \pm \sqrt{R_0^2 - 4R_1})). \end{aligned}$$

Next, we determine the entropy rate of the AR1 process. The excitation E_i and a signal X_i are now related by

$$E_i = X_i - bX_{i-1}.$$

Again the filter output is Gaussian because the input is Gaussian. Because of independence, we see that

$$0 = \mathrm{E}[E_i X_{i-1}] = R_1 - bR_0$$

and, thus, obtain that $b = R_1/R_0$. Furthermore,

$$\begin{aligned} \sigma_E^2 &= \mathrm{E}[X_i^2 - 2bX_iX_{i-1} + b^2X_{i-1}^2] \\ &= R_0 - 2bR_1 + b^2R_0 \\ &= R_0 - \frac{R_1^2}{R_0}. \end{aligned}$$

The differential entropy of the AR1 process is simpler to evaluate than that of the MA process:

$$\begin{aligned} h_\infty(X_i) &= h(X_i|X_{i-1}, X_{i-2}, \cdots) \\ &= h(X_i|X_{i-1}) \\ &= h(E_i + bX_{i-1}|X_{i-1}) \\ &= h(E_i) \\ &= \log(2\pi\mathrm{e}\sigma_E^2) \\ &= \log(2\pi\mathrm{e}(R_0 - \frac{R_1^2}{R_0})). \end{aligned}$$

We now compare the results for the MA1 and the AR1 signals directly. First of all, we see that all we have to compare is the variance of the excitation signal, since in both cases the differential entropy rate of the process is $\log(2\pi e\sigma_E^2)$ where σ_E^2 is the excitation signal variance.

We proceed as follows: we start with an MA1 process with an excitation with unity variance and evaluate R_0 and R_1. We then compute the variance of the excitation of an AR1 process with the same R_0 and R_1. (Since we can scale the signals, we have not lost any generality.) If the AR1 excitation variance is larger than unity, then the entropy rate of the AR1 process is larger. We consider the MA1 process $X_i = E_i + aE_{i-1}$. We earlier showed that then $R_0 = 1 + a^2$ and $R_1 = a$. The variance of an AR1 model is then

$$\begin{aligned} \sigma_E^2 &= R_0 - \frac{R_1^2}{R_0} \\ &= 1 + a^2 - \frac{a^2}{1 + a^2} \\ &\geq 1. \end{aligned}$$

Herewith, we have shown that, for given R_0 and R_1, the variance and thus the entropy rate of an AR1 process is larger than that of an MA1 process. This confirms the general results.

4.5 Problems

1. Consider an alphabet $\{1, 2, 3\}$ and a sequence 2311.

   (a) Using the maximum-likelihood method, compute the empirical probability-mass function.

   (b) Using the Laplace's rule, compute the empirical probability-mass function.

2. Consider random sequences of bits with probability P(X=0)=0.3, P(X=1)=0.7. Using a random number generator, create two such sequences of 10000 symbols.

   (a) Using the maximum-likelihood method, compute the empirical probability-mass function, for k = 0, 10, 100, 1000, 10000 using the observed values of k_0.

   (b) Using the Laplace's rule, compute the empirical probability-mass function, for k = 0, 10, 100, 1000, 10000 using the observed values of k_0.

3. A variable has an alphabet with four symbols of probability 0.3, 0.3, 0.2, and 0.2 respectively.

   (a) Give a tight lower bound for the bit rate to encode a sequence of such variables.

   (b) What is the lower bound on the rate for a code if the symbols had a uniform probability mass function.

   (c) You estimate the probability mass function from data using the maximum-likelihood estimate. Estimate the variance in the symbol probabilities after 10, 100, 1000, and 10000 symbols.

4. Consider a random variable with a normal density of unknown mean μ and known variance σ^2. The variable is sampled k times with outcomes $\{x_1, \cdots, x_k\}$.

   (a) Find an expression for the maximum-likelihood estimate for the mean μ that is recursive in the sampling index i.

   (b) Assume that a first estimate of the mean itself is a normal density with zero mean and variance γ^2. Find the MAP estimate for μ and the a-posteriori density.

   (c) Generate data of a normal density with $\mu = 1$ and $\sigma^2 = 9$. Assuming that σ^2 is known, plot your maximum-likelihood and MAP estimates of μ as a function of k.

5. Consider a Gaussian random vector with covariance matrix

$$R = \begin{bmatrix} 1 & 1/4 & 1/8 \\ 1/4 & 1 & b \\ 1/8 & a & 1 \end{bmatrix}.$$

   (a) Compute the differential entropy of the vector for $a = b = 0$.

   (b) Find the values for a and b that maximize the entropy of the vector while making sure that the matrix is a covariance matrix. Provide the resulting entropy.

   (c) Prove explicitly that your optimized a and b result in a proper covariance matrix.

   (d) Find a linear transform of the process vector of maximum entropy that renders the vector components uncorrelated.

   (e) Plot a high-level schematic of an efficient coding and decoding system of the maximum-entropy vector process that uses only scalar quantizers.

6. We consider the maximum-entropy distribution for the discrete case.

   (a) Find the equivalent of equation 4.41 for the general discrete alphabet case.

   (b) Find the maximum-entropy probability mass function for a variable with a discrete alphabet $\mathcal{A} = \{2, 3, 4, 5\}$ under the constraint that $\mathrm{E}[X] = 4$.

   (c) For the same alphabet, find the maximum-entropy probability mass function under the constraint that $\mathrm{E}[X^2] = 8$.

7. You code a variable with a continuous density with a uniform quantizer. The quantizer step size is sufficiently small and the density sufficiently smooth that the density does not change significantly within the quantizer cells. You apply a lossless coder to the quantizer indices. You should use a true probability density $f_X(x)$ but you only have available an estimate $\hat{f}_X(x)$.

   (a) Write the increase in the coding rate resulting from having to use the estimated rather than the true density in terms of the true and estimated densities.

   (b) The density $f_X(x)$ is a zero-mean Gaussian density with unity variance. Find the maximum-likelihood fit of a Laplacian density, $\hat{f}_X(x)$, to this Gaussian density and evaluate the increase in coding rate resulting from using the mismatched density.

8. Let X^2 (the superscript indicates dimensionality) be a two-dimensional vector with independent components each with a distribution that is uniform on $[0, 1.0)$. We consider the random vector $Y^2 = BX^2$, where

$$B = \begin{bmatrix} 1 & 0.2 \\ 1 & -0.2 \end{bmatrix}.$$

The vector has a density f_{Y^2} and the density model is denoted as $f_{Y^2;\theta}$.

   (a) Find the "maximum-likelihood" multi-variate Gaussian density model for Y^2. Estimate the relative entropy $h(f_{Y^2} \| f_{Y^2;\theta})$ in bits by Monte Carlo integration. (Hint: note that $h(Y^2)$ can be computed analytically.)

   (b) Generate 10000 realizations of the random vector Y^2. Plot a scatter plot.

   (c) Train a Gaussian mixture density model with two components using the EM algorithm. Estimate the relative entropy $h(f_{Y^2} \| f_{Y^2;\theta})$ in bits.

   (d) Train a Gaussian mixture density model with two components with *diagonal* covariance matrices. Estimate the relative entropy $h(f_{Y^2} \| f_{Y^2;\theta})$ in bits.

   (e) Train a Gaussian mixture density model with four components with *diagonal* covariance matrices. Estimate the relative entropy $h(f_{Y^2} \| f_{Y^2;\theta})$ in bits.

9. Consider a data base of one minute of speech recorded at 8 kHz. Divide the signal into block of 80 samples and measure the energy of these segments on a per sample basis. Consider the measured energies to be realizations of the random variable X.

   (a) Consider a mixture model of Laplacian densities. Determine the EM algorithm to train this mixture for a given data base.

   (b) Using the EM algorithm, find the one-component, two-component, and four-component Laplacian density models to the describe the energy distribution of speech. Plot the model distributions.

   (c) Using the EM algorithm, find the one-component, two-component, and four-component Gaussian density models to the describe the energy distribution of speech. Plot the model distributions.

   (d) Estimate $\int f_X(x) \log(f_{\hat{X}}(x)) dx$ for all your distributions and interpret the results.

10. In this problem we consider different approaches to the probability-mass estimation and coding for a discrete variable X with alphabet $\{-2, 0, 2\}$.

   (a) Find the maximum-entropy distribution under the constraint that $\mathrm{E}[X^2] = 2$.

   (b) You have four (4) independent observations of X: $\{-2, -2, 0, 2\}$. Given these observations, find the maximum-likelihood distribution for X.

   (c) The prior distribution is $P(\{p_X(-2) = 0.33, p_X(2) = 0.33, p_X(0) = 0.33\}) = 0.5$, $P(\{p_X(-2) = 0.5, p_X(2) = 0.25, p_X(0) = 0.25\}) = 0.5$, Compute the maximum a posteriori estimate given the observations $\{-2, -2, 0, 2\}$.

   (d) Explain when and why the maximum-a-posteriori estimate converges to the maximum-likelihood estimate with increasing number of data.

   (e) Find the penalty in bit rate per sample if you apply (optimal) lossless coding to the data using as distribution estimate $\hat{p}_X(-2) = \hat{p}_x(2) = \hat{p}_X(0) = 0.33$ but the true distribution is $p_X(-2) = 0.5, p_x(2) = 0.25$, $p_X(0) = 0.25$.

5

Lossless Coding

5.1 Introduction

In chapter 2, we discussed the relation between the entropy of a random variable with a discrete alphabet and the mean length of the source code. Using a Shannon code, it was shown to be possible to define a uniquely decodable code for which the mean codeword length in bits is within one bit of the entropy of the random variable. Since the reconstruction of the discrete-alphabet variable is exact, a Shannon code is a particular **lossless code**. In general, lossless codes exploit the probability mass function of the symbols by assigning longer codewords to symbols or sequences of symbols of lower probability. In this chapter, we describe a number of common lossless coding procedures.

It is natural to use lossless coding when storing or transmitting sequences of symbols, where the data must be preserved, such as a text file. However, lossless coding is also used in conjunction with lossy coding as is illustrated in example 5.1.

> **Example 5.1: Lossless coding in MPEG-2 audio coder**
> Lossless coding is featured in the MPEG-2 audio coding scheme [1]. In this coder, a filter bank is used to map the one-dimensional audio signal into a set of signals with lower sampling rate. Each of these down-sampled signals is subjected to an adaptive scalar quantizer that accounts for perceptual effects. The quantization indices produced by the encoder form a discrete alphabet that is not uniformly distributed. A lossless coder is then used to describe the quantization indices at a low average bit rate. This lossless coder exploits only the marginal probability density functions.

Lossless codes can be divided into two classes: *i*) codes that require an a-priori probability distribution and *ii*) universal codes that do not require an a-priori probability distribution. Within the universal codes one often distinguishes the subclass of adaptive codes. Adaptive codes are identical to codes that require an a-priori probability distribution except that they are augmented with the computation of the corresponding empirical probability-distribution. We start the chapter with codes that require a-priori knowledge of the source-symbol statistics, and then discuss universal codes.

5.2 Codes Using Known Source-Symbol Statistics

In chapter 2, we defined a prefix- or instantaneous code as a code in which a codeword cannot be a prefix to another codeword. It was also shown that, for any set of codeword lengths satisfying the Kraft inequality, a prefix code can be found. In other words, no optimal code is better than the best prefix code, which is thus also optimal. In this section, we will discuss two prefix codes: Shannon codes and Huffman codes.

5.2.1 Shannon Codes

Shannon codes can be motivated by a minimization of the average codeword length without taking into account that the final codewords must be an integer number of symbols (usually bits). That is, we start with designing a code that minimizes the average codeword length

$$L = E[l(X)] = \sum_{x \in \mathcal{A}} p_X(x) l(x) \tag{5.1}$$

under the constraint that the Kraft inequality is satisfied. If a noninteger number of bits is allowed, we can always create a shorter unique code by shortening an arbitrary codeword if the Kraft inequality is a true inequality. Thus, for this "noninteger" case the Kraft inequality is an equality for an optimal code,

$$\sum_{x \in \mathcal{A}} 2^{-l(x)} = 1. \tag{5.2}$$

If we allow noninteger bit assignments, a simple constrained optimization problem results. Using the Lagrange multiplier μ, the criterion to be minimized becomes

$$\eta = \sum_{x \in \mathcal{A}} p_X(x) l(x) + \mu(\sum_{x \in \mathcal{A}} 2^{-l(x)} - 1). \tag{5.3}$$

Differentiating η with respect to $l(x)$ and setting the result equal to zero, we obtain a set of equations of the following form:

$$p_X(x) - \mu' 2^{-l(x)} = 0, \tag{5.4}$$

where $\mu' = \log(2)\mu$. Summing over the alphabet, $\mathcal{A}$, results in

$$1 - \mu' \sum_{x \in \mathcal{A}} 2^{-l(x)} = 0 \tag{5.5}$$

and comparing this with 5.2 shows that $\mu' = 1$. We find that, if the bit allocations are allowed to be noninteger, the optimal codeword lengths are

$$l(x) = -\log_2(p_X(x)). \tag{5.6}$$

Naturally, we cannot use fractional bit assignments in reality. To obtain a practical implementation, we round the bit assignment up and obtain

$$l(x) = \lceil -\log_2(p_X(x)) \rceil. \tag{5.7}$$

As was shown in section 2.4.2, a simple tree structure can be used to construct an actual code and such a code is called a Shannon code. A Shannon code was used in section 2.4.2 to obtain an upper bound on the difference between the mean codeword length and the entropy.

A Shannon code is optimal when the codewords lengths resulting from equation 5.6 are integer. However, in many cases Shannon codes are not optimal, as is illustrated in example 5.2. For this reason, Shannon codes are not commonly used. To ensure optimality Huffman codes are used instead; they form the subject of the next subsection.

Example 5.2: Nonoptimality of Shannon codes
Consider a variable with alphabet $\{a, b\}$ and probability mass function $p_X(a) = 0.01$ and $p_X(b) = 0.99$. What is the codeword length of the Shannon code? The solution is simple: $l(a) = \lceil -\log_2(0.01) \rceil = 7$ and $l(b) = \lceil -\log_2(0.99) \rceil = 1$. However, the optimal code obviously has a codeword length of one bit for both symbols, illustrating the nonoptimality of a Shannon code.

5.2.2 Huffman Codes

For finite alphabets for which the probability mass function of the symbols is known, a Huffman code [20] is often a good solution. Huffman codes are commonly used in practical applications. A Huffman code is designed with a simple iterative procedure where at each iteration step one

1. selects the two source symbols of the alphabet with lowest probability,
2. assigns them codewords with the same unknown prefix and 1 and 0 as their last bits, respectively, and
3. combines them into one source symbol, which has the unknown prefix as codeword.

This procedure is repeated until the remaining alphabet has only one entry.

Let us attempt to write the Huffman design procedure in a somewhat more formal manner. We consider a random variable X, with symbol set (alphabet) $\mathcal{A}$, and probability mass function $p_X(x)$. The binary codeword for a symbol x is denoted as $\mathsf{c}(x)$ (the notation $\mathsf{c}(x)1$ denotes the codeword $\mathsf{c}(x)$ with the symbol "1" appended). Furthermore, the notation $\mathcal{A} \cup \mathcal{B}$ indicates the union of the set $\mathcal{A}$ and the set $\mathcal{B}$ and $\mathcal{A} - \mathcal{B}$ indicates the set $\mathcal{A}$ without its subset $\mathcal{B}$. Thus, if $w \in \mathcal{A}$, then $\mathcal{A} - \{w\}$ is the set $\mathcal{A}$ with the symbol w removed. Using this notation, the formal Huffman procedure is given in table 5.1.

Example 5.3: Design of Huffman code
We design a Huffman code for an alphabet with probability mass function $\{0.1, 0.2, 0.3, 0.4\}$. The design is most easily illustrated graphically as is done in figure 5.1. The codewords are 011, 010, 00 and 1.

It is not possible to design a prefix code shorter in length[1] than the Huffman code. Since we have already shown in section 2.4.2 that the best prefix codes are optimal, this

[1] We use "code length" to refer to $L = \mathrm{E}[l(x)]$, the average codeword length of a code.

Table 5.1: The Huffman code design algorithm.

1. set $p^{(0)}(x) = p_X(x), \ \forall_{x\in\mathcal{A}}$
   set $\mathcal{A}^{(0)} = \mathcal{A}$

2. select the two symbols from $\mathcal{A}^{(m)}$ with the lowest probability:
   $y = \underset{x\in\mathcal{A}^{(m)}}{\operatorname{argmin}}\, p^{(m)}(x)$
   $z = \underset{x\in\mathcal{A}^{(m)}-\{y\}}{\operatorname{argmin}}\, p^{(m)}(x)$

3. define a new alphabet $\mathcal{A}^{(m+1)} = (\mathcal{A}^{(m)} - \{y, z\}) \cup \{w\}$
   with probability mass function
   $p^{(m+1)}(w) = p^{(m)}(y) + p^{(m)}(z)$
   $p^{(m+1)}(x) = p^{(m)}(x), \ x \in \mathcal{A}^{(m+1)} - \{w\}$.
   and with the following relation between the codewords:
   $\mathsf{c}(y) = \mathsf{c}(w)1$
   $\mathsf{c}(z) = \mathsf{c}(w)0$

4. if $\mathcal{A}^{(m+1)}$ has more than one entry, $m \to m+1$ and go to 2

5. assign a zero-length codeword to the single-alphabet entry;
   trace back to find the codewords for $\mathcal{A}$

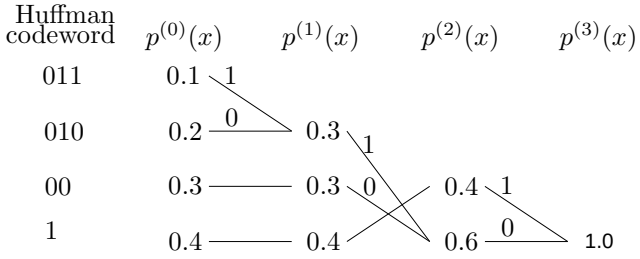

Figure 5.1: The iterative procedure to obtain a Huffman code for example 5.3.

implies that the Huffman code is optimal:

Theorem 15 *The Huffman code is a uniquely decodable code with the shortest average codeword length for a finite-alphabet discrete variable with known probability mass function.*

What remains to be shown to prove this theorem is that the Huffman code is an optimal prefix code. First, we note that in an optimal binary prefix code for a random variable X, $p_X(a) < p_X(b)$ implies that $l(a) \geq l(b)$. This follows immediately by contradiction. If it were true that $l(a) < l(b)$, then we could exchange the codewords and obtain a shorter code. Second, we note that in an optimal binary prefix code, the two symbols with the lowest probability have identical codeword length, and that there exists such a code for which these codewords differ only in their last bit. This can also be proven by contradiction. Assume that there is a codeword that is at least one bit longer than any other codeword. Since a prefix of a codeword in a prefix code cannot itself be a

codeword, one can remove at least one bit from this codeword. Furthermore, it is always possible to make the symbol share all but the last bits with the symbol of next lowest probability.

We now can argue why the Huffman code is an optimal prefix code. Consider the alphabet $\mathcal{A}^{(m)}$ with code $\mathsf{C}^{(m)}$. Combining the two symbols of lowest probability from an alphabet, y and z, into a new symbol w results in a new alphabet $\mathcal{A}^{(m+1)}$,

$$\mathcal{A}^{(m+1)} = (\mathcal{A}^{(m)} - \{y, z\}) \cup \{w\}. \tag{5.8}$$

We assume that the code $\mathsf{C}^{(m+1)}$ is obtained by using the shared prefix of the codewords for x and y as the codeword for the new symbol w. We have

$$\begin{aligned} L(\mathsf{C}^{(m)}) &= \sum_{x \in \mathcal{A}^{(m)}} p^{(m)}(x) l(x) \\ &= p^{(m)}(y) l(y) + p^{(m)}(z) l(z) + \sum_{x \in \mathcal{A}^{(m)} - \{y,z\}} p^{(m)}(x) l(x) \\ &= (p^{(m)}(y) + p^{(m)}(z))(l(w) + 1) + \sum_{x \in \mathcal{A}^{(m+1)} - \{w\}} p^{(m+1)}(x) l(x) \\ &= p^{(m+1)}(w)(l(w) + 1) + \sum_{x \in \mathcal{A}^{(m+1)} - \{w\}} p^{(m+1)}(x) l(x) \\ &= p^{(m+1)}(w) + L(\mathsf{C}^{(m+1)}). \end{aligned} \tag{5.9}$$

Now we note that if $\mathsf{C}^{(m+1)}$ is optimal, then necessarily $\mathsf{C}^{(m)}$ must be optimal too, since the operations performed retain optimality. Hence, we have shown that the Huffman code is an optimal prefix code, and thus, an optimal code in general.

While the Huffman code is optimal, it does have its drawbacks. Naturally, the source symbol probabilities must be correct for the code to be optimal and the source symbols must be independent and identically distributed (iid). Furthermore, since it encodes each symbol separately with an integer number of bits, the Huffman code can be up to one bit above the entropy and this may increase the bit rate significantly at low rates. The latter problem (and partly the problem of having strings that are not iid) can be eliminated by gathering sequences of symbols into supersymbols, and have the Huffman code operate on these supersymbols. However, the size of the tables required for this supersymbol-based approach grows exponentially with the length of the sequences. Thus, if supersymbols are used, the Huffman code requires a large amount of storage. Furthermore, the computational effort becomes very large if the code must be adapted to changing probabilities. Arithmetic codes, which are discussed in the next section, do not suffer from these disadvantages.

5.2.3 Arithmetic Codes

Recursive arithmetic codes were originally developed by Elias in the fifties but not published. About two decades later, practical implementations for finite-precision machines were developed by Pasco and Rissanen [21, 22].

Arithmetic codes are based on partitioning the interval $[0, 1)$ into cells that are each associated with a source symbol. The size of the cells is proportional to the probability

of the symbol, as is shown for an example in figure 5.2. The codewords are truncated binary representations of the midpoints of the cells. In the following, we start with describing the requirements for such simple truncation-based codes and then move to the proper arithmetic codes, which handle sequences of source symbols in an efficient manner.

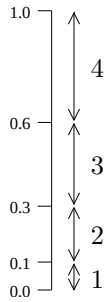

Figure 5.2: The cells that partition $[0, 1)$ for the source symbols 1, 2, 3, and 4 with probabilies 0.1, 0.2, 0.3, and 0.4, respectively.

Truncation-Based Coding

Let us consider a discrete random variable X with alphabet $\mathcal{A}$ of finite cardinality $|\mathcal{A}|$. Without loss of generality we consider the alphabet to be $\mathcal{A} = \{1, 2, \cdots, |\mathcal{A}|\}$. The corresponding cumulative distribution function is

$$F_X(x) = \sum_{j=1}^{x} p_X(j). \tag{5.10}$$

We extend the domain of the cumulative distribution function with the symbol 0, with $F_X(0) = 0$. The cumulative distribution $F_X(x)$ can then be associated with a partition of the interval $[0, 1)$ into cells $[F_X(x-1), F_X(x))$, for $x \in \mathcal{A}$. Each cell $[F_X(x-1), F_X(x))$ represents a symbol x of the alphabet $\mathcal{A}$. Thus, we must identify each cell of the partition of $[0, 1)$ with a unique codeword.

To identify each cell with a unique codeword, we use an l-bit representation to specify the location of the midpoint g of the cell. To guarantee that the truncation to l bits of $g = (F_X(x-1) + F_X(x))/2$ does not result in a value less than $F_X(x-1)$, the least significant bit should span an interval less than $(F_X(x) - F_X(x-1))/2$ (see figure 5.3). That is, the requirement is that

$$l = \min_{m \in \mathcal{N}} \{m \ : \ 2^{-m} < (F_X(x) - F_X(x-1))/2\} \tag{5.11}$$

or, equivalently

$$\begin{aligned} l &= \min_{m \in \mathcal{N}} \{m \ : \ m > -\log_2((F_X(x) - F_X(x-1))/2)\} \\ &= \lceil -\log_2(F_X(x) - F_X(x-1)) \rceil + 1 \\ &= \lceil -\log_2(p_X(x)) \rceil + 1. \end{aligned} \tag{5.12}$$

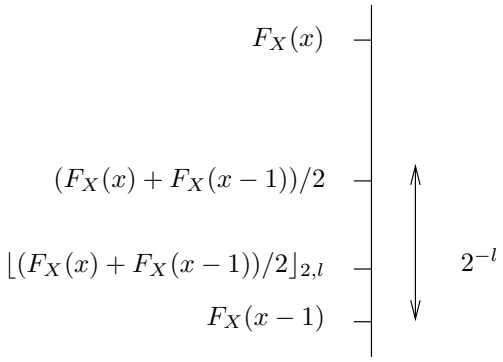

Figure 5.3: The interval $[F_X(x-1), F_X(x))$, the midpoint $g = (F_X(x) + F_X(x-1))/2$, its l-bit representation $\lfloor (F_X(x) + F_X(x-1))/2 \rfloor_{2,l}$, and the range, 2^{-l}, of points that truncate to the same l-bit representation.

We must also verify that the l-bit codeword for a cell $[F_X(x-1), F_X(x))$ cannot be the prefix for a cell $[F_X(x), F_X(x+1))$. Let $\lfloor g \rfloor_{2,l}$ denote the value of the l-bit truncated representation of $g = (F_X(x) + F_X(x-1))/2$. It is clear (see also figure 5.3) that the interval $[\lfloor g \rfloor_{2,l}, \lfloor g \rfloor_{2,l} + 2^{-l})$ falls entirely within the interval $[F_X(x-1), F_X(x))$. This confirms that the l-bit codeword cannot be a prefix to a codeword identifying another cell. Thus, the truncated midpoints form the codewords of a prefix code identifying the cells of the partition.

Table 5.2: Truncation-based encoding for an alphabet $\mathcal{A}$.

1. for a source symbol $x \in \mathcal{A}$ compute $g = (F_X(x-1) + F_X(x))/2$
2. truncate g to $\lceil -\log_2(p_X(x)) \rceil + 1$ bits to obtain the codeword for x

The truncation-based encoding algorithm for a single variable is shown in table 5.2. To decode, we need to compare the truncated value $\lfloor g \rfloor_{2,l}$ to the cumulative distribution function $F_X(x)$, and select $x = \min_{y \in \mathcal{A}} F_X(y) > \lfloor g \rfloor_{2,l}$.

In our derivation of the truncation-based code, we tacitly selected a particular partition of the interval $[0, 1)$. Other partitions can be used, and we must consider the merits of the partition based on $F_X(x)$ that we selected. The selection leads to codewords that are of length $\lceil -\log_2(p_X(x)) \rceil + 1$. If we ignore the rounding and the additional one bit overhead, the selected partition provides the optimal codeword length given by equation 5.6. That is, it is impossible to do much better by selecting another partition. More-over, truncation-based codes are generally used with large alphabets and, thus, long codewords, which means that the partition is essentially optimal.

While the partition selected is good, when comparing truncation-based coding to the Shannon code (equation 5.7) and the Huffman code, we see that its performance is at least one bit worse. The truncation-based code is within two bits of the entropy. At first sight, this suggests there is no advantage to using a truncation-based code. However, a major distinguishing factor of the truncation-based code is that we can compute the codeword for a particular symbol x without computing the codewords for other

symbols in $\mathcal{A}$. Furthermore, as we will see below, in *arithmetic coding* we compute these codewords efficiently in a recursive manner. This means that truncation-based (arithmetic) codes facilitate continuous adaptation to empirical probabilities and are useful for alphabets with large cardinality.

Example 5.4: Truncation-based coding
Consider the random variable with alphabet $\mathcal{A} = \{1, 2, 3, 4\}$ with probability mass function $p_X(1) = 0.1$, $p_X(2) = 0.2$, $p_X(3) = 0.3$, and $p_X(4) = 0.4$. We want to determine the truncation-based code. The cell partitioning is $\{F_X(x)\}_{x\in\mathcal{A}}$ with $F_X(0) = 0$, as shown in figure 5.2. We then take the midpoints of the interval, $g = (F_X(x-1) + F_X(x))/2$, and truncate their description to $\lceil -\log_2(p_X(i))\rceil + 1$ bits and get the results shown in table 5.3.

Table 5.3: The truncation-based code of example 5.4.

x	$p_X(x)$	$F_X(x)$	$g = (F_X(x-1) + F_X(x))/2$	$\lceil -\log_2(p_X(x))\rceil + 1$	codeword
1	0.1	0.1	0.05	5	00001
2	0.2	0.3	0.20	4	0011
3	0.3	0.6	0.45	3	011
4	0.4	1.0	0.80	3	110

Let us compare the performance of Huffman and truncation-based codes for sequences of symbols. Both Huffman and truncation-based codes become more efficient when longer sequences are encoded with a single codeword, since the bound on the overhead (one bit for Huffman code, two bits for truncation-based code) is then spread over more source symbols. However, for the Huffman code this is associated with exponential growth of the encoding table with the sequence length, limiting the practicality of this solution. The computational effort for the truncation-based code is linear with the number of samples, facilitating applications requiring the encoding of long sequences.

Unconstrained-Precision Arithmetic Coding

The real advantage of truncation-based coding is the possibility to encode a sequence of symbols, i.e., a message, recursively. Consider the sequence $x^k = [x_1, x_2, \cdots, x_k]$, where each sample is a realization of the random variable X_i with alphabet $\mathcal{A} = \{1, \cdots, |\mathcal{A}|\}$ and probability distribution $p_X(x_i)$. To use the truncation-based code, we must define a cumulative distribution function for a vector x^k. That is, we must partition the interval $[0, 1)$ so that each message corresponds to a unique cell. We define the vector cumulative distribution function $F_{X^k}(x^k)$ as

$$F_{X^k}(x^k) = F_X(x_1 - 1) + p_X(x_1)(F_X(x_2 - 1) + p_X(x_2)(F_X(x_3 - 1) + \cdots))). \quad (5.13)$$

The definition of $F_{X^k}(x^k)$ implies a particular truncation-based code for the entire sequence x^k. Most importantly, the structure of equation 5.13 immediately provides a recursive encoding procedure to obtain $F_{X^k}(x^k)$ and $F_{X^k}([x_1, \cdots, x_{k-1}, x_k - 1])$ (which correspond to $F_X(x)$ and $F_X(x-1)$, respectively, for the single-variable case). The

algorithm is shown in the first four steps of the algorithm in table 5.4. The values $b_{\text{bot}}^{(k)}$ and $b_{\text{top}}^{(k)}$ form $F_{X^k}([x_1, \cdots, x_{k-1}, x_k - 1])$ and $F_{X^k}(x^k)$, respectively. Note that we only compute these two values of the cumulative distribution function, thus minimizing computational effort.

The arithmetic code of x^k is the truncated binary representation of

$$g = (F_{X^k}([x_1, \cdots, x_{k-1}, x_k - 1]) + F_{X^k}(x^k))/2 \tag{5.14}$$

in the interval $[0, 1)$ rounded to

$$\lceil -\log(F_{X^k}(x^k) - F_{X^k}([x_1, \cdots, x_{k-1}, x_k - 1])) \rceil + 1 \tag{5.15}$$

bits. These operations correspond to step 5. and step 6. in table 5.4. The arithmetic encoding procedure is illustrated in example 5.5.

Table 5.4: Unconstrained-precision arithmetic encoding of a source sequence $[x_1, \cdots, x_k]$ with all $x_i \in \mathcal{A} = \{1, \cdots, |\mathcal{A}|\}$.

1. initialize:
   $$\begin{aligned} i &= 0 \\ b_{\text{bot}}^{(0)} &= 0 \\ b_{\text{top}}^{(0)} &= 1 \end{aligned}$$
2. update bounds:
   $$\begin{aligned} b_{\text{bot}}^{(i+1)} &= b_{\text{bot}}^{(i)} + (b_{\text{top}}^{(i)} - b_{\text{bot}}^{(i)}) F_X(x_{i+1} - 1) \\ b_{\text{top}}^{(i+1)} &= b_{\text{bot}}^{(i)} + (b_{\text{top}}^{(i)} - b_{\text{bot}}^{(i)}) F_X(x_{i+1}) \end{aligned}$$
3. update counter: $i := i + 1$
4. if $i < k$ go to 2
5. find g:
   $g = (b_{\text{bot}}^{(k)} + b_{\text{top}}^{(k)})/2$
6. truncate g to $\lceil -\log_2(b_{\text{top}}^{(k)} - b_{\text{bot}}^{(k)}) \rceil + 1$ bits

Example 5.5: Arithmetic encoding of a sequence
Let us consider a sequence of independent random variables, each with alphabet $\{1, 2, 3, 4\}$ and probabilities $p_X(1) = 0.1$, $p_X(2) = 0.2$, $p_X(3) = 0.3$, and $p_X(4) = 0.4$. What is the arithmetic code for the sequence $[2, 4, 3, 3]$? The process of selecting the bounds is illustrated in figure 5.4. The probability of this sequence is 0.0072 and the codeword length is thus 9 bits. The midpoint between the bounds $b_2^{(4)}$ and $b_3^{(4)}$ is 0.2548, and with 9-bit precision this results in the code 010000010.

The decoding can be performed with a similar process as the encoding. The procedure is shown in table 5.5. The binary encoding is first converted into a floating-point number, g. The symbols are identified one at a time by recursively determining the region boundaries for the successive symbols. The decoding is terminated when $\lceil -\log(b_{\text{top}}^{(i)} -$

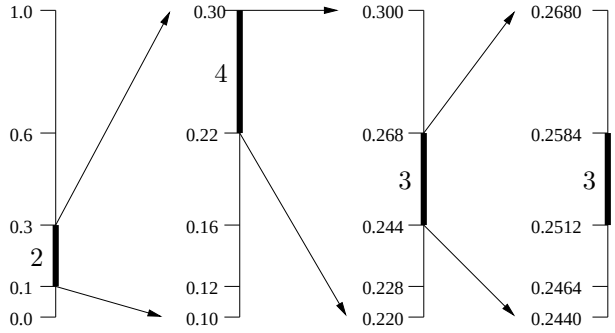

Figure 5.4: The arithmetic-coder recursion for the sequence $[2, 4, 3, 3]$, with alphabet $\{1, 2, 3, 4\}$ of example 5.5.

$b_{\text{bot}}^{(i)})\rceil + 1$ equals the number of bits of the encoded message. To know when this is the case, the arithmetic code requires a termination character or an initial specification of the number of bits that are encoded. (Note that if we encode a particular finite symbol sequence with a Huffman code for each symbol, we would also need to specify when the Huffman coded sequence ends and another type of encoding starts.)

Table 5.5: Unconstrained-precision arithmetic decoding of a codeword encoding a source sequence with alphabet $\mathcal{A} = \{1, \cdots, |\mathcal{A}|\}$.

1. binary codeword has l bits
   g is decimal representation of binary codeword
2. initialize:
   $$\begin{aligned} i &= 0 \\ b_{\text{bot}}^{(0)} &= 0 \\ b_{\text{top}}^{(0)} &= 1 \end{aligned}$$
3. find x_{i+1}:
   $$x_{i+1} = \underset{x \in \mathcal{A}}{\text{argmin}} \{x : b_{\text{bot}}^{(i)} + (b_{\text{top}}^{(i)} - b_{\text{bot}}^{(i)}) F_X(x) > g\}$$
4. update bounds:
   $$\begin{aligned} b_{\text{bot}}^{(i+1)} &= b_{\text{bot}}^{(i)} + (b_{\text{top}}^{(i)} - b_{\text{bot}}^{(i)}) F_X(x_{i+1} - 1) \\ b_{\text{top}}^{(i+1)} &= b_{\text{bot}}^{(i)} + (b_{\text{top}}^{(i)} - b_{\text{bot}}^{(i)}) F_X(x_{i+1}) \end{aligned}$$
5. update counter: $i := i + 1$
6. if $\lceil -\log_2(b_{\text{top}}^{(i)} - b_{\text{bot}}^{(i)})\rceil + 1 < l$ go to 3

Finite-Precision Arithmetic Coding

While the above encoding and decoding procedures should work in theory, practical problems occur upon their implementation. First, computers use finite-precision repre-

sentations, thus limiting the precision with which the bounds $b_{\text{bot}}^{(i)}$ and $b_{\text{top}}^{(i)}$ and the value g can be written. Second, encoding can start only once the entire sequence $[x_1, \cdots, x_k]$ is known and decoding can only start once the entire codeword has been received. These problems can be eliminated by explicitly integerizing all variables in the algorithm and by using scaling and incremental coding. At the encoder we release bits to the channel as soon as their value is known. Similarly, at the decoder this means that the source symbols are made available as soon as the received information is sufficient to specify them and the corresponding bit stream segment is destroyed.

To operate with integer arithmetic, the cumulative distribution function is replaced by a cumulative frequency count:

$$\tilde{F}_X(x) = \lfloor (M-1) F_X(x) \rfloor, \tag{5.16}$$

where we will assume that M is a power of two that is large enough that the cumulative distribution is represented with sufficient precision and, at the same time, realizable on a computer. The bounds $b_{\text{bot}}^{(i)}$ and $b_{\text{top}}^{(i)}$ are replaced by corresponding integer bounds $\tilde{b}_{\text{bot}}^{(i)}$ and $\tilde{b}_{\text{top}}^{(i)}$. To retain consistency, we define the cells as before, $[\tilde{b}_{\text{bot}}^{(i)}, \tilde{b}_{\text{top}}^{(i)})$. That is, the integer value $\tilde{b}_{\text{top}}^{(i)})$ is not included in the cell. (This convenient choice costs one bit in precision for $\tilde{F}_{X^k}$ and for this reason other conventions are commonly used cf., e.g., [23].)

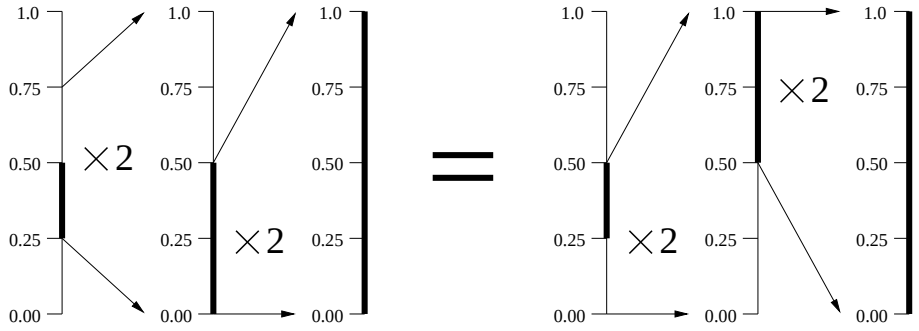

Figure 5.5: Illustration showing that a center scaling followed by a lower scaling is identical to a lower scaling followed by an upper scaling.

In the unconstrained-precision arithmetic coding algorithm, the descriptions of the cell boundaries are specified with increasing precision whenever a new source symbol is read. In the finite-precision algorithm, this precision increase is balanced by processing the information about a source symbol immediately and removing it from g by bit-wise left shifts. Let us consider scaling at the encoder first. It is clear that if $\tilde{b}_{\text{top}}^{(i)} \leq M/2$, then the most significant bit of the codeword must be 0. Thus, we can release a zero to the channel. We account for this in the bounds by a left shift, or, equivalently, by a multiplication by a factor two. This means the most significant bit is lost. We refer to this as a *lower scaling*. Similarly, if $\tilde{b}_{\text{bot}}^{(i)} \geq M/2$, then the most significant bit of the codeword must be 1. We can release a one to the channel and account for that in the bounds by subtracting $M/2$ and followed by a left shift. Again the most significant bit is lost. We refer to this as an *upper scaling*.

The upper and lower scaling strategies do not function for cells that have a lower bound at or below $M/2$ and an upper bound above $M/2$. Thus, if successive source symbols

Table 5.6: Finite-precision arithmetic encoding of a source sequence $[x_1, \cdots, x_k]$ with all $x_i \in \mathcal{A} = \{1, \cdots, |\mathcal{A}|\}$. The symbol $|\mathcal{A}|$ is reserved for end-of-message and $x_k = |\mathcal{A}|$. It is given the lowest possible probability that the precision allows. The range of $\tilde{F}$ is $[0, M-1]$. To guaranteed that the upper and lower bounds are never identical, all symbols must have a probability larger than 4.

1. initialize:
   $$\begin{aligned} i &= 0 \\ \tilde{b}_{\text{bot}}^{(0)} &= 0 \\ \tilde{b}_{\text{top}}^{(0)} &= \tilde{F}(|\mathcal{A}|) = M-1 \\ c &= 0 \end{aligned}$$
2. update bounds:
   $$\begin{aligned} \tilde{b}_{\text{bot}}^{(i+1)} &= b_{\text{bot}}^{(i)} + \lfloor (\tilde{b}_{\text{top}}^{(i)} - \tilde{b}_{\text{bot}}^{(i)}) \tilde{F}_X(x_{i+1}-1)/M \rfloor \\ b_{\text{top}}^{(i+1)} &= b_{\text{bot}}^{(i)} + \lfloor (\tilde{b}_{\text{top}}^{(i)} - \tilde{b}_{\text{bot}}^{(i)}) \tilde{F}_X(x_{i+1})/M \rfloor \end{aligned}$$
3. scale: while ($\tilde{b}_{\text{top}}^{(i+1)} < M/2 \mid \tilde{b}_{\text{bot}}^{(i+1)} \geq M/2 \mid (\tilde{b}_{\text{bot}}^{(i+1)} \geq M/4$ and $\tilde{b}_{\text{top}}^{(i+1)} < 3M/4)$):
   if $\tilde{b}_{\text{top}}^{(i+1)} \leq M/2$:
   release to channel: 0 and c times 1
   $c = 0$
   else if $\tilde{b}_{\text{bot}}^{(i+1)} \geq M/2$:
   release to channel: 1 and c times 0
   $c = 0$
   $\tilde{b}_{\text{bot}}^{(i+1)} := \tilde{b}_{\text{bot}}^{(i+1)} - M/2$
   $\tilde{b}_{\text{top}}^{(i+1)} := \tilde{b}_{\text{top}}^{(i+1)} - M/2$
   else if $\tilde{b}_{\text{bot}}^{(i+1)} \geq M/4$ and $\tilde{b}_{\text{top}}^{(i+1)} < 3M/4$:
   $c := c + 1$
   $\tilde{b}_{\text{bot}}^{(i+1)} := \tilde{b}_{\text{bot}}^{(i+1)} - M/4$
   $\tilde{b}_{\text{top}}^{(i+1)} := \tilde{b}_{\text{top}}^{(i+1)} - M/4$
   $\tilde{b}_{\text{bot}}^{(i+1)} := 2\tilde{b}_{\text{bot}}^{(i+1)}$
   $\tilde{b}_{\text{top}}^{(i+1)} := 2\tilde{b}_{\text{top}}^{(i+1)}$
4. update counter: $i := i + 1$
5. if $x_{i-1} \neq |\mathcal{A}|$ go to 2
6. truncate $g = (\tilde{b}_{\text{bot}}^{(k)} + \tilde{b}_{\text{top}}^{(k)})/(2M)$ to $\lceil -\log_2((\tilde{b}_{\text{top}}^{(k)} - \tilde{b}_{\text{bot}}^{(k)})/M) \rceil + 1$ bits

are represented by this cell, no upper and lower scalings occur, and another scaling strategy, *center scaling*, must be used to prevent problems with machine precision. In center scaling we check that $\tilde{b}_{\text{bot}}^{(i+1)} \geq M/4$ and $\tilde{b}_{\text{top}}^{(i+1)} \leq 3M/4$ and then subtract $M/4$ from the bounds and multiply them by two. As for upper and lower scalings, center scaling reduces the precision of the bounds by one bit. However, in contrast to upper and lower scalings, it is not immediately clear for a central scaling what information (bits) should be released to the channel. We simply note that we performed a center scaling and move on. That is, we have a counter c that counts the number of consecutive center scalings. (The value of c is set to zero after each lower and upper scaling.) For each lower and upper scaling we note the value of c and then release $c+1$ bits. If we have c center scalings followed by a lower scaling, then the first bit should be a 0 since the cell is located below $M/2$. Given this first 0, we know that the cell must be in the interval of length $M/(2^{c+1})$ adjacent to $M/2$, and that means the remaining c bits must all be 1. A graphic explanation of this procedure is provided in figure 5.5 for the case $c = 1$. The same principles holds when c center scalings are followed by an upper scaling. In this case the lower bound of the cell starts at $M/2$ or higher, and the first bit to be released to the channel must be a 1. The remaining c bits should all be 0, again since the cell is located in the interval of length $M/(2^{c+1})$ that is adjacent to $M/2$. The resulting encoder is shown in table 5.6.

The finite-precision decoding procedure has similar changes from the unconstrained-precision algorithm as the encoder. The algorithm is shown in table 5.7. In the decoder it is essential to make sure that the variables are bit-exact replicas of those in the encoder. Thus, the encoding and decoding algorithms are started and scaled in an identical manner. Furthermore, it is important in the decoder that the $\tilde{g}$ (the integer equivalent of g) and the integer bounds $\tilde{b}_{\text{bot}}^{(0)}$ and $\tilde{b}_{\text{top}}^{(0)}$ are subjected to identical scalings. Thus, we start with $\log_2(M)$ bits of precision for both $\tilde{g}$ and the bounds. Furthermore, $\tilde{g}$ and the bounds are scaled up together at the end of step 5. All scalings (lower, upper, and center) correspond to reducing the precision of g by 1 bit, allowing the reading of one new bit (which is included in the last part of step 5.).

Adaptive Arithmetic Coding

Arithmetic coding requires computation rather than tables (as, e.g., Huffman codes) to find the code for a particular source symbol sequence. This means that no additional effort is required to adapt the code to a changing probability mass function. The adaptation can be performed on a sample-by-sample basis. That is, the (scaled) cumulative distribution function $\tilde{F}(x_i)$ changes with time and can be written as $\tilde{F}^{(i)}(x_i)$. The operation of the method is not affected by this change. The adaptation is generally based on measurements on the data stream, which is available at both encoder and decoder. Methods for computing the probability distribution of the symbols in the alphabet are provided in section 4.2.

5.2.4 Sensitivity to Probability Mass Function Accuracy

So-far in this chapter, we have assumed that the probability mass function was known. In many cases, the probability mass function has been misestimated. How disastrous is a mismatch between the actual probability mass function and the one we use to design

Table 5.7: Finite-precision arithmetic decoding corresponding to the encoding of table 5.6. The function next_bit() returns the next bit from the channel; when no more bits are available from the channel it returns arbitrary bits. To guaranteed that the upper and lower bounds are never identical, all symbols must have a probability larger than 4.

1. initialize:

$$\begin{aligned} i &= 0 \\ \tilde{b}^{(0)}_{\text{bot}} &= 0 \\ \tilde{b}^{(0)}_{\text{top}} &= \tilde{F}(|\mathcal{A}|) = M - 1 \\ \tilde{g} &= 0 \end{aligned}$$

2. read first $m = \log_2(M)$ bits; do m times $\tilde{g} := 2 * \tilde{g} + \text{next\_bit}()$

3. find x_{i+1}: $x_{i+1} = \underset{x \in \mathcal{A}}{\text{argmin}} \{x : \tilde{b}^{(i)}_{\text{bot}} + \lfloor (\tilde{b}^{(i)}_{\text{top}} - \tilde{b}^{(i)}_{\text{bot}}) \tilde{F}_X(x)/M \rfloor > \tilde{g}\}$

4. update bounds:

$$\begin{aligned} \tilde{b}^{(i+1)}_{\text{bot}} &= \tilde{b}^{(i)}_{\text{bot}} + \lfloor (\tilde{b}^{(i)}_{\text{top}} - \tilde{b}^{(i)}_{\text{bot}}) \tilde{F}_X(x_{i+1} - 1)/M \rfloor \\ \tilde{b}^{(i+1)}_{\text{top}} &= \tilde{b}^{(i)}_{\text{bot}} + \lfloor (\tilde{b}^{(i)}_{\text{top}} - \tilde{b}^{(i)}_{\text{bot}}) \tilde{F}_X(x_{i+1})/M \rfloor \end{aligned}$$

5. scale: while ($\tilde{b}^{(i+1)}_{\text{top}} < M/2 \mid \tilde{b}^{(i+1)}_{\text{bot}} \geq M/2 \mid (\tilde{b}^{(i+1)}_{\text{bot}} \geq M/4$ and $\tilde{b}^{(i+1)}_{\text{top}} < 3M/4)$):

   if $\tilde{b}^{(i+1)}_{\text{bot}} \geq M/2$:

$$\begin{aligned} \tilde{b}^{(i+1)}_{\text{bot}} &:= \tilde{b}^{(i+1)}_{\text{bot}} - M/2 \\ \tilde{b}^{(i+1)}_{\text{top}} &:= \tilde{b}^{(i+1)}_{\text{top}} - M/2 \\ \tilde{g} &:= \tilde{g} - M/2 \end{aligned}$$

   else if $\tilde{b}^{(i+1)}_{\text{bot}} \geq M/4$ and $\tilde{b}^{(i+1)}_{\text{top}} < 3M/4$:

$$\begin{aligned} \tilde{b}^{(i+1)}_{\text{bot}} &:= \tilde{b}^{(i+1)}_{\text{bot}} - M/4 \\ \tilde{b}^{(i+1)}_{\text{top}} &:= \tilde{b}^{(i+1)}_{\text{top}} - M/4 \\ \tilde{g} &:= \tilde{g} - M/4 \end{aligned}$$

   $\tilde{b}^{(i+1)}_{\text{bot}} := 2\tilde{b}^{(i+1)}_{\text{bot}}$
   $\tilde{b}^{(i+1)}_{\text{top}} := 2\tilde{b}^{(i+1)}_{\text{top}}$
   $\tilde{g} := 2 * \tilde{g} + \text{next\_bit}()$

6. update counter: $i := i + 1$

7. if $x_i \neq |\mathcal{A}|$ go to 3

a code? In this section, we will find that if the relative entropy between the probability densities is small, then the loss in performance compared to the optimal case is small as well.

Let us consider random variable X with probability mass function $p_X(x)$. Furthermore let us define (as in section 2.4.3)

$$p_L(x) \equiv \frac{2^{-l(x)}}{\sum_{x\in\mathcal{A}} 2^{-l(x)}}. \tag{5.17}$$

Then it straightforward to see that

$$\begin{aligned} L &= \sum_{x\in\mathcal{A}} p_X(x) l(x) \\ &= -\sum_{x\in\mathcal{A}} p_X(x)\log_2(p_L(x)) - \sum_{x\in\mathcal{A}} p_X(x)\log_2(\sum_{x\in\mathcal{A}} 2^{-l(x)}) \\ &\geq -\sum_{x\in\mathcal{A}} p_X(x)\log_2(p_L(x)) \\ &= -\sum_{x\in\mathcal{A}} p_X(x)\log_2(p_X(x)) + \sum_{x\in\mathcal{A}} p_X(x)\log_2(\frac{p_X(x)}{p_L(x)}) \\ &= H(X) + H(p_X||p_L), \end{aligned} \tag{5.18}$$

where we exploited the Kraft inequality $\sum_{x\in\mathcal{A}} 2^{-l(x)} \leq 1$ that is valid for uniquely decodable codes (see section 2.4.1). Thus, the average codeword length is larger than or equal to the sum of the entropy of X and the relative entropy between $p_X(x)$ and $p_L(x)$. This bound is reached when the Kraft inequality becomes an equality.

Now we can give, albeit somewhat imprecisely, a description of the impact of using an incorrect probability mass function instead of the correct one. The precise effect depends on the particularities of the design method and the parameters at hand. However, since $p_L(x)$ will be close to the incorrect probability mass function $q_X(x)$, equation 5.18 suggests that, at least for higher rates, the relative entropy $H(p_X||q_X)$ will be a good indicator of the mismatch. This suggests the expected result that if $q_X(x)$ is close to $p_X(x)$, then coding performance will be good in practice.

Example 5.6: Code length for misestimated distribution
Let us consider a random variable X with an alphabet $\mathcal{A} = \{a, b\}$ and $p_X(a) = 0.2$ and $p_X(b) = 0.8$ and an entropy $H(X) = 0.72$. If we use instead the incorrect estimate $\hat{p}_X(a) = \hat{p}_X(b) = 0.5$, the bit rate will increase. The relative entropy is $H(p_X||\hat{p}_X) = 0.28$ and a uniquely decodable code will most likely have a code length of more than $L = 0.72 + 0.28 = 1$ bit. This is significantly more than the entropy of the random variable.

5.2.5 Run-Length Coding

Run-length coding is a computationall efficient method that is applicable when long sequences of identical symbols occur. This is commonly the case in image coding. Run-length coding reduces the rate by specifying the number of identical subsequent symbols.

This may take the form of explicitly listing the number of symbols of a particular value that follows. For block-wise coding, it is common to indicate that all remaining symbols are identical to the previous symbols.

Often, run-length coding is used as a first step in a lossless coding procedure. As a second step, the resulting symbol stream is subjected to either Huffman or arithmetic coding.

5.3 Universal Coding

In many practical applications, the actual probability mass function of the alphabet is not known. This problem can be resolved with universal codes. In the following, we distinguish "true" universal codes, where the probability mass function is never made explicit, and **adaptive codes**, which are conventional codes augmented with a estimation procedure for the probability mass function.

The ability to create a code without having to know the probability mass function of the random variable is very useful for many practical applications. However, it would be too good to be true if such flexibility did not have its price. An obvious question is whether universal codes can be asymptotically optimal. Consider a stationary and ergodic sequence. Is it possible to encode such a sequence with a universal code at a rate that asymptotically approaches the entropy rate with increasing sequence length? It turns out that this is indeed possible even for some universal codes (e.g., [7]). Unfortunately, this still does not mean that universal codes are always preferred. When the probability mass function is known, simple nonadaptive codes that exploit this information usually have as advantages a short coding delay, a low computational effort, and near-optimality for short input symbol sequences. In contrast, universal codes often suffer from one or more of the following disadvantages: long delay, high computational requirements, and slow convergence to optimality with increasing input sequence length.

We continue with a brief discussion of adaptive codes. Thereafter, we discuss a commonly used true universal code, the Ziv-Lempel code.

5.3.1 Code Adaptation and Probability Mass Estimation

The Huffman and arithmetic codes described in the previous sections require a known probability mass function. In section 4.2 we discussed methods for estimating the probability mass function from the processed data, which can be combined with the codes described in the previous sections of this chapter.

We can distinguish two common strategies to use the empirical probability mass function. Perhaps the simplest strategy is to compute the empirical probability mass function for the entire symbol sequence first, and then use this in a conventional Huffman or arithmetic code. Obviously, this method requires a long delay (since the entire sequence must be seen first) and an additional set of bits are required for a header containing the probability mass function. The second strategy is to update the probability mass function during the coding process. The decoder and encoder make the same computations, and the information about the probability mass function is not transmitted explicitly.

As we saw in section 5.2.3, the method where the probability mass function is updated as the data are processed is particularly natural for the arithmetic codes. This is a major reason for the common usage of adaptive arithmetic codes. For the Huffman code, adaptation while processing is less natural. However, computationally efficient procedures that exploit the relations between updates do exist (such a procedure is given in [24]).

5.3.2 The Ziv-Lempel Code

In the adaptive codes described above, the probability mass function for the random variable was determined empirically from the data. Thus, we could retain the structures familiar from codes requiring known probability mass functions. In this subsection, we describe the commonly used Ziv-Lempel code, which is a "proper" universal code: it does not require the explicit computation of empirical probability mass functions. It is used, for example, in the "gzip" facility of the Unix operating system. It can be shown that, for stationary and ergodic processes, the Ziv-Lempel code converges to the entropy rate of the process. For a proof of this asymptotic optimality we refer to [7].

In the Ziv-Lempel code, the source symbol sequence is first parsed (segmented) into a sequence of symbol subsequences called **phrases**. Starting from the end of the last phrase, a new phrase is the shortest symbol subsequence that has not been observed earlier. In a phrase of length N, the first $N-1$ symbols form a previously observed phrase. Thus, we can decompose each phrase into a prefix, which is the earlier observed phrase, and one additional symbol. Once the parsing is completed, each phrase is assigned an address. The codeword for a phrase consists of two parts: *i*) the address of its prefix and *ii*) the last symbol. The Ziv-Lempel code can be adapted to other alphabets, but is most commonly applied to bit sequences.

Example 5.7: Ziv-Lempel code for 011010010010111
The sequence is parsed into 0,1,10,100,1001,01,11. The coding of the phrases is shown in table 5.8. Let us consider the phrase 100: the phrase prefix is 10, which has address 011. Concatenating the last bit of the phrase to the address of the phrase prefix gives as code 0110. The code for the sequence is thus 000000010100011010010011010 1. In this example, the code is very inefficient and this is typical for short sequences.

Table 5.8: Ziv-Lempel code for the binary sequence "011010010010111" of example 5.7.

phrase	address	prefix address	code
0	001	000	0000
1	010	000	0001
10	011	010	0100
100	100	011	0110
1001	101	100	1001
01	110	001	0011
11	111	010	0101

In its basic form, the Ziv-Lempel algorithm requires two passes. The first pass is used to determine the number of phrases, and thus the number of bits required to describe the prefix address, which is $\lceil \log(\text{number of phrases}) \rceil$. The empty phrase is included in the number of phrases and has as address all zeros. In the second pass the actual code is created.

The inversion procedure for the Ziv-Lempel code is straightforward. An example is treated in problem 8.

5.4 Problems

1. Consider the symbols with probabilities $\{0.01, 0.09, 0.2, 0.2, 0.2, 0.3\}$.
   (a) Design a Shannon code for the set.
   (b) Design a binary Huffman code for the set.
   (c) Find the average codeword length for Shannon and Huffman codes and compare this with the entropy.
2. Which of the following codes cannot be a Huffman code and why?
   (a) $\{0, 10, 11\}$.
   (b) $\{00, 01, 001, 100, 101, 110, 111\}$.
   (c) $\{0, 10, 1100, 1101\}$.
   (d) $\{1, 01, 0001, 0010, 0011\}$.
3. For many codes satisfying the Kraft inequality, the inequality is strict.
   (a) Prove that for $\{0, 10, 1100, 1101, 1110\}$, the inequality is strict.
   (b) Provide an example of a sequence of code symbols undecodable by this code.
   (c) Relate the strict inequality to the fact that there exist sequences of code symbols that cannot be decoded.
4. Consider the generalization of the presented Huffman procedures so that we can design ternary codes (i.e., codes using three coding symbols). We consider the source symbols with probabilities $\{0.1, 0.2, 0.2, 0.2, 0.3\}$.
   (a) Design a binary Huffman code for the source.
   (b) Design a ternary Huffman code for the source.
   (c) Which code is closer to optimal? Is there a fundamental reason for this?
5. Similarly to an arithmetic code, a Shannon code can also be associated with a partition of the interval $[0, 1)$.
   (a) Show that you can construct a Shannon prefix code by truncating an associated cumulative distribution function. (Hint: it is important to select the correct order for the alphabet elements.)
   (b) The above Shannon code cannot be applied to a source symbol sequence in the iterative manner that makes arithmetic codes so powerful. Explain why the Shannon code fails on this account.

6. Consider a sequence of length k of discrete-amplitude iid samples, X_i. We use a Huffman code to encode the individual samples of the sequence. We know that the Huffman code is within one bit of the sample entropy and that it is optimal. Find the Huffman codes when the random variables X_i are quantization indices of a uniform quantizer with step size $\frac{1}{M}\sigma_{X_i}$ and a range of $[-3\sigma_{X_i}, -3\sigma_{X_i}]$ operating on a Gaussian density with with variance σ^2. Use the step sizes $M = 2, 4, 8, 16, 32$. Compare the resulting mean codeword length to the entropy and with the mean codeword length obtained when the indices have equal probability.

7. Consider an alphabet $\{1, 2, 3\}$ and a sequence 2311.

   (a) Using Laplace's rule, compute the empirical probability mass function.

   (b) Design a Huffman code and encode the sequence with it.

   (c) Encode the sequence with an arithmetic code (use the above probability mass function). Show the steps explicitly.

   (d) Decode the coded sequence for your arithmetic code. Show the steps explicitly.

8. (a) Obtain the Ziv-Lempel code for the sequence 000110101101000111111.

   (b) Consider the Ziv-Lempel code 000000010100011000111011001O. Decode this sequence assuming four-bit codewords.

9. Consider the alphabet $\mathcal{A} = \{A, B, C, D\}$. Program, in either Matlab or C, a universal lossless encoder and decoder that uses Laplace's rule in combination with arithmetic coding. You may use two passes, the first to obtain the probability distribution (to be written in an ASCII file with floating point numbers, which is provided to the decoder). The encoder and decoder should be able to handle sequences of symbols from $\mathcal{A}$ of arbitrary length, which means that scaling is required. Your encoder and decoder programs should read and write ASCII files. Thus, the encoder input file contains an ASCII sequence like ABBDABD and its output file an ASCII sequence like 100010111.

10. For arithmetic coding, define an efficient strategy (i.e., a strategy that does not require additional bits) to complete the bit stream if the last scaling is a center scaling. (Hint: consider scaling of g.)

11. Consider the alphabet and probability distribution of example 5.5. Write in either Matlab or C two programs that can perform arithmetic encoding and decoding, respectively, on sequences of arbitrary length (this means you need scaling). The encoder program should be named "encoder" and read its input from the text file "input.txt" containing an ASCII text sequence of symbols from the alphabet. The encoded bit stream should be called "channel.txt" and consist of ASCII 1 and 0 characters. The decoder program should be called "decoder", read "channel.txt" and write to "output.txt". Provide the (commented) programs and the bit stream for 3223421323 for 16-bit precision.

12. In this problem we perform arithmetic coding of English text. The coding exploits only the marginal probability distribution of the symbols.

   (a) Write a program that converts text to a sequence of 32 symbols. Retain the letters of the alphabet, and the space, comma and period characters. Remove capitalization and eliminate all other characters.

(b) Using tables 5.6 and 5.7 write an adaptive finite-precision arithmetic encoder and decoder programs in C or Matlab that can read the fore-mentioned text (from the file "input.txt") and writes a proper binary code ('"channel.d"). The source symbols are initially assumed to be of equal probability. The probabilities must be adapted using Laplace's rule.

(c) Estimate the first-order entropy of the simplified English text.

(d) Find the mean-codeword length per source symbol for English text of your program as a function of the number of source symbols (average over a number of runs).

6

Rate-Distortion Theory

6.1 Introduction

The storage and transmission of continuous-time, continuous-amplitude (analog) signals, such as audio and video signals, is needed in many applications. In modern technology, these signals are first converted to discrete-time, discrete-amplitude signals by an analog-to-digital (A/D) converter. While the output of an A/D converter is discrete-amplitude, it is often both convenient and accurate to neglect the amplitude quantization. From sampling theorems we know that there is, at least in principle, a one-to-one mapping between the discrete-time and band-limited continuous-time signals. This motivates us to study the minimum average bit rate required to encode discrete-time signals (with both discrete and continuous amplitudes) at a given fidelity. Naturally, this can be done only when certain conditions are imposed on the signals.

Rate-distortion theory, also called **distortion-rate theory**, deals with the determination of bounds specifying the optimal trade-off between average bit rate and average distortion for sequences of random variables (processes). Most of the theory focuses on the so-called rate-distortion function or, equivalently, the distortion-rate function. These functions specify, respectively, the lowest average rate possible for a given average distortion and the lowest average distortion possible for a given average rate.

Analytical expressions for the rate-distortion function have been obtained for only a few cases. Thus, bounds on the rate-distortion function, such as the Shannon lower bound, and a numerical method to compute the rate-distortion function, the Blahut algorithm, play an important role in rate-distortion theory.

While rate-distortion theory provides bounds on source-coding rates, it does not specify practical source-coding methods that achieve these bounds. The theory generally assumes coding methods that would require exceedingly high computational complexity because of the high dimensionality. Despite this, rate-distortion theory is often useful in the design of coders. For example, rate-distortion theory shows that, under certain conditions, the channels of a vector process can be coded independently without loosing coding efficiency. In general, rate-distortion theory is a useful analysis tool in the design of source codes, but it does not replace the need for imaginative source coder design.

6.2 The Rate-Distortion Function

When a variable is encoded and decoded in a communication process, the intention is to convey information from encoder to decoder. In other words, the essential aspect of the communication process is the sharing of information between original and decoded variables. This shared information can be quantified with mutual information. Thus, it is natural that mutual information is important for the description of the **rate-distortion function**, which is defined as a tight lower bound on the bit rate required to transmit a stationary process at a given level of distortion (fidelity). We first discuss some relevant properties of mutual information, then define the rate-distortion function, and finally discuss its properties.

6.2.1 Mutual Information, Quantization, and Noise

To prepare for the definition of the rate-distortion function, it is useful to discuss some properties of mutual information. We consider the encoding-decoding of a continuous-alphabet random variable and the addition of random noise to a random variable.

Let us consider the encoding and decoding of a continuous-alphabet random variable, X. The decoded variable is assumed to be a discrete-alphabet (quantized) variable, which we write as $\hat{X}$. The mutual information is related to the entropy $H(\hat{X})$ of the quantized variable by

$$\begin{aligned} H(\hat{X}) &\geq H(\hat{X}) - H(\hat{X}|X) \\ &= I(X;\hat{X}), \end{aligned} \tag{6.1}$$

which simply states that the mutual information cannot be more than the entropy of the decoded variable. For a conventional quantizer $H(\hat{X}|X) = 0$ since $\hat{X}$ is completely determined by X. The mutual information is then simply the entropy of the decoded variable $\hat{X}$. In general, the term $H(\hat{X}|X)$ represents information present in the decoded variable that is independent from the information in the original variable. Such information can be generated by a noisy channel or by a noisy reconstruction process.

Example 6.1: Maximum mutual information 1-bit quantizer
Consider a random variable X that has a nonzero density in the interval $[0, 1)$:

$$f_X(x) = \begin{cases} 2x, & 0 \leq x < 1, \\ 0, & \text{elsewhere.} \end{cases}$$

The 1-bit quantizer has a threshold $b \in [0, 1)$. We want to determine b such that the entropy of the quantized output and, thus, the mutual information between the variable and the quantization index is maximized. This mutual information is

$$I(\hat{X};X) = H(\hat{X}) = -b^2 \log(b^2) - (1-b^2)\log(1-b^2).$$

It is straightforward to see that the mutual information is maximum for

$$b_{\text{ME}} = \underset{b}{\operatorname{argmax}} \; -b^2 \log(b^2) - (1-b^2)\log(1-b^2) = \sqrt{\frac{1}{2}}$$

and that the mutual information in this case is one bit.

Let us now consider the mean squared error criterion. The value of b that minimizes the mean squared error is

$$b_{\mathrm{MSE}} = \operatorname*{argmin}_b \int_0^1 (x-b)^2 2x dx = \frac{2}{3}.$$

This solution provides the lowest mean squared error, but X and $\hat{X}$ share less information in this case than in the case of the the maximum mutual information solution. Less shared information between X and $\hat{X}$ suggests that a lower rate is possible.

Next, we consider the mutual information between a continuous alphabet random variable and the same random variable with additive noise. We add an independent noise variable, N, also with continuous alphabet, to a variable X to obtain a distorted variable $X+N$. The mutual information between the parameter $X+N$ and the parameter X is now simply the difference between the differential entropies of $X+N$ and N:

$$\begin{aligned} I(X; X+N) &= h(X+N) - h(X+N|X) \\ &= h(X+N) - h(N). \end{aligned} \tag{6.2}$$

The mutual information between X and $X+N$ is finite (when the relevant probability densities are integrable) even though the entropies of X and of $X+N$ are infinite. Equation 6.2 suggests that it might be possible to reconstruct a "noisy version" of an original signal (with infinite entropy rate) from a finite bit rate encoding. Note that $h(X+N) \geq h(X+N|X) = h(N)$, and that, thus, equation 6.2 satisfies the basic property that mutual information is always nonnegative.

Example 6.2: Gaussian variable with additive Gaussian noise
We compute the mutual information for a Gaussian variable with variance σ_X^2 with additive Gaussian noise σ_N^2. To this purpose, we can exploit that the summation of two independent Gaussian variables gives another Gaussian variable that has as variance the sum of the variances of the two original variables. Furthermore, we already know the differential entropy of a Gaussian variable (equation 3.25 with $k=1$). The mutual information is then

$$\begin{aligned} I(X; X+N) &= h(X+N) - h(X+N|X) \\ &= h(X+N) - h(N) \\ &= \frac{1}{2}\log(2\pi \mathrm{e}(\sigma_X^2 + \sigma_N^2)) - \frac{1}{2}\log(2\pi \mathrm{e}\sigma_N^2) \\ &= \frac{1}{2}\log(\frac{\sigma_X^2 + \sigma_N^2}{\sigma_N^2}). \end{aligned}$$

It is seen that, as we would expect, the mutual information decreases with increasing noise variance.

If the noise and variable variance are equal, then the mutual information between X and $X+N$ is a half bit. For zero noise variance, the mutual information is infinite, while for infinite noise variance the mutual information vanishes.

The discussion in this subsection suggests that the mutual information measure is related to the bit rate required for encoding a signal. In the next section, we use mutual information to define a lower bound on the bit rate possible at a certain fidelity.

6.2.2 Definition of the Rate-Distortion Function

Our goal is to find a function defining a tight lower bound on the bit rate required to transmit a stationary process at a certain level of distortion (fidelity). We consider k-dimensional random vectors, $X^k = [X_1, \cdots, X_k]^T$ for which a probability density function $f_{X^k}(x^k)$ or a probability mass function $p_{X^k}(x^k)$ is defined. We can then define an iid stationary vector process consisting of a sequence of vectors X^k and it is this process that our rate and distortion apply to. Thus, when we use rate-distortion theory, we consider the encoding of a sequence of vectors (or scalars) X^k rather than the encoding of a single X^k. An important case is where the vectors X^k are blocks of a scalar process; we then ignore the dependencies between the blocks.

In rate-distortion theory, *distortion* refers to the mean distortion between X^k and the reconstructed vector $\hat{X}^k$. We generally restrict ourselves to distortions that can be computed on a **single-letter**(per-sample) basis. If the distortion for observed samples is written as $d(x_i, \hat{x}_i)$, then the single-letter distortion of a k-dimensional vector is

$$d(x^k, \hat{x}^k) = \sum_{i=1}^{k} d(x_i, \hat{x}_i). \tag{6.3}$$

We want to specify a bound on rate for a realizable coding system given a distortion. A coding system that is, at least in principle, realizable, generally involves two deterministic mappings: one from a sequence of observed vectors x^k to a codeword (or codewords) and one from the codeword (or codewords) to an (approximate) reconstruction of the sequence. The most general deterministic mapping operates simultaneously on an infinite-length sequence of vectors rather than on individual vectors x^k. We denote a sequence of n vectors as $x^{k;n}$. The encoding and decoding operation of a vector $x^{k;n}$ together correspond to a mapping $\mathcal{Q}_n$ to a reconstruction $\hat{x}^{k;n} \in \mathcal{C}^{kn}$:

$$\mathcal{Q}_n : \mathbb{R}^{kn} \to \mathcal{C}^{kn}, \tag{6.4}$$

where $\mathcal{C}^{kn} \subset \mathbb{R}^{kn}$ is the set of reconstruction points. The mapping $\mathcal{Q}_n$ is associated with both a rate and a distortion. The **rate**, R, required to encode the random sequence $X^{k;n}$ is the *average* bit allocation used to specify a reconstruction vector $\hat{X}^{k;n}$.

We say that a rate-distortion pair, (R, D), for a random vector X^k, is **achievable** if a deterministic encoding-decoding operation $\mathcal{Q}_n(\cdot)$ exists that maps any sequence $x^{k;n}$ of n vectors x^k into one sequence of a set $\mathcal{C}^{kn}$ of 2^{nR} sequences such that

$$\lim_{n\to\infty} \frac{1}{n} \mathrm{E}[d(X^{k;n}, \mathcal{Q}_n(X^{k;n}))] \leq D. \tag{6.5}$$

Note that $X^{k;n}$ is a sequence of n vectors X^k, drawn independently from $f_{X^k}(x^k)$. The implicit choice of equal codeword length by selecting the cardinality of $\mathcal{C}^{kn}$ is reasonable given asymptotic equipartition. The rate-distortion function, $R(D)$, is the highest lower bound for the achievable rate required to perform the encoding-decoding operation with a mean distortion (in the sense of equation 6.5) less than a specified value D.

For a scalar stationary process X_i, the **order-k rate-distortion function** $R_k(D)$ is the infimum of the achievable rates for coding subsequent vectors of dimension k, with a given mean distortion D, where rate and distortion are normalized on a per sample

basis, and not accounting for dependencies between the vectors. It follows that for an iid process $R_1(D) = R_\infty(D)$. For a general scalar stationary process, the rate-distortion function $R(D)$ is defined as

$$R(D) = \min_i R_i(D) = R_\infty(D). \tag{6.6}$$

To find the rate-distortion function, we use a statistical mapping as a substitute for the deterministic encoding-decoding process. The statistical mapping from X^k to $\hat{X}^k$ is specified by the conditional probability density $f_{\hat{X}^k|X^k}(\hat{x}^k|x^k)$. We define a set $B_k(D)$ that contains all the conditional probability functions for $\hat{X}^k$ given X^k that lead to an average distortion that is less than or equal to D:

$$B_k(D) = \{f_{\hat{X}^k|X^k} : E[d(X^k, \hat{X}^k)] \leq D\}. \tag{6.7}$$

Note that in contrast to equation 6.5, the definition of $B_k(D)$ does not involve the notion of a sequence and that it does not assume the existence of a deterministic mapping $Q_n(\cdot)$. In equation 6.7 the density $f_{X^k}(x^k)$ is given, determined by the properties of the variable (or sequence), whereas $f_{\hat{X}^k|X^k}(\hat{x}^k|x^k)$ is not predetermined but related to the coding method. Importantly, with the definition of $f_{\hat{X}^k|X^k}(\hat{x}^k|x^k)$ we have not assumed a deterministic mapping from X^k to $\hat{X}^k$.

A measure of the information shared between X^k and $\hat{X}^k$ is the mutual information. For a given distortion D, the lower bound on this mutual information is

$$\inf_{f_{\hat{X}^k|X^k} \in B_k(D)} I(X^k; \hat{X}^k). \tag{6.8}$$

It is intuitive that one cannot reconstruct X^k with a distortion less or equal to D at a rate lower than this shared information. It is less obvious how this relates to the encoding of a sequence of n vectors X^k and that this rate is actually a bound on achievability. The *rate-distortion theorem* states that the infimum mutual information between X^k and $\hat{X}^k$ is, in fact, the lowest achievable rate:

Theorem 16 *For a random vector X^k with probability density $f_{X^k}(x^k)$ and with a bounded distortion criterion $d(x^k, \hat{x}^k)$, the rate-distortion function is $R(D) = \inf_{f_{\hat{X}^k|X^k} \in B_k(D)} I(X^k; \hat{X}^k)$.*

For processes, theorem 16 generalizes to the next theorem:

Theorem 17 *For a stationary process and a bounded single-letter distortion criterion, $R(D) = R_\infty(D) = \lim_{k\to\infty} \frac{1}{k} \inf_{f_{\hat{X}^k|X^k} \in B_k(kD)} I(X^k; \hat{X}^k)$,*

where we were careful to define $R(D)$ on a per-sample basis.

The rate-distortion theorem shows that finding the rate-distortion function is a constrained optimization problem: we want to find the $f_{\hat{X}^k|X^k}$ that minimizes the mutual information $I(X^k; \hat{X}^k)$ under the constraint that the distortion does not exceed D. This suggests that standard solution methods for constrained optimization, such as variational theory for continuous distributions or the Lagrange multiplier method for

discrete distributions should be used. Unfortunately, these methods do not lead to general, simple analytical solutions. However, a large number of techniques that find lower bounds on $R(D)$ or numerical representations of $R(D)$ are based on these approaches. We will discuss some of these methods in section 6.5.

In the literature, more general rate-distortion theorems can be found. In particular, rate-distortion theorems for processes that have relatively loose stationarity conditions have been derived. The proof of the rate-distortion theorems is not straightforward, particularly for less-restrictive cases. Thus, we first discuss some effects that make the rate-distortion function plausible in section 6.2.5. Then, in section 6.2.4, we prove the simpler part: that it is not possible to code at a lower rate for a given distortion than the rate specified by the rate-distortion function. In section 6.2.6 we prove that the rate given by the rate-distortion function can be reached, asymptotically with increasing sequence length, with a deterministic code. Since this part of the proof is challenging, we restrict ourselves to a simple case.

Example 6.3: Squared-error rate-distortion for Gaussian density
In this example, we compute the rate-distortion function for a normal variable. In general, it is difficult to minimize the mutual information directly, and we use a common strategy: we first find a lower bound on the mutual information, and then show that we can reach this lower bound. We write the random variable as X, its reconstruction as $\hat{X}$, and the quantization noise as $Y = \hat{X} - X$. For any X and $\hat{X}$ we have that

$$\begin{aligned} I(X;\hat{X}) &= h(X) - h(X|\hat{X}) \\ &= h(X) - h(\hat{X} - Y|\hat{X}) \\ &= h(X) - h(Y|\hat{X}) \\ &\geq h(X) - h(Y). \end{aligned}$$

If Y is selected so as to maximize $h(Y)$ for the given mean squared error D, then we have a general lower bound. We know that the density for Y that maximizes its differential entropy, $h(Y)$, is a Gaussian density with variance D and $h(Y) = \frac{1}{2}\log(2\pi \mathrm{e} D)$.

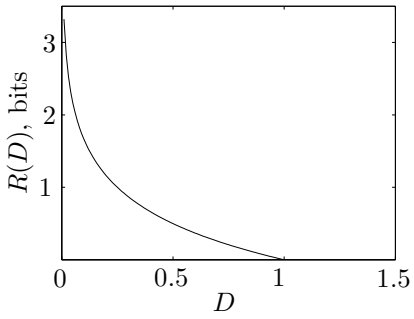

Figure 6.1: Rate-distortion function for one-dimensional Gaussian source with unit variance and squared-error criterion. Rate-distortion pairs above the rate-distortion function are achievable.

For the lower bound to be achievable we must have that $h(Y|\hat{X}) = h(Y)$. This is true if Y is independent from $\hat{X}$. That is, it should be possible to interpret the

signal X as the summation of independent variables Y and $\hat{X}$, implying that

$$\int f_{\hat{X}}(x-y)f_Y(y)dy = f_X(x).$$

It is simple to find a valid solution: we select $\hat{X}$ to be normal with variance $\sigma^2_{\hat{X}} = \sigma^2_X - D$, for $D \leq \sigma^2_X$. For $D > \sigma^2_X$ the infimum mutual information to reach a distortion D is zero; the optimal $\hat{X}$ is then simply the mean value of X. In general, the variance of the reconstruction is less than that of the original. The rate-distortion function for a Gaussian variable and a squared-error distortion criterion is thus

$$R(D) = h(X) - h(Y) = \frac{1}{2}\log(\frac{\sigma^2_X}{D}), \;\; D \leq \sigma^2_X.$$

This rate-distortion function is illustrated in figure 6.1 for the case where $\sigma^2_X = 1$.

We will return to the methods used in this example in a more general setting in section 6.4.

6.2.3 Nonincreasing and Convexity Properties

Two important properties of the rate-distortion function are that it is always nonincreasing and that it is convex. A typical example is given in figure 6.1. In this subsection, we prove the nonincreasing and convexity properties of the rate-distortion function.

First we show that the rate-distortion function is nonincreasing. The set $B_k(D)$ of conditional probability densities increases with D, with the smaller sets being subsets of the larger sets. The search for the infimum is therefore performed over a larger set for increased mean distortion D, and, as a result, the infimum found for a specific D must be less than or equal to that found at a smaller distortion. This reasoning leads to the conclusion that the rate-distortion function is strictly nonincreasing.

Next, we show that the rate-distortion function is convex. For simplicity we write down the proof for the scalar continuous-$f_{\hat{X}}(\hat{x})$ case. The logic remains identical when the scalar case is replaced with the vector and when the continuous-density case is replaced with the discrete-density case. Let $f_0(\hat{x}|x)$ be a conditional probability density for which $\int f_0(\hat{x}|x)f_X(x)d(x,\hat{x})dx \leq D_0$ and, similarly, let $f_1(\hat{x}|x)$ be a conditional probability density for which $\int f_1(\hat{x}|x)f_X(x)d(x,\hat{x})dx \leq D_1$. We define $f_\alpha(\hat{x}|x)$ with $\alpha \in [0,1]$ to be

$$f_\alpha(\hat{x}|x) \equiv \alpha f_0(\hat{x}|x) + \tilde{\alpha} f_1(\hat{x}|x), \quad \alpha \in [0,1], \tag{6.9}$$

where $\tilde{\alpha} = 1 - \alpha$. We first show that $f_\alpha(\hat{x}|x)$ is a conditional probability function that maps X to $\hat{X}$ at a mean distortion of at most $D_\alpha \equiv \alpha D_0 + \tilde{\alpha} D_1$:

$$\begin{aligned} D_\alpha &\geq \alpha \int\int f_0 f_X d(x,\hat{x})dxd\hat{x} + \tilde{\alpha}\int\int f_1 f_X d(x,\hat{x})dxd\hat{x} \\ &= \int\int f_\alpha f_X d(x,\hat{x})dxd\hat{x}, \end{aligned} \tag{6.10}$$

where we wrote f_α for $f_\alpha(\hat{x}|x)$. In the following, we will continue this notation and furthermore make the conditional probability function used explicit when writing the mutual information; we will write, for example, $I_{f_\alpha}(X;\hat{X})$ for the mutual information using the statistical mapping f_α.

Next, we show that $\inf_{f_0 \in B_1(D_0),\, f_1 \in B_1(D_1)} I_{f_\alpha}(X, \hat{X})$ is a convex function of α. We first simplify our notation further by abbreviating $\inf_{f_0 \in B_1(D_0),\, f_1 \in B_1(D_1)}$ to $\inf_{f_0, f_1}$. We thus have

$$\begin{aligned}
&\inf_{f_0,f_1} I_{f_\alpha}(X;\hat{X}) \\
&\quad = \inf_{f_0,f_1} \int\int f_X f_\alpha \log(\frac{f_X f_\alpha}{f_X f_{\hat{X}}}) dx d\hat{x} \\
&\quad = \inf_{f_0,f_1} \int\int (f_X(\alpha f_0 + \tilde{\alpha} f_1)\log(\alpha f_0 + \tilde{\alpha} f_1) - f_X(\alpha f_0 + \tilde{\alpha} f_1)\log(f_{\hat{X}})) dx d\hat{x} \\
&\quad \leq \inf_{f_0,f_1} \int\int (f_X(\alpha f_0 \log(f_0) + \tilde{\alpha} f_1 \log(f_1)) - f_X(\alpha f_0 + \tilde{\alpha} f_1)\log(f_{\hat{X}})) dx d\hat{x} \\
&\quad = \inf_{f_0,f_1} \int\int \alpha f_X f_0 \log(\frac{f_0}{f_{\hat{X}}}) + \tilde{\alpha} f_X f_1 \log(\frac{f_1}{f_{\hat{X}}}) dx d\hat{x} \\
&\quad = \inf_{f_0 \in B_1(D_0)} \alpha I_{f_0}(X;\hat{X}) + \inf_{f_1 \in B_1(D_1)} \tilde{\alpha} I_{f_1}(X;\hat{X}), \qquad (6.11)
\end{aligned}$$

where the inequality results from the fact that $x \log(x)$ is convex on $(0, 1)$.

We note furthermore that

$$\begin{aligned}
R(D_\alpha) &= \inf_{f_{\hat{X}|X} \in B_1(D_\alpha)} I(X;\hat{X}) \\
&\leq \inf_{f_0,f_1} I_{f_\alpha}(X;\hat{X}). \qquad (6.12)
\end{aligned}$$

Before continuing we summarize our results:

1. $R(D_0) = \inf_{f_0,f_1} I_{f_0}(\hat{X}; X)$.
2. $R(D_\alpha) \leq \inf_{f_0,f_1} I_{f_\alpha}(\hat{X}; X)$.
3. $R(D_1) = \inf_{f_0,f_1} I_{f_1}(\hat{X}; X)$.
4. $\inf_{f_0,f_1} I_{f_\alpha}(\hat{X}; X)$ is convex.

From these results it follows that $R(D)$ is convex, i.e. that

$$R(D_\alpha) \leq \alpha R(D_0) + \tilde{\alpha} R(D_1), \quad \forall\ \alpha \in [0, 1]. \qquad (6.13)$$

This reasoning is illustrated in figure 6.2.

We now have completed a proof of the following theorem:

Theorem 18 *The rate-distortion function $R(D)$ is convex and nonincreasing,*

for the case of continuous distributions for $\hat{X}$ and X. The cases where $\hat{X}$ and/or X have discrete distributions follows the same outline.

Knowledge of the convexity of the rate-distortion function is useful for a number of purposes. We wil use it in section 6.2.4 to show that the rate-distortion function is lower bound on rate. An interesting corollary of convexity is that the rate-distortion function must be continuous (note that a discontinuous function cannot be convex). Furthermore, it shows that lowering a high distortion by a given amount is generally less expensive in terms of bit rate than lowering an already low distortion by the same amount. This is relevant when considering the simultaneous coding of several different sources.

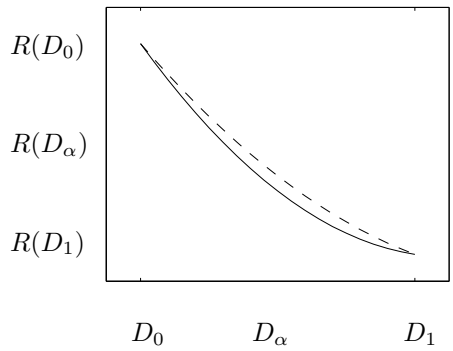

Figure 6.2: Illustration of the proof of convexity: the convex dashed curve, representing $\inf_{f_0,f_1} I_{f_\alpha}(X;\hat{X})$ bounds the solid curve, representing $R(D_\alpha)$ as a function of α. If the endpoints are identical and if $\inf_{f_0,f_1} I_{f_\alpha}(X;\hat{X})$ is below the curve connecting the endpoints, then $R(D_\alpha)$ must also be below the curve connecting the endpoints.

6.2.4 The Rate-Distortion Function is a Lower Bound on Rate

We note that deterministic mappings form a subset of statistical mappings. Thus, if we show that there is no statistical mapping that can achieve a lower rate at a given distortion than the rate-distortion function, then we have also shown that there is no deterministic mapping that can achieve a lower rate than the rate-distortion function.

Let us consider a deterministic mapping $\mathcal{Q}$, associated with a particular deterministic code, that maps a sequence of n random iid k-dimensional vectors X_i^k, forming a vector denoted as $X^{k;n}$, to the reconstruction vector $\hat{X}^{k;n} = \mathcal{Q}(X^{k;n})$ at an average distortion of $D_\mathcal{Q} = \frac{1}{n}E[d(X^{k,n}, \hat{X}^{k;n})]$. Here, we considere the case where the vectors X_i^k are segments of a stationary process. We begin with the notion that the rate $R_\mathcal{Q}$ must allow specification of the reconstruction. It is important to note that the deterministic code, in general, gives a different mean distortion for each i; that is the $E[d(X_i^k, \hat{X}_i^k)]$ are not equal for different i. The source-coding theorem (theorem 2) then tells us that

$R_{\mathcal{Q}} \geq H(\hat{X}^{k;n})$. We can then write

$$\begin{aligned}
R_{\mathcal{Q}} &\geq H(\hat{X}^{k;n}) \\
&\geq H(\hat{X}^{k;n}) - H(\hat{X}^{k;n}|X^{k;n}) \\
&= I(\hat{X}^{k;n}; X^{k;n}) \\
&= \sum_{i=1}^{n} H(X_i^k) - H(X^{k;n}|\hat{X}^{k;n}) \\
&= \sum_{i=1}^{n} H(X_i^k) - \sum_{i=1}^{n} H(X_i^k|\hat{X}^{k;n}, X_{i-1}^k, \cdots, X_1^k) \\
&\geq \sum_{i=1}^{n} \Big(H(X_i^k) - H(X_i^k|\hat{X}_i^k) \Big) \\
&= \sum_{i=1}^{n} I(\hat{X}_i^k; X_i^k) \\
&\geq \sum_{i=1}^{n} \inf_{f_{Y_i^k|X_i^k} \in B_k(D=E[d(X_i^k,\hat{X}_i^k)])} I(Y_i^k; X_i^k) \\
&= \sum_{i=1}^{n} R_k(E[d(X_i^k, \hat{X}_i^k)]) \\
&\geq nR_k(\frac{1}{n}\sum_{i=1}^{n} E[d(X_i^k, \hat{X}_i^k)]) \\
&= nR_k(D_Q) \\
&\geq nR(D_Q), \qquad (6.14)
\end{aligned}$$

where we used the convexity property of the rate-distortion function. Thus, we have shown that encoding and decoding a sequence of n k-dimensional segments of a stationary scalar process, at a mean distortion per vector of D, requires a rate of at least $R_k(D)$. Because of the ordering property of the order-k rate-distortion functions (see problem 5), no code can do better than the order-infinity rate-distortion function. After replacing $R_k(\cdot)$ with $R(\cdot)$ (removing the last step), the derivation of equation 6.14 applies to the encoding of a sequence of k-dimensional vectors.

6.2.5 The Role of Dimensionality

Before we continue with a more formal proof of the reachability of the rate-distortion function 6.2.6, it is useful to gain some insight in the effects of increasing sequence length n. While these effects do not directly prove the rate-distortion theorem function (theorem 16), they do make it more plausible.

We first show that the data density approaches uniformity in its region of support with increasing n. This suggests that a codebook with uniformly distributed reconstruction points is optimal even when encoding indices with fixed codeword length.

We then show that, given a uniform random codebook, the distance of a random point to the nearest member of the codebook tends towards being a constant with increasing n and is dependent only on the codebook point density. The previous result indicated

that both the data and a good codebook are effectively uniform. These results together are consistent with the notion that a *random* codebook can provide asymptotically optimal performance with increasing n and is relatively easy to analyse because data and codebook are uniform. This allows a back-of-the-envelope computation of the rate-distortion function. More-over it outlines the underlying principles that can be exploited in rate distortion proofs. Indeed, the standard proof of reachability of the rate-distortion function described in section 6.2.6 uses a random codebook.

For simplicity, we consider a scalar random variable X, with a density $f_X(x)$ with $x \in \mathbb{R}$. If we consider equation 6.5, then we see that our objective is equivalent to coding a vector X^n whose components are samples of the iid process X_i, with each sample x_i drawn independently from $f_X(\cdot)$.

Asymptotic Equipartition and Data Density

The realizations of the random vector X^n form a set of vectors $x^n \in \mathbb{R}^n$. Let us partition each dimension of x^n using a uniform, scalar quantizer (say with cell size δ). That is, we map $x_i \ \forall_i \in \{1, \cdots, n\}$ into a countable set of symbols that can be described by a set of quantizer indices. The source sequence (vector) x^n is mapped into a sequence of n indices that we write as the vector y^n. The random vector X^n, with a continuous alphabet, is mapped into the random vector Y^n with a discrete alphabet. Each particular y^n corresponds to a particular hypercube in the partition of $\mathbb{R}^n$ invoked by the scalar quantizers.

As n increases, the asymptotic equipartition theorem described in section 2.8 becomes applicable to the sequence of n source indices described by the vector y^n. That is, almost all realizations y^n of the random vector of indices belong to the typical set, and by selecting a sufficiently large value for n, the typical sequences have a minus log probability that is within an arbitrary value ϵ of $nH(Y)$, where $H(Y)$ is the entropy of the scalar quantizer index Y.

The probability of the sequence y^n is directly related to the probability density of the random vector X^n. The ratio of the probability of the sequence y^n and the volume of the hypercube $V = \delta^n$ asymptotically (with decreasing δ) approaches the density $f_{X^n}(\cdot)$. Under conditions on the smoothness of $f_X(\cdot)$, asymptotic equipartition leads to a density $f_{X^n}(x^n)$, $x^n \in \mathbb{R}^n$ that essentially takes only two values: a particular nonzero value or zero. This suggests that a codebook that is uniform over the range of support of $f_X(x^n)$ should be used.

Sphere Hardening

We now show that the distance from a randomly selected data point x^n to the nearest point in a random, uniform codebook becomes independent of the realization (approaches a fixed number) with increasing n. To this purpose, let us consider a hypercube of volume V that contains N randomly distributed codebook points. Around the randomly selected data point x^n we consider an n-ball (for $n = 3$ this is just the regular sphere) of volume V_b and radius r, with $V_b << V$. Our goal is to estimate the radial density $f_R(r)$ of the nearest codebook point to x^n. To compute this density, we first determine the probability that there is no codebook point inside the n-ball of radius r

and the probability that there is a codebook point in a shell of thickness dr at radius r.

The volume of the n-ball satisfies $V_b = Ar^n$, where the constant is $A = \frac{\pi^{\frac{n}{2}}}{\Gamma(\frac{n}{2}+1)}$. (Note that after an initial increase up to $n = 5$, A decreases rapidly for large n.) The volume of the shell of radial thickness dr at the surface of the n-ball is $nAr^{n-1}dr$.

The probability that the n-ball of volume V_b contains none of the N codebook points decreases rapidly with increasing volume of the n-ball:

$$\begin{aligned} P(\text{no points in ball}) &= (1 - \frac{V_b}{V})^N \\ &= \exp(\log(1 - \frac{V_b}{V})^N) \\ &= \exp(-\frac{N}{V}V_b) \\ &= \exp(-A(f_C\, r)^n) \end{aligned} \tag{6.15}$$

where $f_C = \frac{N}{V}$ is the density of the codebook points in one dimension and $f_C\, r$ can be interpreted as the codebook-density normalized radius of the n-ball.

Let us denote the volume of the shell by V_s. The probability that at least one codebook point is in the shell is then, for $V_s << V$

$$\begin{aligned} P(\text{one point in shell}) &= N\frac{V_s}{V}(1 - \frac{V_s}{V})^{N-1} \\ &\approx N\frac{V_s}{V}. \end{aligned} \tag{6.16}$$

We treat the approximate equality as an equality, which is correct for vanishing shell volume. We see that the probability that one codebook point is contained within the shell increases rapidly with the radius of the shell:

$$\begin{aligned} P(\text{one point in shell}) &= N\frac{V_s}{V} = N\frac{nAr^{n-1}dr}{V} \\ &= n\frac{N}{V}\frac{V_b}{r}dr \\ &= nA(f_C\, r)^{n-1} f_C\, dr. \end{aligned} \tag{6.17}$$

Thus, the expression $nA(f_C\, r)^{n-1}$ forms the (codebook-density normalized) radial density that there is a codebook point in the shell.

By multiplying the results of equations 6.15 and 6.17 it follows that the probability density $f_{f_C\, R}(f_C\, r)$ that the nearest codebook point is at normalized radius $f_C\, r$ satisfies

$$f_{f_C\, R}(f_C\, r) = nA\exp(-A(f_C\, r)^n)(f_C\, r)^{n-1}. \tag{6.18}$$

The density of equation 6.18 is plotted in figure 6.3 for a number of values of n. As expected, with increasing value of n the radial density function approaches a delta function. That is, for a randomly selected point in the support range of the uniformly distributed density, the distance to the nearest codebook entry converges with increasing n to being almost surely at a known distance.

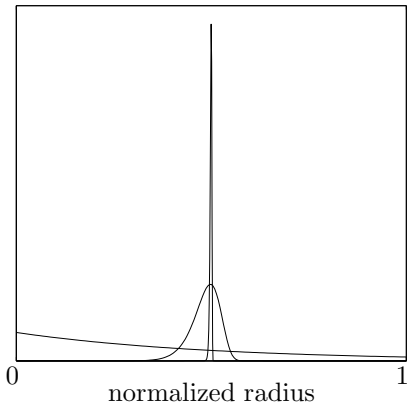

Figure 6.3: Illustration of sphere hardening: the (codebook-density normalized) radial density of the first neighbor in a random codebook as a function of radius, for vector dimension 1, 16, and 256. The vertical scales are not identical to facilitate the plot.

6.2.6 Reachability of the Rate-Distortion Function

In this subsection, we prove that the rate-distortion function can be reached with an actual code (i.e., not just with a statistical mapping) for the case of an iid scalar process. Together with the proof given in section 6.2.4 that one cannot code at lower rates (for given distortion) than the rate-distortion function indicates, this constitutes a proof of theorem 16. In the information-theory literature, rate-distortion theorems for more general circumstances have been proven. However, such theorems are complex, and fall outside the scope of this chapter.

Outline of the Proof

We start with an outline of the basic principle of the proof of the procedure that will follow below. We consider a discrete alphabet case. Let $p_{\hat{X}|X}(\hat{x}|x)$ be the conditional probability that results in the minimum (as mentioned before, for the discrete case, the infimum is a minimum) mutual information given the distortion D. We have an associated unconditional probability mass for $\hat{X}$ that is $p_{\hat{X}}(\hat{x}) = \sum_x p_{\hat{X}|X}(\hat{x}|x)p_X(x)$ and a joint probability $p_{\hat{X},X}(\hat{x}, x)$.

We start with creating random codebooks for the vector X^n, which consists of n independent components, each with the distribution p_X. (Note that n has the same meaning as in equation 6.5.) We generate, for each codebook, 2^{nR} reconstruction values $\hat{X}$ using the probability mass function $p_{\hat{X}^n}(\hat{x}^n) = \prod_{i=1}^{n} p_{\hat{X}}(\hat{x}_i)$. We thus create codebooks of dimension n with 2^{nR} entries each. Assuming that all codewords have equal length, this corresponds to a rate of R bits per sample. (Note that it follows from asymptotic equipartition (cf. section 2.8) that the usage of equal-codeword length codewords does not lead to a penalty in bit rate for sufficiently large n.) That is, we need nR bits to specify a particular entry from such a random codebook. We then evaluate the performance of these random codebooks and show that the associated distortion converges in probability to D with increasing n, proving the rate-distortion theorem.

In the proof, we make the assumption that the distortion has a finite maximum:

$$d(X^n, \hat{X}^n) \leq d_{\max}. \tag{6.19}$$

We also assume that the distortion is normalized to a per-sample basis, and is a single-letter distortion criterion (see equation 6.3).

For a random vector X^n, we split the expected distortion from the randomly selected codebooks into two contributions: one part resulting from codebooks that have a vector $\hat{X}^n$ that is within $D + \epsilon$, where ϵ is arbitrarily small, and one part resulting from codebooks that have no entry this close to the vector X^n. Let P_D be the probability that a particular random codebook has an entry that results in a distortion less than $D+\epsilon$ for a random vector X^n and let $\tilde{P}_D$ be the probability that there is no such vector ($P_D + \tilde{P}_D = 1$) in said codebook. Then we have

$$\begin{aligned} \mathrm{E}[d(X^n, C)] &< P_D(D + \epsilon) + \tilde{P}_D d_{\max} \\ &\leq D + \epsilon + \tilde{P}_D d_{\max}, \end{aligned} \tag{6.20}$$

where C denotes a random codebook consisting of 2^{nR} random entries, and $d(x^n, C)$ is the minimum of the distortions between x^n and the codebook entries. In the following, we will prove that $\tilde{P}_D$ converges to zero with increasing n, thus proving a restricted version of the rate-distortion theorem.

An Expression for $\tilde{P}_D$

First, we find the probability that a given input vector does not have an entry in a particular codebook $\mathcal{C}^n$ with distortion less than $D + \epsilon$. The probability that a random codebook vector $\hat{X}$ is *not* within $D + \epsilon$ of the vector x^n is

$$P(d(x^n, \hat{X}^n) \geq D + \epsilon) = 1 - P(d(x^n, \hat{X}^n) < D + \epsilon). \tag{6.21}$$

For an entire random codebook C, with 2^{nR} entries, the probability that none of the 2^{nR} entries is within $D + \epsilon$ of the vector x^n is

$$P(d(x^n, C) \geq D + \epsilon) = (1 - P(d(x^n, \hat{X}^n) < D + \epsilon))^{2^{nR}}. \tag{6.22}$$

So-far, we have neglected to average over the inputs x^n to obtain the probability. Performing this average, we obtain

$$\begin{aligned} \tilde{P}_D &= \sum_{x^n} p_{X^n}(x^n) P(d(x^n, C) \geq D + \epsilon) \\ &= \sum_{x^n} p_{X^n}(x^n)(1 - P(d(x^n, \hat{X}^n) < D + \epsilon))^{2^{nR}} \\ &= \sum_{x^n} p_{X^n}(x^n)(1 - \sum_{\hat{x}^n} p_{\hat{X}^n}(\hat{x}^n) K_D(x^n, \hat{x}^n))^{2^{nR}}, \end{aligned} \tag{6.23}$$

where $K_D(x^n, \hat{x}^n)$ is an indicator function:

$$K_D(x^n, \hat{x}^n) = \begin{cases} 1, & \text{if } d(x^n, \hat{x}^n) < D + \epsilon \\ 0, & \text{if } d(x^n, \hat{x}^n) \geq D + \epsilon \end{cases}. \tag{6.24}$$

Now consider the insight from section 6.2.5 that for large n the data density becomes essentially uniform and that the distance to the nearest neighbor is then essentially fixed. This leads us to expect that for high dimensionality n the indicator function is either 0 or 1 for almost all data points drawn from a distribution. We will find this to be correct.

The Distortion Typical Set

To prove the rate-distortion theorem we must now show that 6.23 converges to zero with increasing n. To this purpose, we first define the distortion-typical set. This set is simply the jointly typical set (i.e., a set of pairs of vectors) defined in section 2.8 extended with the constraint that the distortion between the vectors of each pair is close to D:

$$\begin{aligned} A_d^n(\epsilon) = \{(x^n, \hat{x}^n) \quad : \quad & |\frac{1}{n}\log(p_{X^n}(x^n)) + H(X)| < \epsilon, \\ & |\frac{1}{n}\log(p_{\hat{X}^n}(\hat{x}^n)) + H(\hat{X})| < \epsilon, \\ & |\frac{1}{n}\log(p_{X^n\hat{X}^n}(x^n, \hat{x}^n)) + H(X, \hat{X})| < \epsilon \\ & |d(x^n, \hat{x}^n) - D| < \epsilon\}. \end{aligned} \tag{6.25}$$

We furthermore define a new indicator function, which is related to the distortion-typical set:

$$K_{DT}(x^n, \hat{x}^n) = \begin{cases} 1 & \text{if } (x^n, \hat{x}^n) \in A_d^n(\epsilon) \\ 0 & \text{if } (x^n, \hat{x}^n) \notin A_d^n(\epsilon) \end{cases} . \tag{6.26}$$

Exploiting the Distortion-Typical Set for the Bound

We first note that

$$\begin{aligned} \sum_{\hat{x}^n} p_{\hat{X}^n}(\hat{x}^n) K_D(x^n, \hat{x}^n) \quad &\geq \quad \sum_{\hat{x}^n} p_{\hat{X}^n}(\hat{x}^n) K_{DT}(x^n, \hat{x}^n) \\ &\geq \quad \sum_{\hat{x}^n} p_{\hat{X}^n|X^n}(\hat{x}^n|x^n) 2^{-nI(X;\hat{X})-3n\epsilon} K_{DT}(x^n, \hat{x}^n), \end{aligned} \tag{6.27}$$

where we used in the first step that there are additional constraints on the vector $\hat{X}^n$ in the distortion-typical set, and where we used inequality 2.66 to write the second step.

Inserting inequality 6.27 in equation 6.23 leads to

$$\begin{aligned} \tilde{P}_D \quad &= \quad \sum_{x^n} p_{X^n}(x^n)(1 - \sum_{\hat{x}^n} p_{\hat{X}^n}(\hat{x}^n) K_D(x^n, \hat{x}^n))^{2^{nR}} \\ &\leq \quad \sum_{x^n} p_{X^n}(x^n)(1 - \sum_{\hat{x}^n} p_{\hat{X}^n|X^n}(\hat{x}^n|x^n) 2^{-nI(X;\hat{X})-3n\epsilon} K_{DT}(x^n, \hat{x}^n))^{2^{nR}} \end{aligned} \tag{6.28}$$

It now becomes clear why we use the distortion-typical set: we see that both $I(X;\hat{X})$ and R appear as arguments of exponentials. To proceed, we use the fact (cf. problem 14)

$$(1 - ab)^m \leq 1 - a + \mathrm{e}^{-bm} \tag{6.29}$$

to write inequality 6.28 as

$$\begin{aligned}\tilde{P}_D &\leq \sum_{x^n} p_{X^n}(x^n)(1-\sum_{\hat{x}^n} p_{\hat{X}^n|X^n}(\hat{x}^n|x^n)K_{DT}(x^n,\hat{x}^n)+\exp(-2^{nR}2^{-nI(X;\hat{X})-3n\epsilon})) \\ &= 1-\sum_{x^n}\sum_{\hat{x}^n} p_{\hat{X}^n,X^n}(\hat{x}^n,x^n)K_{DT}(x^n,\hat{x}^n)+\exp(-2^{nR-nI(X;\hat{X})-3n\epsilon}). \qquad (6.30)\end{aligned}$$

The exponential term in equation 6.30 approaches zero rapidly with increasing n if $R > I(X;\hat{X}) + 3\epsilon$, i.e., if *the rate-distortion theorem holds.* The second term in equation 6.30 forms the probability that a pair $(x^n, \hat{x}^n)$ drawn from the probability mass function $p_{X^n\hat{X}^n}(x^n, \hat{x}^n)$ is within the distortion-typical set. This probability approaches unity with increasing n. To show this, the argument used in the description of the joint typical set in section 2.8 can be used. We quickly reiterate the reasoning. From the weak law of large numbers it follows that there is a n such that each of the conditions in the probability in 6.25 individually is not satisfied is less than, say, δ. The probability that all conditions are satisfied must then be more than $1 - 4\delta$. Since we can set δ to any number in $(0, 1]$, we have shown that the second term approaches unity with increasing n, and that, therefore, $\tilde{P}_D$ approaches zero with increasing n.

6.3 The Distortion-Rate Function

As its name suggests, the rate-distortion function displays the rate as a function of distortion. Often the distortion-rate function, $D(R)$, is used instead. The **distortion-rate function** for a process is defined as the infimum distortion over all codes of given rate under the conditions of stationarity, ergodicity, and a bounded single-letter criterion.

To characterize the distortion-rate function, one can define the set of probability densities $G_k(R)$:

$$G_k(R) = \{f_{\hat{X}^k|X^k} : I(X^k;\hat{X}^k) \leq R\}. \qquad (6.31)$$

The **order-k distortion-rate function** is then defined as

$$D_k(R) = \inf_{f_{\hat{X}^k|X^k} \in G_k(R)} E[d(X^k,\hat{X}^k)]. \qquad (6.32)$$

Using the same reasoning as for the rate-distortion function, we see that the distortion-rate function is nonincreasing.

Since the sets $G_k(X^k)$ become less restrictive with increasing k, it is clear that the order-k distortion-rate functions form an ordered set $D_k(R) \geq D_{k+1}(R)$. One can then prove that the distortion-rate function is the order-infinity distortion-rate function. We will not pursue this track, but instead show simply that the distortion-rate function is the inverse of the rate-distortion function.

We now show by contradiction that the distortion-rate function equals the inverse of the rate-distortion function, whenever the rate-distortion function is strictly decreasing. Consider a rate ρ and a conditional probability density $f_\beta(x^k, \hat{x}^k)$ that corresponds to a point on the rate-distortion curve (δ, ρ). Then we have that $\rho = R_k(\delta)$. Now let us assume that the pair (ρ, δ) is not on the distortion-rate curve, i.e., that $\delta > D_k(\rho)$. Continuing from the distortion-rate perspective, this implies that there is a rate $r < \rho$

that has distortion $\delta = D_k(r)$. However, let us now take the rate-distortion perspective: from the fact that the rate-distortion function is strictly decreasing it follows that if $\delta > D_k(\rho)$ and δ is on the rate-distortion function, then the minimum rate required for a distortion $D_k(\rho)$ must be higher than ρ, contradicting our earlier statement that $r < \rho$. This contradiction is resolved only if the rate-distortion function and the distortion-rate function are each others inverse. Thus, we have proven the following theorem:

Theorem 19 *The rate-distortion function $R(D)$ and the distortion-rate functions $D(R)$ are inverses whenever $R(D)$ is strictly decreasing.*

Note that, since the rate-distortion function is convex, the distortion-rate function is also convex. This implies that, at the rate-distortion bound, bits are more efficient in decreasing distortion at low rate than at high rate. Thus, if we quantize two identical signals with separate encodings, but under an overall rate constraint, it is most efficient to give them an identical rate allocation. We will return to simultaneous encoding in section 6.6.2, where we discuss reverse water filling.

Example 6.4: Squared-error distortion-rate for Gaussian density
It is straightforward to obtain the distortion-rate function for a Gaussian variable X with variance σ_X^2 and with a squared-error criterion. By inverting the rate-distortion function in example 6.3 we obtain:

$$D(R) = \sigma_X^2 \mathrm{e}^{-2R}, \quad R \geq 0.$$

This relation clearly shows that with increasing bit rate, the distortion decreases less rapidly.

6.4 The Shannon Lower Bound

The Shannon lower bound is a lower bound for the rate-distortion function. For Gaussian sources with the commonly used squared-error criterion, the bound is tight, i.e., for Gaussian sources with this criterion the Shannon lower bound and the rate-distortion function are identical. We first derive a common form of the bound, which applies to difference-based criteria, and then describe conditions under which the bound and the rate-distortion function coincide.

6.4.1 Derivation of the Shannon Lower Bound

We consider a vector X^k, a reconstruction $\hat{X}^k$, and a reconstruction error W^k. The Shannon lower bound for a difference based criterion can be found as follows:

$$\begin{aligned}
R(D) &= \inf_{f_{\hat{X}^k|X^k} \in B_k(D)} I(X^k; \hat{X}^k) \\
&= \inf_{f_{\hat{X}^k|X^k} \in B_k(D)} (h(X^k) - h(X^k|\hat{X}^k)) \\
&= h(X^k) - \sup_{f_{\hat{X}^k|X^k} \in B_k(D)} h(X^k|\hat{X}^k) \\
&= h(X^k) - \sup_{f_{\hat{X}^k|X^k} \in B_k(D)} h(X^k - \hat{X}^k|\hat{X}^k) \\
&\geq h(X^k) - \sup_{f_{\hat{X}^k|X^k} \in B_k(D)} h(X^k - \hat{X}^k) \\
&= h(X^k) - \sup_{\{f_{W^k} : \int f_{W^k}(w^k) d(w^k) dw^k \leq D\}} h(W^k), \qquad (6.33)
\end{aligned}$$

where we assumed a difference criterion, i.e., $d(x^k, \hat{x}^k) = d(x^k - \hat{x}^k)$, and where we used that

$$\begin{aligned}
&\sup_{f_{\hat{X}^k|X^k} \in B_k(D)} h(X^k - \hat{X}^k) \\
&= \sup_{\{f_{\hat{X}^k|X^k} : \int\int f_{\hat{X}^k|X^k}(\hat{x}^k|x^k) f_{X^k}(x^k) d(\hat{x}^k, x^k) d\hat{x}^k dx^k \leq D\}} h(X^k - \hat{X}^k) \\
&= \sup_{\{f_{\hat{X}^k - X^k} : \int\int f_{\hat{X}^k - X^k}(\hat{x}^k - x^k) f_{X^k}(x^k) d(\hat{x}^k - x^k) d\hat{x}^k dx^k \leq D\}} h(X^k - \hat{X}^k) \\
&= \sup_{\{f_{W^k} : \int\int f_{W^k}(w^k) f_{X^k}(x^k) d(w^k) dw^k dx^k \leq D\}} h(W^k) \\
&= \sup_{\{f_{W^k} : \int f_{W^k}(w^k) d(w^k) dw^k \leq D\}} h(W^k) \qquad (6.34)
\end{aligned}$$

We have thus proven the following theorem:

Theorem 20 *The Shannon lower bound,*

$$R_{\mathrm{SLB}}(D) = h(X^k) - \sup_{\{f_{W^k} : \int f_{W^k}(w^k) d(w^k) dw^k \leq D\}} h(W^k) \qquad (6.35)$$

is a lower bound on the rate-distortion function, $R_{\mathrm{SLB}}(D) \leq R(D)$.

For a stationary process the equivalent theorem is

Theorem 21 *The order-k Shannon lower bound,*

$$R_{k,\mathrm{SLB}}(D) = \frac{1}{k} h(X^k) - \frac{1}{k} \sup_{\{f_{W^k} : \int f_{W^k}(w^k) d(w^k) dw^k \leq kD\}} h(W^k) \qquad (6.36)$$

is a lower bound on the (order-k) rate-distortion function, $R_{k,\mathrm{SLB}}(D) \leq R_k(D)$.

Because of its significance, we work out the Shannon lower bound for the specific case of the squared-error distortion measure. From theorem 14 we know that the vector of maximum differential entropy for given variance has a Gaussian density with constant-diagonal covariance matrix. Thus, in the squared-error case, the error vector W in the term $\sup_{\{f_{W^k}:\int f_{W^k} d(w^k)dw^k \leq D\}} h(W^k)$ is a Gaussian vector with a diagonal covariance matrix and variance D. The differential entropy for each component of this vector is $\frac{1}{2}\log(2\pi \mathrm{e}\frac{D}{k})$. The Shannon lower bound simplifies for the squared-error distortion measure to

$$R_{k,\mathrm{SLB}}(D) = \frac{1}{k}h(X^k) - \frac{1}{2}\log(2\pi \mathrm{e}\frac{D}{k}). \tag{6.37}$$

A major advantage of the Shannon lower bound is that it is simple to compute. This is illustrated by the next example.

Example 6.5: $R(D)$ bound for Laplace density, squared-error distortion From example 3.2 we know that the differential entropy of a random variable with Laplace density $f_X(x) = \frac{a}{2}e^{-a|x|}$ is $h(X) = \log(\frac{2\mathrm{e}}{a})$. The Shannon lower bound for this variable and a squared error criterion is thus

$$R_{1,\mathrm{SLB}}(D) = \log(\frac{2\mathrm{e}}{a}) - \frac{1}{2}\log(2\pi \mathrm{e}D) = \frac{1}{2}\log(\frac{2\mathrm{e}}{\pi a^2 D}).$$

6.4.2 When is the Shannon Lower Bound Tight?

It is seen from the derivation in equation 6.34 that the Shannon lower bound for a difference distortion measure becomes tight when $h(X^k - \hat{X}^k|\hat{X}^k) = h(X^k - \hat{X}^k)$, i.e., for the case where the mutual information $I(X^k - \hat{X}^k; \hat{X}^k) = 0$. Conversely, if the Shannon lower bound for the difference criterion coincides with the rate-distortion function, then it must be possible to create a variable with the statistics of X by adding the independent variables $\hat{X}$ and V, where V has maximum differential entropy given the distortion measure $D(V) \equiv \int f_V(v)d(v)dv$. This is the same argument that we already saw in example 6.3. It implies that if it is possible to construct a **backward channel**, as in figure 6.4, with the noise variable V having maximum differential entropy, then we can reach the Shannon lower bound with a real code; the Shannon lower bound and the rate-distortion bound are then identical. We summarize this discussion in a theorem:

Theorem 22 *The Shannon lower bound is tight for a difference distortion measure $D(\cdot)$ if, and only if, the variable X can be constructed from the summation of two independent variables $\hat{X}$ and V, where V has maximum differential entropy given $D(V)$.*

It is often straightforward to see whether the Shannon bound is tight. Let us consider the one-dimensional case. If a continuous-alphabet variable X can be constructed from the addition of two independent continuous-alphabet variables $\hat{X}$ and V, then its density is the convolution of the densities of $\hat{X}$ and V. Let $\mathcal{F}$ denote the (continuous-time) Fourier transform, let $\mathcal{G}_1 = \{f_W : \int f_W(w)d(w)dw \leq D\}$, and let V be a variable with $f_V \in \mathcal{G}_1$ such that

$$h(V) \geq h(W), \ \ \forall_{f_W \in \mathcal{G}_1}. \tag{6.38}$$

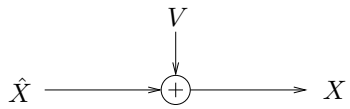

Figure 6.4: The backward channel. The input is the reconstruction $\hat{X}$ and the output the original variable X and V is selected to maximize $h(V)$.

Assuming the densities have a Fourier transform, the convolution of the density can then be written as $\mathcal{F}f_X(\omega) = \mathcal{F}f_{\hat{X}}(\omega)\mathcal{F}f_V(\omega)$ and we have

$$\mathcal{F}f_{\hat{X}}(\omega) = \mathcal{F}f_X(\omega)/\mathcal{F}f_V(\omega). \tag{6.39}$$

If the inverse Fourier transform of $\mathcal{F}f_{\hat{X}}(\omega)$ is a density (i.e., nonnegative[1]), then the Shannon lower bound is tight.

Example 6.6: $R(D)$ for Gaussian variable and squared error
Armed with the results of this section, we revisit example 6.3. The lower bound of equation 6.37 and the fact that the rate never should be less than zero show that the Shannon bound for a Gaussian variable with variance σ^2 is

$$R_{\mathrm{SLB}}(D) = \max(0, \frac{1}{2}\log(\frac{\sigma^2}{D})).$$

Next, we check the possibility of creating a signal with the statistics of X by adding the independent variables V (Gaussian with variance D) and $\hat{X}$ (free to select). As we saw before, selecting $\hat{X}$ to be Gaussian with variance $\sigma^2 - D$ will satisfy these requirements. We thus have $R(D) = R_{\mathrm{SLB}}(D)$ for this case. This result is also obtained in a straightforward manner if equation 6.39 is employed. While this example shows that the Shannon lower bound is always tight for a scalar Gaussian variable and a squared error criterion, this is not always true for a Gaussian variable and a squared error, as we will see in section 6.6.2.

Example 6.7: Shannon bound for rectangular density and squared error
In this example, we use the differential entropy computed in example 3.1 for a density uniform within $[0, a)$ and zero outside this interval. From equation 6.37 we see that the Shannon lower bound for the variable X with rectangular density is

$$\begin{aligned} R_{\mathrm{SLB}}(D) &= h(X) - \frac{1}{2}\log(2\pi\mathrm{e}D) \\ &= \log(a) - \frac{1}{2}\log(2\pi\mathrm{e}D) = \frac{1}{2}\log(\frac{a^2}{2\pi\mathrm{e}D}). \end{aligned}$$

Now let us see if this bound is tight. If so, then we must be able to construct X from adding the Gaussian V of variance D to an independent $\hat{X}$ to obtain X. It is immediately seen that this is impossible, since the support of V exceeds that of X. (The convolution of two densities, in this case f_V and $f_{\hat{X}}$, always has a support at least as large as either density.) The bound is thus not tight.

[1] It is easily shown that if $f_{\hat{X}}$ is nonnegative, and if f_X and f_V integrate to unity, then $f_{\hat{X}}$ must also integrate to unity.

Example 6.8: Shannon bound for Laplace density and absolute error
We consider again the Laplace density of example 3.2, $f_X(x) = \frac{a}{2}e^{-a|x|}$. From that example we know that its differential entropy is $\log(\frac{2\mathrm{e}}{a})$. We must now find the density of the variable V. That is, we must find the $f_V(v)$ that maximizes

$$h(V) = -\int f_V(v)\log(f_V(v))dv$$

under the constraint that

$$D = \int f_V(v)|v|dv.$$

and that $f_V(v)$ is a density, i.e.,

$$\int f_V(v)dv = 1.$$

We also should impose the constraint that $f_V(v) \geq 0$, but, as we will see, the solution satisfies this constraint without imposing it. We assume symmetry and optimize the criterion

$$\eta = \int_0^\infty (-f_V(v)\log(f_V(v)) + \lambda f_V(v)v)\,dv,$$

where λ is the Lagrange multiplier. The Euler-Lagrange equation is

$$-\log(f_V(v)) - 1 + \lambda v = 0.$$

Using the constraint this results in

$$f_V(v) = \frac{1}{2D}e^{-\frac{|v|}{D}},$$

which is, indeed, nonnegative. (This density was found earlier in example 4.6.) The Shannon lower bound is now

$$R_{\mathrm{SLB}}(D) = \log(\frac{2\mathrm{e}}{a}) - \log(2\mathrm{e}D) = -\log(aD).$$

Next, we determine if this bound is tight. We see that

$$\begin{aligned}\mathcal{F}f_V(\omega) &= \int \frac{1}{2D}e^{-\frac{|x|}{D}}\mathrm{e}^{-j\omega x}dx \\ &= \frac{1}{1+D^2\omega^2}.\end{aligned}$$

Thus, equation 6.39 becomes

$$\mathcal{F}f_{\hat{X}}(\omega) = (1+D^2\omega^2)\mathcal{F}f_X(\omega),$$

which corresponds to

$$\begin{aligned}f_{\hat{X}}(x) &= f_X(x) - D^2\frac{d^2f_X(x)}{dx^2} \\ &= (1-a^2D^2)\frac{a}{2}e^{-a|x|} + a^2D^2\delta(x),\end{aligned}$$

where $\delta(x)$ is a Dirac delta function. It is easily verified that this function is nonnegative and, therefore, a density function for $D \leq 1/a$. In other words, the backward channel exists and the Shannon lower bound is tight:

$$R(D) = R_{\mathrm{SLB}}(D) = -\log(aD), \quad D \leq 1/a.$$

Example 6.9: Uniformly distributed variable with threshold criterion
Consider a random variable X with a density, $f_X(x)$, which is uniform in the interval $[0, 2)$ and zero elsewhere. The distortion criterion for the quantization error $w = x - \mathcal{Q}(x)$ is shown in figure 6.5[2]. The objective is to compute the Shannon lower bound for this variable and to see whether it is tight. We first

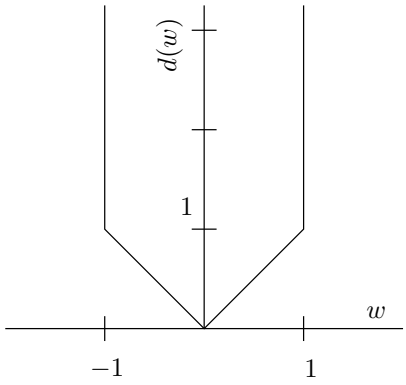

Figure 6.5: The distortion criterion for example 6.9.

compute the density $f_W(w)$ such that $h(W)$ is maximized under the condition that $\mathrm{E}[d(W)] = \int d(w) f_W(w) dw \leq D$. In this computation, we exploit the symmetry of the problem and the fact that $f_W(w) = 0$ for $|w| \geq 1$. The constraints are then $\int_0^1 f_W(w) dw = \frac{1}{2}$, $\int_0^1 w f_W(w) \leq \frac{D}{2}$, and that f_W is nonnegative. Ignoring as usual the latter constraint, the extended criterion becomes

$$\eta = \int_0^1 (f_W(w) \log(f_W(w)) + \lambda' f_W(w) + \mu w f_W(w)) dw,$$

where λ' and μ are Lagrange multipliers. The Euler-Lagrange equation is $\log(f_W(w)) + \lambda' + \mu w = 0$, which results in

$$f_W(w) = \lambda e^{-\mu|w|}, \quad w \in (-1, 1),$$

where $\lambda = \exp(-\lambda')$, which implies that f_W is indeed nonnegative. It remains to determine the constants μ and λ. It is clear that for small D we must have $\mu > 0$ (regions of high density then correspond to regions of low distortion). The case $\mu = 0$ corresponds to a flat distribution and, thus, to the maximum differential entropy. We never allow $\mu < 0$, which corresponds to a situation where $f_W(w)$ is not flat and D is larger than that for maximum differential entropy. The computations below are, therefore, only relevant for $\mu \geq 0$, otherwise we simply set $\mu = 0$.

From $\frac{1}{2} = \int_0^1 f_W(w) dw = \frac{\lambda}{\mu}(1 - \mathrm{e}^{-\mu})$ we obtain $\lambda = \frac{1}{2} \frac{\mu}{1 - e^{-\mu}}$. Furthermore, assuming that the inequality constraint is an equality constraint, we have that $\frac{D}{2} = \int_0^1 w f_W(w) dw = \frac{\lambda}{\mu}(\frac{1}{\mu}(1 - e^{-\mu}) - e^{-\mu})$. Combining these results gives

$$\begin{aligned} D &= \frac{1}{\mu} - \frac{1}{e^{\mu} - 1} \\ &= \frac{1}{2} + \frac{1}{\mu} - \frac{1}{2} \frac{e^{\frac{\mu}{2}} + e^{\frac{-\mu}{2}}}{e^{\frac{\mu}{2}} - e^{\frac{-\mu}{2}}}, \end{aligned}$$

[2]By having a distortion criterion that becomes infinite, the case of example 6.9 contradicts the always-finite condition used in the proof of the reachability of the rate-distortion function. Strictly spoken, this condition is also violated for the squared-error criterion. By using as distortion $d_{\mathrm{b}}(w) = \inf(d(w), L)$ with $L \in \mathbb{R}$ large but finite, this problem is eliminated.

where the latter notation shows that $D(\mu) - \frac{1}{2}$ is an odd function of μ.

Next, we compute the Shannon lower bound. We have that $h(X) = -\int_0^2 \frac{1}{2}\log(\frac{1}{2})dx = \log(\frac{1}{2})$. We also have that

$$\begin{aligned} h(W) &= -2\int_0^1 \lambda \mathrm{e}^{-\mu w}\log(\lambda \mathrm{e}^{-\mu w})dw \\ &= -2\lambda\log(\lambda)\int_0^1 \mathrm{e}^{-\mu w}dw + 2\lambda\mu\int_0^1 w\mathrm{e}^{-\mu w}dw \\ &= 2\frac{\lambda}{\mu}\log(\lambda)(\mathrm{e}^{-\mu}-1) + 2\frac{\lambda}{\mu}(\mathrm{e}^{-\mu}(-\mu-1)+1) \\ &= 2\frac{\lambda}{\mu}(\log(\lambda)-1)(\mathrm{e}^{-\mu}-1) - 2\lambda\mathrm{e}^{-\mu}, \end{aligned}$$

which can be rewritten as

$$h(W) = 1 - \log(\frac{1}{2}\frac{\mu}{1-\mathrm{e}^{-\mu}}) - \frac{\mu\mathrm{e}^{-\mu}}{1-\mathrm{e}^{-\mu}}.$$

Note that, as expected, $h(W)$ approaches one bit as μ approaches zero. Thus, the Shannon lower bound is, in nats, as a function of μ,

$$\begin{aligned} R_{\mathrm{SLB}}(\mu) &= \log(2) - 1 + \log(\frac{1}{2}\frac{\mu}{1-\mathrm{e}^{-\mu}}) + \frac{\mu\mathrm{e}^{-\mu}}{1-\mathrm{e}^{-\mu}} \\ &= -1 + \log(\frac{\mu}{\mathrm{e}^{\frac{\mu}{2}} - \mathrm{e}^{-\frac{\mu}{2}}}) + \frac{\mu}{2}\frac{\mathrm{e}^{\frac{\mu}{2}} + \mathrm{e}^{-\frac{\mu}{2}}}{\mathrm{e}^{\frac{\mu}{2}} - \mathrm{e}^{-\frac{\mu}{2}}}, \end{aligned}$$

where the notation was selected to show the even symmetry around $\mu = 0$. By selecting a set of values of μ and then evaluating the associated D and $R_{\mathrm{SLB}}(D)$ (and converting to bits) we obtain the rate-distortion curve shown in figure 6.6.

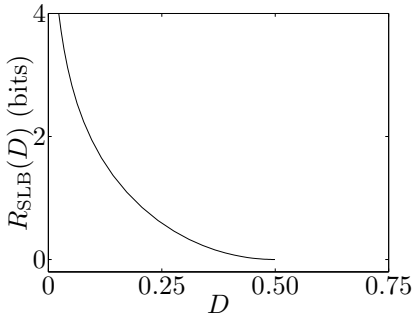

Figure 6.6: The Shannon lower bound of example 6.9.

Let us see if this Shannon lower bound is tight. If a backward channel exists, then we should have that $X = \hat{X} + W$ and, thus, $f_X(x) = f_{\hat{X}}(x) * f_W(x)$ with $\hat{X}$ independent. But we know that f_W has support (is nonzero) only within $(-1, 1)$. The support of f_X is the same as that of f_W, implying that the support of $f_{\hat{X}}$ vanishes. Since f_X and f_W are not identical, $f_{\hat{X}}$ cannot be a Dirac delta function and we obtain a contradiction. Thus, the Shannon lower bound is not tight.

Finally, we re-examine what happens if we omit the inequality in the condition $\mathrm{E}[d(W)] = \int d(w)f_W(w)dw \leq D$ imposed for the Shannon lower bound. The

odd symmetry of $D(\mu) - \frac{1}{2}$ and the even symmetry of $R_{\mathrm{SLB}}(\mu)$ leads to a rate-distortion curve that is symmetric around $D = 0.5$. That is the rate increases with increasing distortion for $D > 0.5$. This corresponds to the density $f_W(w)$ having higher values in regions of high distortion.

6.5 Finding $R(D)$ by Constrained Optimization

The rate-distortion theorem (theorem 16) in effect states the rate-distortion function as the solution to a constrained-optimization problem. In this section, we use conventional optimization methods to try to find the rate-distortion function. In the first of the following two subsections a set of equations is found, which, in principle, can be used to solve for the conditional probability density function $f_{\hat{X}^k|X^k}(\hat{x}^k|x^k)$ (or probability mass function) that determines the rate-distortion function. These equations are nonlinear and are thus difficult to solve analytically. However, they can be used to obtain an algorithm to determine the rate-distortion function numerically. This algorithm, the Blahut algorithm, is the topic of the second subsection.

6.5.1 The Optimal Conditional Probability Mass Function

We recall that the rate-distortion function is defined by the mutual information under the constraint that the conditional probability density $f_{\hat{X}^k|X^k}(\hat{x}^k|x^k)$ falls into the set B_k defined by equation 6.7. A formal solution to this problem can be found by the method of Lagrange multipliers and variational calculus. To simplify the manipulations, we use the one-dimensional case and probability mass functions rather than density functions. (This anticipates the numerical procedures in the next section, which are inherently discrete.) To make the derivations more readable, we will use the following simplified notation: $p(x) = p_X(x)$, $p(\hat{x}|x) = p_{\hat{X}|X}(\hat{x}|x)$, and $p(\hat{x}) = p_{\hat{X}}(\hat{x})$.

The optimization problem for finding the rate-distortion function is then to find the $p(\hat{x}|x)$ that minimizes

$$I(\hat{X}; X) = \sum_{x,\hat{x}} p(\hat{x}|x)p(x) \log(\frac{p(\hat{x}|x)}{p(\hat{x})}) \tag{6.40}$$

under the constraints

$$\sum_{x,\hat{x}} p(\hat{x}|x)p(x)d(x,\hat{x}) = D, \tag{6.41}$$

$$\sum_{\hat{x}} p(\hat{x}|x) = 1, \tag{6.42}$$

$$p(\hat{x}|x) \geq 0. \tag{6.43}$$

In constraint 6.41 we have, in effect, assumed that the distortion is exactly D, which is correct if the rate-distortion function is strictly decreasing. Constraint 6.42 results from the fact that $p(\hat{x}|x)$ is a probability density. Finally, constraint 6.43 states that all probabilities $p(\hat{x}|x)$ must be nonnegative.

In the following derivation we include constraint 6.43 to show more formally how to deal with such an inequality constraint. However, for this problem it is also possible to

take a simpler approach: assume that the inequality constraint is not violated, solve the problem without the inequality constraint and confirm at the end that the constraint is not violated.

Using the Lagrange multiplier method, the criterion can be rewritten as

$$\eta = \sum_{x,\hat{x}} \left(p(\hat{x}|x)p(x)\log(\frac{p(\hat{x}|x)}{p(\hat{x})}) + \lambda p(\hat{x}|x)p(x)d(x,\hat{x}) + \mu(x)p(\hat{x}|x) + \nu(\hat{x},x)p(\hat{x}|x) \right), \tag{6.44}$$

where it is understood that

$$\nu(\hat{x},x) = 0, \qquad p(\hat{x}|x) > 0, \tag{6.45}$$
$$\nu(\hat{x},x) \leq 0, \qquad p(\hat{x}|x) = 0, \tag{6.46}$$

to account for the inequality condition. For the argument of the logarithm it is important to realize that $p(\hat{x}) = \sum_x p(\hat{x}|x)p(x)$. We differentiate the extended criterion of equation 6.44 towards $p(\hat{x}|x)$ to obtain (cf. problem 6)

$$\frac{d\eta}{dp(\hat{x}|x)} = p(x)\log(\frac{p(\hat{x}|x)}{p(\hat{x})}) + \lambda p(x)d(x,\hat{x}) + \mu(x) + \nu(\hat{x},x). \tag{6.47}$$

We note that this derivative is increasing as a function of $p(\hat{x}|x)$ and that, therefore, η is convex in $p(\hat{x}|x)$ and has one minimum. It is, furthermore, clear that the nonnegativity constraint is nonessential if omitting this constraint results in a solution that satisfies it; we set $\nu(\hat{x},x) = 0$ in this situation. If the minimum is not located in $p(\hat{x}|x) \geq 0$, then we modify the criterion such that a minimum is located on the boundary $p(\hat{x}|x) = 0$. Note that the fact that the minimum is located at $p(\hat{x}|x) < 0$ if $\nu(\hat{x},x) = 0$ implies that $\frac{d\eta}{dp(\hat{x}|x)} > 0$ at $p(\hat{x}|x) = 0$. Thus, we have to set $\nu(\hat{x},x) < 0$ to obtain $\frac{d\eta}{dp(\hat{x}|x)} = 0$ at $p(\hat{x},x) = 0$.

Setting the expression of equation 6.47 to zero we find a condition for the minimum. It is convenient to rewrite this condition in a different format:

$$p(\hat{x}|x) = p(\hat{x})\mathrm{e}^{-(\lambda d(x,\hat{x}) + \frac{\mu(x)}{p(x)} + \frac{\nu(\hat{x},x)}{p(x)})}. \tag{6.48}$$

If we sum equation 6.48 over $\hat{x}$, the left-hand side becomes unity. This implies that we always must have that $\mu(x) < \infty$. Furthermore, if $p(\hat{x}|x) = 0$, then, according to condition 6.46, $\nu(\hat{x},x) \leq 0 < \infty$. This means that $p(\hat{x}|x) = 0$ if and only if $p(\hat{x}) = 0$. That is, if $p(\hat{x}|x) = 0$ for any x, then $p(\hat{x}|x) = 0$ for all x. Considering only the relevant case where $p(\hat{x}) > 0$, equation 6.48 can be simplified to

$$p(\hat{x}|x) = p(\hat{x})\mathrm{e}^{-(\lambda d(x,\hat{x}) + \frac{\mu(x)}{p(x)})}. \tag{6.49}$$

We select $\mu(x)$ in equation 6.49 to satisfy condition 6.42:

$$p_{\hat{X}|X}(\hat{x}|x) = \frac{p_{\hat{X}}(\hat{x})\mathrm{e}^{-\lambda d(x,\hat{x})}}{\sum_{\hat{x}'} p_{\hat{X}}(\hat{x}')\mathrm{e}^{-\lambda d(x,\hat{x}')}}, \tag{6.50}$$

where, to facilitate future reference, we have returned to the complete notation for the probability mass functions. Equation 6.50 does not yet provide a solution, since the marginal probability mass function $p_{\hat{X}}(\hat{x})$ is not known.

We can formally solve for $p_{\hat{X}}(\hat{x})$ as follows. Again only considering the relevant cases $p_{\hat{X}}(\hat{x}) > 0$, we multiply equation 6.50 by $p_X(x)/p_{\hat{X}}(\hat{x})$ and sum over x to obtain

$$1 = \sum_x \frac{p_X(x)\mathrm{e}^{-\lambda d(x,\hat{x})}}{\sum_{\hat{x}'} p_{\hat{X}}(\hat{x}')\mathrm{e}^{-\lambda d(x,\hat{x}')}}. \tag{6.51}$$

Equation 6.51 exists for all values $\hat{x}$ of the alphabet of $\hat{X}$ that have nonzero probability, $p_{\hat{X}}(\hat{x}) > 0$. These equations, together with the constraint 6.41 can, in principle, be solved for the values of $p_{\hat{X}}(\hat{x})$ that are nonzero and for λ. The desired solution for $p_{\hat{X}|X}(\hat{x}|x)$ is then obtained from equation 6.50, and this can be used to obtain the extremum of the mutual information, which is the rate-distortion function.

6.5.2 The Blahut Algorithm

The method in the previous section did not provide a practical solution. In general, analytic procedures often fail in the computation of the rate-distortion function. The Blahut algorithm [25] is an iterative method to find a numerical solution for the rate-distortion function. As in the case of the related expectation-maximization algorithm of section 4.3.4, the underlying principle is to reformulate the optimization over one variable as an optimization over two variables, where the optimal value for each variable can be found if the other is kept constant. This leads to an iterative solution procedure where we alternatingly optimize the two variables.

We start with rewriting the result of theorem 16 in a more convenient form:

$$\begin{aligned} R(D) &= \min_{p_{\hat{X}|X} \in B_1(D)} I(X;\hat{X}) \\ &= \min_{p_{\hat{X}|X} \in B_1(D)} H(p_{\hat{X}|X}p_X \| p_X p_{\hat{X}}) \end{aligned} \tag{6.52}$$

It will be shown below that $p_{\hat{X}}$ is the marginal distribution p_Y that minimizes the relative entropy $H(p_{X\hat{X}}\|p_X p_Y)$. That is

$$q_{\hat{X},\mathrm{opt}}(\hat{x}) = p_{\hat{X}}(\hat{x}) = \sum_x p_{\hat{X}|X}(\hat{x}|x)p_X(x). \tag{6.53}$$

We see that the optimal distribution p_Y can be expressed analytically if $p_{\hat{X}|X}$ is assumed known. This means that the rate-distortion function of equation 6.52 can be rewritten in the desired format with a double minimization:

$$R(D) = \min_{p_{\hat{X}|X} \in B_1(D)} \min_{p_Y} H(p_{\hat{X}|X}p_X \| p_X p_Y). \tag{6.54}$$

The minimization over $p_{\hat{X}|X}$ can also be expressed analytically, if we assume that p_Y is known. Although the derivation must be modified from that given in section 6.5.1 if we assume that p_Y is known, the solution is unaffected and is given by equation 6.50. Thus, we intend to find the rate-distortion function by alternatingly minimizing $H(p_{\hat{X}|X}p_X\|p_X p_Y)$ for the discrete functions p_Y and $p_{\hat{X}|X}$. Since relative entropy is nonnegative, this iterative process must have a limiting value and it can be shown that this limiting value is $R(D)$ (in contrast with the expectation-maximization algorithm, the algorithm cannot get stuck in local minima). Equation 6.53 solves the optimization

Table 6.1: The Blahut algorithm for finding the rate-distortion function.

1. initialize the function $p_{\hat{X}}$ and select a value for λ,
2. compute the function $p_{\hat{X}\|X}$ using 6.50,
3. compute the function p_Y using 6.53,
4. check whether iterations have converged; go to 2 if they have not,
5. compute $R(D)$ by evaluating $I(X;\hat{X})$ and compute D using equation 6.41.

for p_Y and equation 6.50 minimizes the expression for $p_{\hat{X}|X}$. The value for λ in equation 6.50 determines the point on the rate-distortion curve that is computed. From equation 6.44 it is seen that, with increasing λ, the distortion gains in importance in the criterion. This means that the distortion decreases and the rate increases with increasing λ. The complete Blahut algorithm is given in table 6.1.

We now show that $p_{\hat{X}}$ indeed is the function p_Y that minimizes the relative entropy $H(p_{X\hat{X}}\|p_X p_Y)$. We can do this using a standard optimization method. We find the function $p_{\hat{X}}$ that minimizes (for a discrete alphabet, the infimum is a minimum)

$$H(p_{\hat{X}|X}p_X\|p_X p_Y) = \sum_{x,\hat{x}} p_{\hat{X}|X}(\hat{x}|x)p_X(x)\log(\frac{p_{\hat{X}|X}(\hat{x}|x)}{p_Y(\hat{x})}) \tag{6.55}$$

under the constraint that $\sum_{\hat{x}} p_Y(\hat{x}) = 1$. The Lagrange multiplier method leads to the solution 6.53. We derived the same result earlier in example 2.13 in a different manner.

6.6 Rate Distribution over Independent Variables

In this section, we consider the problem of having to code a set of independent continuous-alphabet variables $X_1, \cdots, X_k$. We assume that the rate-distortion functions are differentiable whenever the rate is positive. Our analysis will tell us how to distribute the bit rates for ideal coding of independent processes, so as to get the lowest overall distortion. It will be shown that independent coding of the processes is optimal. The results provide useful insight for practical cases such as, for example, the case where we encode the outputs of a filter bank operating on a stationary signal. In section 6.6.1, we analyze the more general case that has as only assumption that the rate-distortion function is differentiable. In section 6.6.2, we discuss Gaussian variables and generalize the results of that case to determine the rate-distortion function for colored Gaussian processes.

6.6.1 General Case with Differentiable $R(D)$ Functions

In this subsection, we consider a random vector consisting of k independent random variables,

$$X^k = [X_1, \cdots, X_k]^T, \tag{6.56}$$

and a distortion measure that is the sum of the distortion measures for the variables,

$$D_{X^k} = \sum_{i=1}^{k} D_{X_i}. \tag{6.57}$$

The probability density functions of the variables are not assumed to be identical. Our goal is to show that we can encode the variables separately without loosing optimality and to find the optimal distribution of the rates between these separate encodings of the independent variables.

We start with finding a relation between the mutual information of the original vectors X^k and reconstructed vectors $\hat{X}^k$ and those of the original variables X_i and the reconstructed variables $\hat{X}_i$. We note that because of the chain rule we can write

$$\begin{aligned} I(X^k;\hat{X}^k) &= h(X^k) - h(X^k|\hat{X}^k) \\ &= h(X^k) - \sum_{i=1}^{k} h(X_i|X_1,\cdots,X_{i-1},\hat{X}^k). \end{aligned} \tag{6.58}$$

Exploiting the independence of the X_i and the fact that conditioning reduces entropy, we obtain

$$\begin{aligned} I(X^k;\hat{X}^k) &= \sum_{i=1}^{k} \Big(h(X_i) - h(X_i|X_1,\cdots,X_{i-1},\hat{X}^k)\Big) \\ &\geq \sum_{i=1}^{k} \Big(h(X_i) - h(X_i|\hat{X}_i)\Big) \\ &= \sum_{i=1}^{k} I(X_i;\hat{X}_i). \end{aligned} \tag{6.59}$$

Next, we move on to the rate-distortion function. Let the conditional probability $p_{X^k|\hat{X}^k}$ be such that $I(X^k;\hat{X}^k)$ on the left hand side of inequality 6.59 corresponds to the rate-distortion limit. Then, we see that

$$R_{X^k}(D_{X^k}) \geq \sum_{i=1}^{k} I(X_i;\hat{X}_i) \geq \sum_{i=1}^{k} R_{X_i}(D_{X_i}). \tag{6.60}$$

This inequality is, in fact, an equality, since $R_{X^k}(D_{X^k})$ is by definition the infimum over all $f_{X^k|\hat{X}^k}$ resulting in a distortion of D_{X^k} or less. It is easy to see that

$$f_{X^k|\hat{X}^k} = \prod_{i=1}^{k} f_{X_i|\hat{X}_i} \tag{6.61}$$

makes inequality 6.60 an equality. Equation 6.61 implies that

$$h(X_i|\hat{X}^k) = h(X_i|\hat{X}_i). \tag{6.62}$$

Equation 6.62 has implications for practical coding. In our proof that the rate-distortion limit is attainable with a deterministic real code we used vector quantizers of very large

dimension. Now let each X_i be a time sample of an independent process with index i (for clarity, we could add a time index n and write $X_{i,n}$). Then it follows from equation 6.62 that we can reach the rate-distortion limit for the k-dimensional vector process by encoding (quantization and lossless coding) each of the k processes independently. We can thus encode the k processes independently without loss of coding efficiency.

We now determine the rate allocation. Our objective is to minimize the overall rate

$$R_{X^k}(D_{X^k}) = \sum_{i=1}^{k} R_{X_i}(D_{X_i}). \tag{6.63}$$

Exploiting our earlier result, we encode each component separately, and require that each has a positive rate. It is convenient to state these constraints in terms of the respective distortions:

$$D_{X_i} \quad \leq \quad \check{D}_{X_i}, \tag{6.64}$$

where $\check{D}_{X_i}$ is the infimum distortion for which the rate is zero,

$$\check{D}_{X_i} = \inf_{\{D:R_{X_i}(D)=0\}} D. \tag{6.65}$$

We first define the problem for the case where all inequality constraints are effectively equality constraints. Then the method of Lagrange multipliers gives as extended criterion

$$\eta = \sum_{i=1}^{k} (R_{X_i}(D_{X_i}) + \nu D_{X_i} + \mu_i D_{X_i}). \tag{6.66}$$

The goal is simply to find the set $\{D_{X_i}\}_{i=1,\cdots,k}$ that minimize this criterion.

For the constrained minimization of the expression in equation 6.66, we use similar logic as that used for minimizing the expression in equation 6.44. We note that η is convex in D_{X_i} since any rate-distortion function is convex and the additional terms are linear in D_{X_i}. The minimum of η is either on the constraint boundary or it is inside the constraint boundary. If it is inside the constraint boundary, constraint 6.64 can be ignored, i.e., μ_i can be set to zero. In the other case, the boundary itself must be a minimum in the extended criterion. Without the constraint, the derivative of η with respect to D_{X_i} must be negative since otherwise the minimum would be inside the constraint boundary. This implies that the derivative of $\mu_i D_{X_i}$ with respect to D_{X_i} must be *positive* and that, therefore, μ_i must be positive. Thus we obtain the relations

$$\mu_i = 0, \qquad D_{X_i} < \check{D}_{X_i}, \tag{6.67}$$

$$\mu_i \geq 0, \qquad D_{X_i} = \check{D}_{X_i}. \tag{6.68}$$

Differentiating η with respect to the D_{X_i} and setting the result to zero leads to the following set of equations:

$$\dot{R}_{X_i}(D_{X_i}) + \nu = 0, \qquad D_{X_i} < \check{D}_{X_i}, \tag{6.69}$$

$$\dot{R}_{X_i}(D_{X_i}) + \nu \leq 0, \qquad D_{X_i} = \check{D}_{X_i}, \tag{6.70}$$

where $\dot{R}_{X_i}(\cdot)$ is the derivative of $R_{X_i}(\cdot)$.

Table 6.2: General rate-distribution algorithm. The overall distortion is the sum of the individual-variable distortions.

1. select a value for ν
2. if possible solve D_{X_i} from $\dot{R}_{X_i}(D_{X_i}) = -\nu$ in the range $D_{X_i} < \check{D}_{X_i}$; otherwise set $D_{X_i} = \check{D}_{X_i}$.
3. the rate for process i is $R_{X_i}(D_{X_i})$.

To find a range of solutions for the optimal rate allocations between sources with known rate-distortion functions is simple. One rate allocation is found for each value of ν. The optimal rate allocation algorithm is given in table 6.2.

It follows from equations 6.69 and 6.70 that the rate derivative, $\dot{R}_{X_i}(D_{X_i})$ is the same for all variables, except for those variables for which this marginal rate does not exist in the range $D_i < \check{D}_{X_i}$, which have zero rate.

6.6.2 Gaussian Variables: Reverse Water Filling

Reverse water filling allows the determination of the rate distribution and the overall rate-distortion function for a set of independent Gaussian variables. For dependent variables, one cannot reach the overall rate-distortion function through independent codings. However, as will be shown below, the reverse water filling technique does provide a solution also for this case, through the use of a suitable transform. The results can be extended to the important case of Gaussian processes as is shown in the last part of this subsection.

Independent Normally-Distributed Variables

Let us first consider independent Gaussian variables, X_i, with variance σ_i^2 and a squared error criterion. In the next subsection it will be seen that these results are also relevant for *dependent* normally-distributed variables with a squared error criterion. The rate-distortion functions are, in nats:

$$R_{X_i}(D_{X_i}) = \begin{cases} \frac{1}{2}\log(\frac{\sigma_i^2}{D_{X_i}}), & D_{X_i} \leq \sigma_i^2 \\ 0 & D_{X_i} = \sigma_i^2 \end{cases} \tag{6.71}$$

Differentiating for $D(X_i) \leq \sigma^2$ results in

$$\dot{R}_{X_i}(D_{X_i}) = -\frac{1}{2D_{X_i}}, \qquad D_{X_i} \leq \sigma_i^2. \tag{6.72}$$

We also note that $R_{X_i}(D_{X_i}) = 0$ corresponds to $D_{X_i} = \sigma_i^2$. Equations 6.70 and 6.69 become

$$-\frac{1}{2D_{X_i}} + \nu = 0, \qquad D_{X_i} < \sigma_i^2, \tag{6.73}$$

$$-\frac{1}{2D_{X_i}} + \nu \leq 0, \qquad D_{X_i} = \sigma_i^2. \tag{6.74}$$

Table 6.3: Rate-distribution algorithm for Gaussian variables and a squared error criterion. The overall distortion is the sum of the individual-variable distortions.

1. select a value for λ
2. $R_{X_i} = \max(0, \frac{1}{2}\log(\frac{\sigma_i^2}{\lambda}))$

Let us set $\lambda = \frac{1}{2\nu}$. The algorithm for constructing the rate distribution then simplifies to that given in table 6.3. The overall bit rate is

$$R = \sum_i R_{X_i} = \sum_i \max(0, \frac{1}{2}\log(\frac{\sigma_i^2}{\lambda})). \tag{6.75}$$

and the overall distortion is

$$D = \sum_i D_{X_i} = \sum_i \min(\lambda, \sigma_i^2). \tag{6.76}$$

Equations 6.75 and 6.76 specify the rate-distortion function for the jointly-normal distributed variable.

Equation 6.76 indicates that the value of λ corresponds to a bound on the distortion allowed for each variable. If the variance is less than this bound, then the variable is not encoded and the variance of the variable equals its distortion upon reconstruction. This implies that the variable is reconstructed by its mean value. On the other hand, if the variance of a variable is higher than the bound on the allowed distortion, then a nonzero rate is allocated to this variable to reduce the distortion so that it satisfies the bound. This is illustrated in the bar diagram of the left side of figure 6.7. The total bar lengths indicate the variance of the variables, and the lightly hatched areas within the bar correspond to the distortion for each particular variable.

At this point, it is useful to remember the relation between the variance of the original and reconstructed variables at the rate-distortion bound. The backward-channel interpretation (valid for Gaussian variables and the squared-error criterion) tells us that in the present case the sum of the reconstructed-variable variance and the distortion equals the variance of the original variable. In other words, the reconstructed-variable variance is simply the original-variable variance minus the distortion, except when this is negative (the reconstructed-variable variance is then zero). Thus, the densely hatched areas in figure 6.7 indicate the variances of the reconstructed variables. When turning the figure upside down the bars are all filled with densely hatched area up to the same level, hence the name **reverse water filling**. The variances of the reconstructed variables are determined by the "water level".

The right-hand side of figure 6.7 displays the same bar diagram with a logarithmic vertical scale. The densely hatched area now has a bit-rate interpretation. From equation 6.75 it follows that in the logarithmic scale bar diagram the densely hatched within the bar above the dashed line corresponds to the rate allocation.

Example 6.10: Rate distribution between two Gaussian variables
We consider the case of two Gaussian variables, X_1 and X_2, with variance $\sigma_1^2 = 1$ and $\sigma_2^2 = 2$. We use the algorithm of table 6.3 to create the individual rate-distortion functions. We list some values thus obtained in table 6.4. More complete results, obtained in a similar fashion, are displayed in figure 6.8.

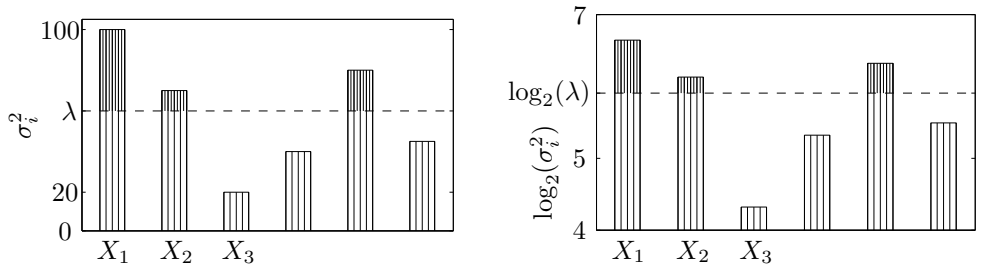

Figure 6.7: The principle of reverse water filling. Left: the original variances (full bar lengths) and the variances of the reconstructed variables (densely hatched); the distortion corresponds to the lightly hatched area within the bars. Right: the same plot on a logarithmic scale. The densely hatched areas now represent the bit rate.

Table 6.4: Some rate-distortion data for example 6.10.

λ	D_{X_1}	D_{X_2}	D	R_{X_1}	R_{X_2}	$R(D)$
0.10	0.10	0.10	0.20	1.66	2.16	3.82
0.50	0.50	0.50	1.00	0.50	1.00	1.50
1.00	1.00	1.00	2.00	0.00	0.50	0.50
2.00	1.00	2.00	3.00	0.00	0.00	0.00

Dependent Normally-Distributed Variables

Next, we consider the case where a set of Gaussian variables X_i are dependent and have covariance matrix $C_{ij} = \mathrm{E}[X_i X_j]$. We can transform the set of variables to a set of independent variables through diagonalization of the covariance matrix. Let the normalized eigenvectors of the symmetric matrix C form the rows of the unitary matrix U. Then the new set of variables W_i is the set of components of the vector W^k

$$W^k = UX^k, \tag{6.77}$$

where U diagonalizes the covariance matrix:

$$\Lambda = UCU^T. \tag{6.78}$$

Λ is a diagonal matrix with the elements λ_i along the diagonal. The squared-error distortion measure is not affected by a unitary transform. Since the determinant of a unitary matrix is unity, the differential entropy is also not affected. Thus, the entire discussion about independent Gaussian variables applies to the transformed variables $\{W_i\}_{i=1,\cdots,k}$. The overall rate is

$$R = \sum_i R_{W_i} = \sum_i \max(0, \frac{1}{2}\log(\frac{\lambda_i}{\lambda})) \tag{6.79}$$

and the overall distortion is

$$D = \sum_i D_{X_i} = \sum_i D_{W_i} = \sum_i \min(\lambda, \lambda_i). \tag{6.80}$$

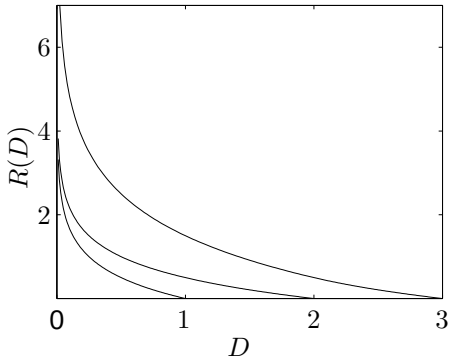

Figure 6.8: Individual and combined rate-distortion functions for the two Gaussian sources of example 6.10.

Normal Processes with Memory

Using the formulation for dependent variables, it is easy to obtain the rate-distortion function for Gaussian processes. By applying theorem 13, we can write for the rate

$$R = \frac{1}{4\pi}\int_{-\pi}^{\pi} \max(0, \log(\frac{R(\mathrm{e}^{j\omega})}{\lambda}))d\omega \tag{6.81}$$

and for the distortion

$$D = \frac{1}{2\pi}\int_{-\pi}^{\pi} \min(\lambda, R(\mathrm{e}^{j\omega}))d\omega, \tag{6.82}$$

where $R(\mathrm{e}^{j\omega})$ is defined in equation 3.30. Figure 6.9 illustrates reverse water filling for a Gaussian process. The densely hatched area defines the power spectrum of the reconstructed process. The area below both λ and the power spectrum defines the distortion of the reconstructed signal. The sum of the distortion and the power spectrum of the reconstructed signal forms the power spectrum of the original process.

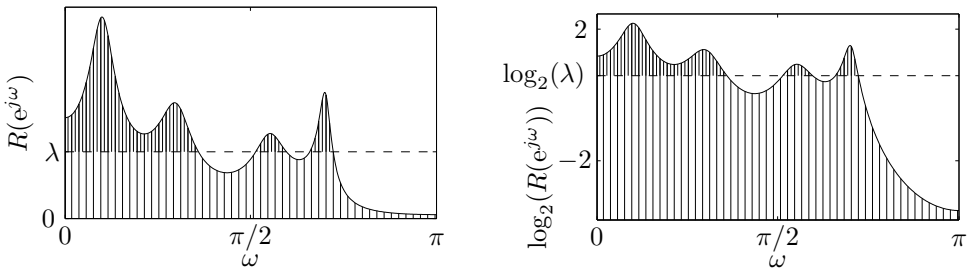

Figure 6.9: Principle of reverse water filling for a Gaussian process. Left: the original power spectrum $R(\mathrm{e}^{j\omega})$, the reconstruction power spectrum $R(\mathrm{e}^{j\omega}) - \lambda$ (densely hatched), and the distortion $\min(R(\mathrm{e}^{j\omega}), \lambda)$ (lightly hatched). Right: the same plot on a logarithmic scale. The densely hatched area now represents the bit rate.

It is interesting to analyze the behavior in the limit of low distortion, i.e., when $\lambda <$

$R(\mathrm{e}^{j\omega})$ for $\omega \in [-\pi, \pi]$. In this case we have as rate

$$R_{\mathrm{ldg}} = \frac{1}{4\pi}\int_{-\pi}^{\pi} \log(R(\mathrm{e}^{j\omega}))d\omega - \frac{1}{2}\log(\lambda), \tag{6.83}$$

where the subscript ldg indicates low distortion and Gaussian distribution. The distortion is simply $D_{\mathrm{ldg}} = \lambda$; inserting this into equation 6.83 gives the rate-distortion (or the distortion-rate) function for Gaussian processes with memory at low distortion:

$$R_{\mathrm{ldg}} = \frac{1}{4\pi}\int_{-\pi}^{\pi} \log(R(\mathrm{e}^{j\omega}))d\omega - \frac{1}{2}\log(D_{\mathrm{ldg}}). \tag{6.84}$$

From equation 6.37 and knowledge of the differential entropy of a Gaussian process (equation 3.32), we see that equation 6.84 corresponds to its Shannon lower bound. (This does not mean that reverse waterfilling implies that the Shannon lower bound is not tight.)

6.7 Problems

1. Prove that $x\log(x)$ is convex on $(0, 1)$.

2. Prove that the rate-distortion function for a Bernoulli(p) source (iid binary source with symbol probabilities p and $1-p$) and the Hamming distortion measure ($d(x, \hat{x}) = 0$ for $x = \hat{x}$ and $d(x, \hat{x}) = 1$ otherwise) is

$$R(D) = \begin{cases} H(p) - H(D), & 0 \le D \le \min(p, 1-p) \\ 0, & D > \min(p, 1-p) \end{cases},$$

   where $H(p)$ is a function defined as $H(p) = -p\log(p) - (1-p)\log(1-p)$.

3. Use the Blahut algorithm to find the rate-distortion function of problem 2.

4. A discrete-alphabet random variable is uniformly distributed over its k source symbols. Using the Hamming distortion measure ($d(x, \hat{x}) = 0$ for $x = \hat{x}$ and $d(x, \hat{x}) = 1$ otherwise), find the rate-distortion function.

5. Prove that $R_k(D) \ge R_{k+1}(D)$.

6. Show that equation 6.47 is correct. (Hint: note that $p(\hat{x}) = \sum_x p(\hat{x}|x)p(x)$.)

7. In rate-distortion theory we usually deal with sampled signals. However, it is straightforward to extend the theory to analog signals. It is well-known that we tolerate about 30 dB signal to noise ratio in telephone bandwidth speech. Let us assume that the telephone signal has a bandwidth of 3.5 kHz and that the signals are Gaussian.

   (a) What is the maximum information rate that can be transmitted over the telephone connection?

   (b) The information rate in speech can be argued to be lower. Why?

8. Find and plot in one plot the Shannon lower bound for the absolute error criterion and a given mean absolute error γ.

(a) An iid process with rectangular sample density.

(b) An iid process with Gaussian sample density.

(c) An iid process with Laplacian sample density.

(d) Indicate in which cases the lower bound coincides with the actual rate-distortion function.

9. Consider the error criterion $d(x, y) = |x - y|^3$.

(a) Using variational calculus, find the density $f_Y(y)$ that maximizes the differential entropy for a given $E[|Y|^3]$.

(b) Find a simple expression for the Shannon lower bound for this error criterion.

(c) In the case of the Gaussian density and the squared error, and also in the case of the Laplacian density and the absolute error (cf. example 6.8), a density that is of the same form as the maximum-differential-entropy density for the criterion leads to a tight Shannon bound. Explain whether this the case for the $d(x, y) = |x - y|^3$ criterion?

10. Consider the quantization of the random variable X. The distortion criterion for the error $w = x - \mathcal{Q}(x)$ is illustrated in figure 6.10 (the distortion is infinite for $|w| > 2$).

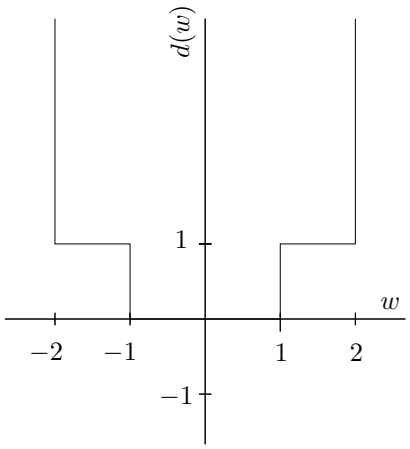

Figure 6.10: Distortion criterion, $d(w)$, as a function of w.

(a) Find the density that maximizes the differential entropy for a given finite $\int f_W(w)d(w)dw$, where $d(w)$ is the distortion criterion. Make a plot of a typical density function.

(b) Consider a Gaussian random variable X with unit variance. Find the Shannon lower bound for this variable with the distortion criterion of figure 6.10 and plot it.

(c) Argue that the Shannon lower bound is tight or not tight.

11. Consider the squared Cauchy density,

$$f_X(x) = \frac{2}{\pi}(1 + x^2)^{-2},$$

and the absolute-error criterion.

(a) Compute and plot the Shannon lower bound.

(b) Show that the bound is tight over only a particular range of distortions.

(c) For the range where the lower bound is not tight, discretize the density and approximate $R(D)$ using the Blahut algorithm. Add your new results to the plot.

12. Consider the rectangular density

$$f_X(x) = \begin{cases} 1, & x \in [0,1) \\ 0, & \text{elsewhere} \end{cases}$$

and a distortion criterion

$$d(x,y) = \begin{cases} 0, & |x-y| \leq 1/4 \\ 1, & |x-y| > 1/4 \end{cases} .$$

Discretize this problem and approximate the rate-distortion function using the Blahut algorithm.

13. Give the Shannon lower bound for a first-order Gauss-Markov process with correlation ρ and the squared error criterion.

14. Prove that $(1-ab)^m \leq 1 - a + \mathrm{e}^{-bm}$.

15. You want to estimate the distortion versus bit rate trade-off for a particular stationary process. For simplicity, you decide to assume the process is a (zero-mean) Gaussian process. You first estimate the power spectrum, $R(\mathrm{e}^{j\omega})$, and find figure 6.11.

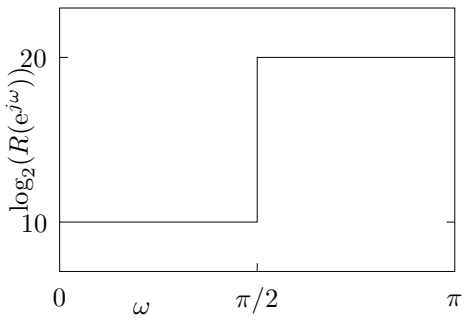

Figure 6.11: Power spectrum for problem 15.

(a) Using the figure, illustrate graphically how you obtain a particular point on the distortion-rate function.

(b) What is the distortion at zero bit rate? Explain.

(c) What bit rate corresponds to zero distortion? Explain.

(d) Describe the bit rate versus distortion trade-off for this case by means of equations.

(e) Plot the rate-distortion curve using your answer in 15d. Is your curve convex or concave?

16. Consider the independent variables X_1 and X_2 with Laplace densities that are of the form $f_{X_i}(x_i) = \frac{a_i}{2} e^{-a_i|x|}$. An absolute error criterion applies and the overall distortion is defined as $E[|x_1 - \hat{x}_1| + |x_2 - \hat{x}_2|]$. In the following, use $a_1 = 1/2$ and $a_2 = 1/3$.

    (a) Determine and plot the rate-distortion curves for the two variables.

    (b) Find a set of equations that determines the overall rate-distortion function. (The theory for Gaussian variables has not been proven to apply.)

    (c) Using a computer, plot the overall rate-distortion function.

17. Let $f_G(x)$ be a Gaussian density with unit variance. We have process, X_i, where the odd samples have distribution $f_{\text{odd}}(x) = f_G(x)$ and the even samples have distribution $f_{\text{even}}(x) = f_G(x-5)$. The odd samples form an iid sequence and the even samples also form an iid sequence. We consider the squared error criterion.

    (a) Compute the differential entropy rate of the random variables having densities $f_{\text{odd}}(x)$ and $f_{\text{even}}(x)$.

    (b) Assuming you know the time index, i.e., you know which samples are odd and even, derive a formula for the operational rate-distortion relation for the process X_i (scalar quantization).

    (c) Consider the case where you got it wrong, i.e., you think that the odd samples are the even samples and vice versa. Derive an operational rate-distortion formula for this mismatched case.

    (d) To prevent problems, you decide to use a single distribution for all samples. Provide the distribution and again derive the operational rate-distortion relation.

7

High-Rate Quantization

7.1 Introduction

It has been found that the relation between rate and distortion becomes amenable to mathematical analysis only under certain conditions. In chapter 6, we considered bounds on the relation between the average rate and average distortion under the assumptions that we code very long symbol sequences, and that the symbols can be encoded jointly. In this chapter, we consider the relation between rate and distortion in the case of high rate, an approach that was pioneered by Bennett [26]. The high-rate assumption implies that the data density does not change significantly within the size of a quantization cell.

Except for the high-rate restriction, high-rate quantization theory can be used similarly to rate-distortion theory to estimate the lowest mean rate per source symbol for stationary sources. However, high-rate quantization theory is more general than rate-distortion theory in two aspects. As mentioned, the coded symbol sequence considered need not be long. Moreover, high-rate quantization theory provides a relation between mean distortion and bit rate when we add the restriction that each source symbol is encoded with a fixed number of bits. In other words, high-rate theory allows us to find a relation between rate and distortion for fixed-rate coders.

High-rate quantization theory leads to the design of practical quantizers, again something that rate-distortion theory does not. While these quantizers are guaranteed to be efficient when the high-rate assumption is correct, they often perform well for practical rates. A particularly attractive result of high-rate quantization theory is that uniform quantizers (all cells equal in size and shape) are optimal if they are followed by lossless coding.

In this chapter, we first define quantization more precisely. We then consider high-rate scalar quantization, followed by the more complicated world of high-rate vector quantization. We compare our results to the rate-distortion function and use high-rate theory to compare the strengths of vector and scalar quantization. We end this chapter with a discussion of the design of practical quantizers based on high-rate theory.

7.2 Quantization: Definitions and Measures

Before we start with our discussion of high-rate quantization, it is necessary to define more clearly the properties of quantizers. We do this first for scalar quantization and then for vector quantization.

7.2.1 The Scalar Quantizer

A scalar quantizer is a noninvertible mapping, $\mathcal{Q}$, of the real line, $\mathbb{R}$, onto a countable set of points, $\mathcal{C} = \{c_i\}_{i\in\mathcal{I}}$, where $\mathcal{I}$ is a countable set of indices:

$$\mathcal{Q}: \ \mathbb{R} \to \mathcal{C}. \tag{7.1}$$

$\mathcal{C}$ is called a **codebook** and the c_i are referred to as codebook entries, centroids, or reconstruction points. A cell $\mathtt{V}_i$ is defined as the collection of points on the real line that the quantizer maps onto c_i:

$$\mathtt{V}_i = \{x \in \mathbb{R} : \mathcal{Q}(x) = c_i\}, \tag{7.2}$$

This formal definition allows the cells to be disjoint (consisting of unconnected parts), a situation that does not occur in most common coding applications. In practice, we deal with **regular quantizers** [24], which are defined as quantizers for which

1. each cell is an open interval (x_{i-1}, x_i) together with one or both endpoints;
2. $c_i \in (x_{i-1}, x_i)$.

For regular quantizers, it is natural to interpret the quantizer output as an approximation of the input value. One is willing to accept the approximation as a substitute for the original value because it allows encoding with a finite number of bits. In other words, we accept distortion of the signal to make coding possible.

The encoding is based on the index i. It is, therefore, often useful to split the quantization mapping $\mathcal{Q}$ into two mappings: a noninvertible encoder mapping $\mathcal{E}$ from input x to an index i, $i = \mathcal{E}(x)$, and an invertible decoder mapping $\mathcal{D}$ from the index i to the quantizer reconstruction point c_i:

$$c_i = \mathcal{D}(i) = \mathcal{D}(\mathcal{E}(x)) = \mathcal{Q}(x), \quad \forall_{x\in\mathtt{V}_i}. \tag{7.3}$$

The mappings can be written as

$$\begin{aligned} \mathcal{E}: &\quad \mathbb{R} \to \mathcal{I}, \\ \mathcal{D}: &\quad \mathcal{I} \to \mathcal{C}. \end{aligned} \tag{7.4}$$

Having defined the scalar quantizer, it is useful to consider measures to evaluate its performance. It is natural to consider the distortion incurred to be the main performance measure. However, without a specification of the conditions under which a particular distortion is obtained, this measure is not meaningful. Considering the fact that coding at a finite bit rate is the motivation for using quantizers, a constraint must be imposed

on the required codeword length for encoding the indices. Two cases are commonly considered: the case where all codewords are constrained to have a specific length and the case where the average codeword length is constrained. These two cases are referred to as **resolution-constrained quantization** and **entropy-constrained quantization**, respectively.

The fixed-codeword-length constraint is the more commonly used rate constraint in the design of practical quantizers. A fixed codeword length in bits corresponds to a fixed codebook size. In quantizer design, the resolution constraint is usually defined as a constraint on the codebook size, with the codebook size often set to be a power of two to facilitate binary encoding.

Naturally, the design of practical quantizers reflects their usage in communication systems. Constrained-resolution quantizers are commonly used in communication systems for a number of reasons. First, and probably most important, is that most traditional communication systems have channels of fixed rate. Second, particularly at low rates (which are discussed in the next chapter), it is often conceptually simpler to find an optimal quantizer under the resolution constraint than under the entropy constraint. Third, the use of entropy-constrained vector quantization implies the usage of a lossless coder, thus increasing system complexity.

The entropy constraint is less restrictive than the resolution constraint, thus resulting in lower average rates, at the penalty of not having a constant rate. With the increase in computational capabilities of hardware and the increase in the usage of statistical communication networks (e.g., code-division multiplexing in mobile communications and packet networks as used in the Internet) that facilitate variable rate, it is becoming more attractive to exploit the rate advantage inherent in constrained-entropy coding.

Naturally, the objective of quantization can also be formulated as the minimization of the entropy or the codebook size, with the distortion as constraint. However, such formulations are uncommon and are not used here.

7.2.2 The Vector Quantizer

The vector quantizer is a straightforward generalization of the scalar quantizer. A vector quantizer is a mapping, $\mathcal{Q}$, of k-dimensional Euclidean space, $\mathbb{R}^k$, onto a countable set of points, $\mathcal{C}^k = \{c_i^k\}_{i\in\mathcal{I}}$, where $\mathcal{I}$ is a countable set of indices:

$$\mathcal{Q}: \ \mathbb{R}^k \to \mathcal{C}^k. \tag{7.5}$$

A cell $\mathtt{V}_i$, associated with c_i^k, is defined as the collection of points in $\mathbb{R}^k$ that the quantizer maps onto c_i^k:

$$\mathtt{V}_i = \{x^k \in \mathbb{R}^k : \mathcal{Q}(x^k) = c_i^k\}. \tag{7.6}$$

As in the scalar case, this definition allows the cell $\mathtt{V}_i$ to be disjoint.

The set of points making up a cell $\mathtt{V}_i$ is called **convex** if any point on the straight line between two points within the set is also in the set. That is, for a convex cell

$$a \in \mathtt{V}_i \wedge b \in \mathtt{V}_i \Rightarrow \alpha a + (1-\alpha)b \in \mathtt{V}_i, \quad \forall_{\alpha\in[0,1]}. \tag{7.7}$$

A vector quantizer is called **regular** [24] if

1. all its cells are convex and
2. $c_i^k \in \mathtt{V}_i$.

As in the scalar case, it is useful to break up the quantization mapping $\mathcal{Q}$ into the noninvertible mapping $\mathcal{E}$ to an index and the invertible mapping $\mathcal{D}$ from the index to the reconstruction vector

$$c_i^k = \mathcal{D}(i) = \mathcal{D}(\mathcal{E}(x^k)) = \mathcal{Q}(x^k), \forall_{x^k \in \mathtt{V}_i}. \tag{7.8}$$

The mappings are now written as

$$\begin{aligned} \mathcal{E}: &\quad \mathbb{R}^k \to \mathcal{I}, \\ \mathcal{D}: &\quad \mathcal{I} \to \mathcal{C}^k. \end{aligned} \tag{7.9}$$

Vector quantizers can be obtained under both the entropy constraint and the resolution constraint. Again resolution-constrained quantization is more commonly used than entropy-constrained quantization and the motivations provided earlier apply here as well.

When coding samples or blocks of stationary sequences, we can use the principle of asymptotic equipartition (cf. section 2.8) to argue qualitatively that the indices of vector quantizers should have more equal probability than the indices of scalar quantizers. As a result, constrained-entropy coding results in a smaller advantage in rate over constrained-resolution coding for vector quantization than for scalar quantization of a stationary sequence.

7.2.3 Distortion Criteria

Before embarking on a discussion of quantization, it is useful to define the notation of the distortion measures. We write the distortion between an original value or vector x^k and its quantized value or vector as $d(x^k, \mathcal{Q}(x^k))$. Let $f_{X^k}(x^k)$ be the probability density function of the random variable X^k. The mean distortion is then

$$D(X^k, \mathcal{Q}(X^k)) = \mathrm{E}[d(X^k, \mathcal{Q}(X^k))] = \int_{\mathbb{R}^k} f_{X^k}(x^k) d(x^k, \mathcal{Q}(x^k)) dx^k, \tag{7.10}$$

In practice, the distortion measure most commonly used in quantization is the mean squared-error criterion. In this case, equation 7.10 becomes

$$D(X^k, \mathcal{Q}(X^k)) = \frac{1}{k} \int_{\mathbb{R}^k} f_{X^k}(x^k) \left(\sum_{m=1}^{k} (x_m^k - \mathcal{Q}(x^k)_m)^2 \right) dx^k. \tag{7.11}$$

In the section of this chapter related to vector quantization, we use a more general distortion criterion than that used in the section on scalar quantization. In this chapter we consider the mean r-th power distortion measure:

$$d(x^k, \mathcal{Q}(x^k)) = \frac{1}{k} \|x^k - \mathcal{Q}(x^k)\|_r \equiv \frac{1}{k} \left(\sum_{m=1}^{k} (x_m - \mathcal{Q}(x^k)_m)^2 \right)^{\frac{r}{2}}, \tag{7.12}$$

a measure that is normalized on a per-dimension basis ($\mathcal{Q}(x^k)_m$ is the m'th component of the k-dimensional vector $\mathcal{Q}(x^k)$). The distortion criterion of equation 7.12 reduces to the squared-error criterion for $r = 2$. It is a single-letter criterion only for the squared-error case.

7.3 High-Rate Scalar Quantization

As mentioned before, the motivation for performing the analysis under the high-rate constraint is that it makes relations analytically tractable. The purpose of this section is to provide the basic theory in a less general but easily comprehensible form. Thus, we restrict ourselves to scalar quantization and the squared-error criterion. We generalize most of the results in section 7.4 to the vector case and a more general distortion criterion.

We make the following additional assumptions. We assume that the scalar quantizer is a regular quantizer and that the reconstruction points are centered within the cells. We also assume that the two unbounded cells do not influence the derivations.

7.3.1 The Distortion as a Function of Centroid Density

We begin with the evaluation of the mean distortion in a particular cell in the high-rate case. Let us denote the integral over a cell i by $\int_{\mathtt{V}_i} dx$. Then, writing the distortion per cell as D_i, we have

$$\begin{aligned} D_i &= \frac{\int_{\mathtt{V}_i} f_X(x) d(x, \mathcal{Q}(x)) dx}{\int_{\mathtt{V}_i} f_X(x) dx} \\ &\approx \frac{f_X(c_i) \int_{\mathtt{V}_i} d(x, \mathcal{Q}(x)) dx}{f_X(c_i)\Delta_i} \\ &= \frac{1}{\Delta_i} \int_{\mathtt{V}_i} (x - c_i)^2 dx \\ &= \frac{1}{\Delta_i} \int_{-\Delta_i/2}^{\Delta_i/2} y^2 dy \\ &= \frac{1}{12}\Delta_i^2, \end{aligned} \tag{7.13}$$

where Δ_i is the step size of the quantizer and where we assumed that the centroid is located in the middle of the cell so as to minimize the distortion.

Equation 7.13 is commonly used. An example of its application is the computation of the quantization step size for which the quantization noise of an audio coder remains inaudible, e.g., [27, 1].

The step size Δ_i of a scalar quantizer is inversely proportional to the local density of the centroids (quantizer reconstruction points) per unit length, denoted as $g_C(x)$:

$$g_C(c_i) = \frac{1}{\Delta_i}. \tag{7.14}$$

Using this fact and the distortion per cell (equation 7.13), we can relate the mean distortion to the density $g_C(c_i)$. Denoting by $p_I(i)$ the index probability we have

$$\begin{aligned} D &= \sum_{i\in\mathcal{I}} p_I(i) D_i \\ &\approx \frac{1}{12}\sum_{i\in\mathcal{I}} p_I(i)\Delta_i^2 \\ &= \frac{1}{12}\sum_{i\in\mathcal{I}} p_I(i) g_C(c_i)^{-2} \\ &= \frac{1}{12}\sum_{i\in\mathcal{I}} \int_{\mathtt{V}_i} f_X(x)dx \; g_C(c_i)^{-2} \\ &\approx \frac{1}{12}\int_{\mathbb{R}} f_X(x) g_C(x)^{-2} dx. \end{aligned} \tag{7.15}$$

This equation relating the overall distortion and the density of the reconstruction points is used in the following sections to find the optimal density of the reconstruction points of a scalar quantizer in the high-rate case for both the constrained-resolution and constrained-entropy cases. From this density, we determine the respective relations between distortion and rate.

7.3.2 Constrained Resolution

While it is generally difficult to determine the optimal constrained-resolution codebook, this task is straightforward when the high-rate approximations apply. We recall that $\log_2(N)$, where N is the number of reconstruction points, is the (fixed) codeword length. Thus, we essentially want to find the centroid density function $g_C(x)$ that minimizes D in equation 7.15 under the constraint that $g_C(x)$ integrates over the real line $\mathbb{R}$ to the total number of reconstruction points N. That is, we want to minimize equation 7.15 under the constraint

$$\int_{\mathbb{R}} g_C(x)dx = N. \tag{7.16}$$

This problem can be solved using variational calculus. Using the Lagrange-multiplier method, we obtain as modified criterion

$$\eta = \frac{1}{12}\left(\int_{\mathbb{R}} (f_X(x) g_C(x)^{-2} + \lambda g_C(x))dx - \lambda N\right), \tag{7.17}$$

where λ is the Lagrange multiplier. The Euler-Lagrange equation for this case is:

$$-2 f_X(x) g_C(x)^{-3} + \lambda = 0. \tag{7.18}$$

Using equations 7.18 and 7.16 we find as solution

$$g_C(x) = N\frac{f_X(x)^{\frac{1}{3}}}{\int_{\mathbb{R}} f_X(x)^{\frac{1}{3}}dx}. \tag{7.19}$$

We can use this to find a relation between distortion and rate. Filling out equation 7.19 into equation 7.15, we obtain

$$D = \frac{1}{12N^2}\left(\int_{\mathbb{R}} f_X(x)^{\frac{1}{3}}dx\right)^3. \tag{7.20}$$

Using the fact that the rate is $R = \log(N)$, we obtain the following relation between rate and distortion:

$$R = -\frac{1}{2}\log(12D) + \frac{3}{2}\log(\int_{\mathbb{R}} f_X(x)^{\frac{1}{3}} dx). \tag{7.21}$$

Example 7.1: Constrained-resolution coding of a Gaussian variable
Let us first consider the centroid density for the high-rate encoding of a Gaussian variable with variance σ^2 under the constrained-resolution constraint. Equation 7.19 shows that, for a mean squared-error criterion, the density of the centroids is less emphasized than the data density. In other words, to prevent the error from growing too large in regions of low data density, more centroids must be committed there. Figure 7.1 shows the centroid distribution for the case $\sigma^2 = 1$. In this example, the centroid density is still relatively large in regions where the data density is already relatively small.

Next, we consider the relation between rate and distortion. Using equation 7.21, we obtain, in bits

$$\begin{aligned} R &= -\frac{1}{2}\log_2(12D) + \frac{1}{2}\log_2(2\pi\sigma^2 3^{\frac{3}{2}}) \\ &= \frac{1}{2}\log_2(\frac{\sigma^2}{D}) + \frac{1}{2}\log_2(\frac{2\pi 3^{\frac{3}{2}}}{12}), \end{aligned}$$

which is $\frac{1}{2}\log_2(\frac{2\pi 3^{\frac{3}{2}}}{12}) \approx 0.72$ bits above the rate-distortion function (see example 6.3), independently of the distortion level.

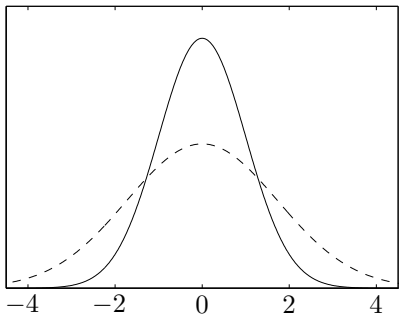

Figure 7.1: A Gaussian data density of unity variance (solid line) and its optimal centroid density (dashed line) for a mean squared-error criterion.

7.3.3 Constrained Entropy

If a lossless encoding of the quantization indices can be used, then one should consider an entropy constraint for the indices when evaluating distortion. In practice, the *first-order* entropy of the indices is generally used as the measure of rate. Let I be the random quantization index. Then the first-order entropy of the quantization indices is

$$H(I) = -\sum_{i \in \mathcal{I}} p_I(i)\log(p_I(i)),$$

where

$$p_I(i) = P(x \in \mathtt{V}_i) = \int_{\mathtt{V}_i} f_X(x)dx. \tag{7.22}$$

In the following, we first further develop the expression for $H(I)$ and then determine the relation between the rate $H(I)$ and distortion. We compare the resulting relation to the Shannon lower bound on the rate-distortion function obtained in section 6.4.

Relation of Distortion and Rate

We first determine the (first-order) index entropy in terms of the (first-order) differential entropy of the data and in terms of the centroid density $g_C(x)$:

$$\begin{aligned} H(I) &= -\sum_{i\in\mathcal{I}} p_I(i)\log(p_I(i)) \\ &\approx -\sum_{i\in\mathcal{I}} f_X(c_i)\Delta_i \log(f_X(c_i)\Delta_i) \\ &\approx -\int_{\mathbb{R}} f_X(x)\log(\frac{f_X(x)}{g_C(x)})dx \\ &= h(X) + \mathrm{E}[\log(g_C(X))]. \end{aligned} \tag{7.23}$$

This relation is used below.

The problem we set out to solve is to minimize the distortion

$$D = \int_{\mathbb{R}} f_X(x)d(x, \mathcal{Q}(x))dx \tag{7.24}$$

for a given entropy in equation 7.23. In the high-rate case, this problem is straightforward to solve. First we recall the high-rate equations. The distortion is (see equation 7.15)

$$D = \frac{1}{12}\int_{\mathbb{R}} f_X(x)g_C(x)^{-2}dx \tag{7.25}$$

and the constraint is equation 7.23.

We note that the differential entropy of X, $h(X)$, does not vary during the optimization, and that, thus, the constraint can be simplified to

$$b = \mathrm{E}[\log(g_C(X))] = \int_{\mathbb{R}} f_X(x)\log(g_C(x))dx, \tag{7.26}$$

where b is a constant. Using the by-now familiar Lagrange-multiplier approach we obtain as modified distortion criterion

$$\eta = \frac{1}{12}\left(\int_{\mathbb{R}} \left(f_X(x)g_C(x)^{-2} + \lambda f_X(x)\log(g_C(x))\right)dx - \lambda b\right), \tag{7.27}$$

which renders the Euler-Lagrange equation

$$-2g_C(x)^{-3} + \lambda g_C(x)^{-1} = 0, \tag{7.28}$$

which is solved by

$$g_C(x) = g_C = \text{constant}. \tag{7.29}$$

Thus, entropy-constrained optimization of the high-rate scalar quantizer results in a quantizer with uniform reconstruction-point density.

An important corollary of the uniformity of the reconstruction-point density is that the quantizers are the same for all distortion measures. *This means that we can design an optimal entropy-constrained quantizer even when the precise distortion measure is not known.*

The simplicity of the solution for the optimal constrained-entropy quantizer allows us to rewrite earlier expressions. From 7.23 it follows that

$$g_C = 2^{H(I)-h(X)}. \tag{7.30}$$

Combining this with 7.14 it is seen that for high rate

$$\Delta = 2^{-H(I)+h(X)}, \tag{7.31}$$

where we omitted the subscript i, since all Δ_i are equal.

We also find that

$$\begin{aligned} D &\approx \frac{1}{12}\int_{\mathbb{R}} f_X(x) g_C^{-2} dx \\ &= \frac{1}{12} 2^{-2H(I)+2h(X)}, \end{aligned} \tag{7.32}$$

or, asymptotically

$$H(I) = h(X) - \frac{1}{2}\log(12D). \tag{7.33}$$

This equation shows that, in the high-rate, entropy-constrained case, the logarithm of the RMS distortion is just the difference between the differential entropy of the variable and the entropy of the quantization indices (plus a constant). As expected, if the entropy of the quantization indices increases, the distortion decreases.

Example 7.2: Constrained-entropy coding of a Gaussian variable
We consider a Gaussian variable with variance σ^2. As for any random variable, the optimal high-rate centroid density for entropy-constrained coding is uniform. The quantizer step size as a function of the rate follows from equation 7.31:

$$\Delta = \sqrt{2\pi \mathrm{e} \sigma^2}\ 2^{-H(I)},$$

where the rate $H(I)$ is specified in bits.

The relation between rate $H(I)$ and distortion D for a Gaussian variable follows from equations 7.33 and 3.22:

$$H(I) = \frac{1}{2}\log(\frac{\sigma^2}{D}) + \frac{1}{2}\log(\frac{2\pi \mathrm{e}}{12}),$$

which is $\frac{1}{2}\log(\frac{2\pi \mathrm{e}}{12}) \approx 0.25$ bit above the rate-distortion function (cf. example 6.3). Comparing constrained-entropy coding to constrained-resolution coding (example 7.1) we see that that constrained-entropy coding has a rate advantage of almost 0.5 bit for a Gaussian variable.

Since the differential entropy appears in equation 7.33, we can use it to determine useful bounds on this high-rate rate-distortion relation. Consider the case where we do not know the probability density function of a random variable X but we do know its variance σ^2. The Gaussian distribution maximizes the differential entropy of a random variable given only its variance. The result already provided in example 7.2 then becomes an upper bound (over all data densities) on the high-rate rate-distortion relation:

$$H(I) \leq \frac{1}{2}\log(\frac{\sigma^2}{D}) + \frac{1}{2}\log(\frac{2\pi \mathrm{e}}{12}). \tag{7.34}$$

Encoder Mismatch and Robust Encoding

It is common that the probability density of the variable to be encoded is not precisely known. More-over this probability-density may not always be the same in a particular application. It then becomes important to design a *robust* encoder, that guarantees that we never exceed a known rate. We can do this by optimizing our encoder for a Gaussian variable [28, 29]. Below, the random variable X is the Gaussian design variable, whereas the random variable Y is the actual input variable, which has some arbitrary probability density. To simplify our notation, we assume that both variables are zero mean.

We start with noting that the distortion does not depend on the probability density of Y:

$$\begin{aligned} D &\approx \frac{1}{12}\int_{\mathbb{R}} f_Y(y) g_C^{-2} dy \\ &= \frac{1}{12} g_C^{-2} = \frac{1}{12}\Delta^2. \end{aligned} \tag{7.35}$$

Next, we consider the rate. We start using the vantage point of section 5.2.4, which discusses the rate obtained when a lossless code is mismatched to the actual symbol probability distribution. We consider the quantization indices j of Y to be distributed according to $p_J(j)$ and the quantization indices of X to be distributed according to $p_I(i)$. Furthermore, we assume that $l(x) = -\log_2(p_I(i))$ is integer for all i (this implies that the Shannon code is optimal, and, therefore, that the Kraft inequality is an equality). Under these conditions, if we transmit the random variable Y using the optimal scalar encoding of a variable X, the overall rate is

$$\begin{aligned} R_{\text{mismatch}} &= H(J) + H(p_J||p_I) \\ &= -\sum_{i\in\mathcal{I}} f_Y(c_i)\Delta\log(f_Y(c_i)\Delta) + \sum_{i\in\mathcal{I}} f_Y(c_i)\Delta\log(\frac{f_Y(c_i)}{f_X(c_i)}) \\ &= -\sum_{i\in\mathcal{I}} f_Y(c_i)\Delta\log(f_X(c_i)/g_C) \\ &\approx -\int f_Y(y)\log(f_X(y)/g_C)dy \\ &= \log(g_C) - \int f_Y(y)\log(f_X(y))dy \end{aligned} \tag{7.36}$$

Now let us consider the special case that the variable X is Gaussian, with a variance

σ^2 that is identical to the variance of Y:

$$\begin{aligned} R_{\text{mismatch}} &= \log(g_C) - \int f_Y(y) \log(\frac{1}{\sqrt{2\pi\sigma^2}} \mathrm{e}^{\frac{-y^2}{2\sigma^2}}) dy \\ &= -\frac{1}{2}\log(12D) + \frac{1}{2}\log(2\pi\sigma^2) + \int f_Y(y) \frac{y^2}{2\sigma^2} dy \\ &= \frac{1}{2}\log(\frac{\sigma^2}{D}) + \frac{1}{2}\log(\frac{2\pi\mathrm{e}}{12}). \end{aligned} \tag{7.37}$$

This result is independent of f_Y, optimal for Gaussian f_Y with $f_Y = f_X$, and $\frac{1}{2}\log_2(\frac{2\pi\mathrm{e}}{12}) \approx 0.25$ bit above the rate-distortion function for a Gaussian. In other words, the rate of this encoder is independent of the statistics of the input variable Y, other than that we must ensure that the variance of Y is σ^2. Thus, we have constructed a high-rate robust encoder that provides a fixed mean rate for a given distortion, independently of the probability density of the input variable! The mean rate is identical to that of the Gaussian variable X with variance σ^2.

Comparison to the Rate-Distortion Function

It is interesting to compare the relation between rate and distortion obtained with the high-rate theory for the scalar case with those obtained using the rate-distortion function approach discussed in chapter 6, both for the squared-error criterion. As it turns out, the equations look very similar when the constrained-entropy case is considered for the high-rate theory. We use the Shannon lower bound for the rate-distortion function given by equation 6.37. We can compare this with the high-rate, constrained-entropy expression for the scalar quantizer of equation 7.33. Combining equations 7.33 and 6.37 results in

$$H(I) - R(D) \leq \frac{1}{2}\log(\frac{2\pi\mathrm{e}}{12}). \tag{7.38}$$

The inequality becomes an equality in the case of a Gaussian source and the squared-error criterion, where the Shannon lower bound equals the rate-distortion function. Thus, we see that at a given distortion and in the high-rate approximation, the optimal constrained-entropy scalar quantizer can perform within $\frac{1}{2}\log_2(\frac{2\pi\mathrm{e}}{12}) \approx 0.25$ bits of the rate-distortion function[1].

The above results show that, for many practical purposes, the performance of scalar quantizers in combination with efficient lossless coding is sufficient if the rates are high and the samples of the process are identically distributed. The independence condition can be satisfied by applying suitable transformations of the signal; such transformations is discussed in chapters 9 and 10.

7.4 High-Rate Vector Quantization

We now move on to considering high-rate theory for the vector quantization case. Many of the procedures in this section form a generalization of the methods seen in section 7.3.

[1] It is more difficult to obtain such bounds for the unrestricted (i.e., not high-rate) case. It has been shown that the best scalar quantizer always performs within about 0.75 bit of the rate-distortion limit [30].

For clarity, we re-list the assumptions for the vector-quantization case. We assume that the quantizers are regular and that the centroids are located so as to provide minimum distortion. We use the assumption that the distortion criterion is of the form given by equation 7.12. It is also assumed that the unbounded cells are not of significance in the derivations. In general, at a given rate per dimension the fraction of the cells that are unbounded increases with the dimensionality of the quantizer. Thus, this assumption is not as natural as it was in the scalar case. This issue is addressed in [31].

7.4.1 Distortion and Centroid Density

We first determine the average distortion for each cell, on a per dimension basis:

$$\begin{aligned} D_i &= \frac{1}{k}\frac{\int_{\mathtt{V}_i} f_{X^k}(x^k)\|x^k - \mathcal{Q}(x^k)\|_r dx^k}{\int_{\mathtt{V}_i} f_{X^k}(x^k)dx^k} \\ &\approx \frac{1}{kV_i}\int_{\mathtt{V}_i} \|x^k - c_i^k\|_r dx^k \\ &= V_i^{\frac{r}{k}}\ \frac{1}{k}V_i^{-\frac{r+k}{k}}\int_{\mathtt{V}_i} \|x^k - c_i^k\|_r dx^k \\ &= V_i^{\frac{r}{k}}\ C(r,k,\mathcal{G}(i)), \end{aligned} \tag{7.39}$$

where the volume of the cell $\mathtt{V}_i$ is V_i, c_i^k is the centroid and $C(r,k,\mathcal{G}(i))$ is the **normalized moment of inertia** or **coefficient of quantization**:

$$C(r,k,\mathcal{G}(i)) = \frac{1}{k}\frac{1}{V_i^{\frac{k+r}{k}}}\int_{\mathtt{V}_i} \|x^k - c_i^k\|_r dx^k, \tag{7.40}$$

where $\mathcal{G}(i)$ indicates the geometry of cell i. As was mentioned, we assume that the centroid c_i^k is always located so as to minimize the normalized moment of inertia. The coefficient of quantization is then a scale-independent and rotation-independent parameter that is dependent only on r, k, and the geometry of the cell. In general, an objective of quantization is to find cell shapes that minimize the value of $C(r,k,\mathcal{G}(i))$. If we consider only a single cell, the optimal geometry that minimizes the normalized moment of inertia is a hypersphere. With increasing k (and for the trivial case $k = 1$) practical tessellations of cells facilitate shapes that approach a hypersphere.

To evaluate quantizer performance we must know the mean distortion per dimension over all cells. To obtain this, we make the additional assumption that, if $f_{X^k}(x^k)$ is sufficiently smooth and the number of centroids is sufficiently large, then the cell geometry varies slowly with the cell index and we can modify the notation for the geometry

$$\mathcal{G}(i) \ \rightarrow\ \mathcal{G}(x^k). \tag{7.41}$$

The function $C(r,k,\mathcal{G}(x^k))$ of x^k is referred to as the **inertial profile** [32]. We then

obtain for the mean distortion per dimension:

$$\begin{aligned} D &= \sum_{i\in\mathcal{I}} p_I(i) D_i \\ &\approx \sum_{i\in\mathcal{I}} p_I(i) V_i^{\frac{r}{k}} C(r,k,\mathcal{G}(i)) \\ &\approx \int_{\mathbb{R}^k} f_{X^k}(x^k) C(r,k,\mathcal{G}(x^k)) g_{C^k}(x^k)^{-\frac{r}{k}} dx^k. \end{aligned} \tag{7.42}$$

It is commonly assumed that for the optimal geometry the normalized moment of inertia does not vary with the cell index, i.e.,

$$\mathcal{G}_{\text{opt}}(x^k) = \mathcal{G}_{\text{opt},k}. \tag{7.43}$$

This property is called **Gersho's conjecture** [33]. If Gersho's conjecture holds, equation 7.42 can be written as

$$D \approx C(r,k,\mathcal{G}_{\text{opt},k}) \int_{\mathbb{R}^k} f_{X^k}(x^k) g_{C^k}(x^k)^{-\frac{r}{k}} dx^k. \tag{7.44}$$

Example 7.3: Values for the normalized moment of inertia for $r = 2$
Assuming the Gersho conjecture is correct, it is interesting to find the normalized moment of inertia of the optimal geometry for different dimensions. For the one-dimensional case, geometry plays no role, and we have that the normalized moment of inertia of a cell is fixed at

$$C(r=2,k=1,\mathcal{G}(i)) = C(r=2,k=1) = \frac{1}{\Delta_i^{\frac{3}{1}}} \int_{-\Delta/2}^{\Delta/2} x^2 dx = \frac{1}{12} \approx 0.0833,$$

a result we could have seen immediately from equation 7.13.

For the two-dimensional case, the optimal geometry $g_{\text{opt},k=2}$ is hexagonal:

$$C(r=2,k=2,\mathcal{G}_{\text{opt},k=2}) = \frac{5}{36\sqrt{3}} \approx 0.0802.$$

For the case $k \to \infty$ the optimal geometry g_{opt} results in

$$\lim_{k\to\infty} C(r=2,k=\infty,\mathcal{G}_{\text{opt},k=\infty}) = (2\pi\text{e})^{-1} \approx 0.0585.$$

Equation 7.39 shows that for a fixed cell volume, the distortion depends only on the normalized moment of inertia. Let us consider a k-dimensional cell for $k \to \infty$. For the optimal geometry the normalized moment of inertia is $(2\pi\text{e})^{-1}$. For hypercubic cells, the normalized moment of inertia is $1/12$. The difference in the normalized moment of inertia for the hypercube and optimal geometries results in a difference in distortion per dimension of 1.53 dB.

Example 7.4: The inertial profile of a scalar quantizer for $r = 2$
Consider a k-dimensional vector with independent components. It is straightforward to write down the inertial profile for a scalar quantizer for $r = 2$. The

inertial profile of the k-dimensional vector quantizer that is the product of k scalar quantizers is

$$\begin{aligned} C(2,k,\mathcal{G}(x^k)) &= \frac{1}{k}\frac{1}{V(x^k)^{\frac{k+2}{k}}}\int_{\mathbb{V}(x^k)}\sum_{i=1}^{k} y_i^2 dy^k \\ &= \frac{1}{12k}\prod_{j=1}^{k} g_{C_j}(x_j)^{\frac{2}{k}}\sum_{i=1}^{k}\frac{1}{g_{C_i}(x_i)^2} \\ &= C(2,1,\mathcal{G}_1)\frac{1}{k}\sum_{i=1}^{k}\frac{\prod_{j=1}^{k} g_{C_j}(x_j)^{\frac{2}{k}}}{g_{C_i}(x_i)^2}. \end{aligned}$$

We can use this inertial profile to compute directly the distortion per dimension of a k-dimensional quantizer that uses the cells of the scalar quantizer:

$$\begin{aligned} D_{\text{scalar}} &= \int_{\mathbb{R}^k} C(2,k,\mathcal{G}(x^k))\prod_{i=1}^{k}(f_{X_i}(x_i)g_{C_i}(x_i)^{\frac{-2}{k}})dx^k \\ &= \int_{\mathbb{R}^k}\frac{1}{12k}\sum_{l=1}^{k}\frac{\prod_{j=1}^{k} g_{C_j}(x_j)^{\frac{2}{k}}}{g_{C_l}(x_l)^2}\prod_{i=1}^{k}(f_{X_i}(x_i)g_{C_i}(x_i)^{\frac{-2}{k}})dx^k \\ &= \frac{1}{12k}\sum_{i=1}^{k}\int_{\mathbb{R}} g_{C_i}(x_i)^{-2} f_{X_i}(x_i)dx_i \end{aligned}$$

which is, as expected, simply the distortion as provided by equation 7.15 averaged over the dimensions.

7.4.2 Constrained Resolution

The resolution constraint limits the size of the codebook. In the high-rate vector-quantization case we can write this as

$$\int_{\mathbb{R}^k} g_{C^k}(x^k)dx^k = N, \tag{7.45}$$

where N is the codebook size, i.e., the number of centroids. Assuming Gersho's conjecture holds, the goal is to find an expression for $g_{C^k}(x^k)$ that results in a minimum for equation 7.44, under constraint 7.45. Using the familiar method of Lagrange multipliers, we write the augmented distortion criterion

$$\eta = C(r,k,\mathcal{G}_{\text{opt},k})\int_{\mathbb{R}^k}(f_{X^k}(x^k)g_{C^k}(x^k)^{-\frac{r}{k}} + \lambda g_{C^k}(x^k))dx^k. \tag{7.46}$$

The Euler-Lagrange equation for this optimization is

$$\frac{-r}{k}f_{X^k}(x^k)g_{C^k}(x^k)^{-\frac{r+k}{k}} + \lambda = 0. \tag{7.47}$$

Equations 7.47 and 7.45 lead to the **optimal constrained-resolution centroid density**

$$g_{C^k}(x^k) = N\frac{f_{X^k}(x^k)^{\frac{k}{k+r}}}{\int_{\mathbb{R}^k} f_{X^k}(x^k)^{\frac{k}{k+r}}dx^k}. \tag{7.48}$$

The associated distortion is obtained by inserting equation 7.48 into equation 7.44:

$$\begin{aligned} D_{CR}(r,k,\mathcal{G}_{\text{opt},k},N) &\approx C(r,k,\mathcal{G}_{\text{opt},k})N^{-\frac{r}{k}} \int_{\mathbb{R}^k} f_{X^k}(x^k) \left(\frac{f_{X^k}(x^k)^{\frac{k}{k+r}}}{\int_{\mathbb{R}^k} f_{X^k}(x^k)^{\frac{k}{k+r}} dx^k} \right)^{-\frac{r}{k}} dx^k \\ &= C(r,k,\mathcal{G}_{\text{opt},k})N^{-\frac{r}{k}} \left(\int_{\mathbb{R}^k} f_{X^k}(x^k)^{\frac{k}{k+r}} dx^k \right)^{\frac{r+k}{k}} \\ &= C(r,k,\mathcal{G}_{\text{opt},k})N^{-\frac{r}{k}} \langle f_{X^k}(x^k) \rangle_{\frac{k}{r+k}}, \end{aligned} \tag{7.49}$$

where the subscript in D_{CR} indicates "constrained resolution" and where[2]

$$\langle f_{X^k}(x^k) \rangle_\alpha \equiv \left(\int_{\mathbb{R}^k} f_{X^k}(x^k)^\alpha dx^k \right)^{\frac{1}{\alpha}}. \tag{7.50}$$

Considering that the logarithm of the cardinality of the codebook is the rate, R, we can write equation 7.49 in the form

$$D_{CR}(r,k,\mathcal{G}_{\text{opt},k},R) \approx C(r,k,\mathcal{G}_{\text{opt},k}) \mathrm{e}^{-\frac{r}{k}R} \langle f_{X^k}(x^k) \rangle_{\frac{k}{r+k}}, \tag{7.51}$$

where the rate is given in nats. Equation 7.51 provides the relation between distortion and rate for the high-rate constrained-resolution case.

An interesting fact is that the distortion per cell multiplied by the probability of the cell is constant [34]. This so-called **equidistortion principle** for constrained resolution and high rate can be shown as follows:

$$\begin{aligned} D_i\, p_{\mathcal{I}}(i) &= C(r,k,\mathcal{G}_{\text{opt},k}) V_i^{\frac{r}{k}}\ f_{X^k}(c_i^k) V_i \\ &= C(r,k,\mathcal{G}_{\text{opt},k}) f_{X^k}(c_i^k)\, g_{C^k}(c_i^k)^{-\frac{r+k}{k}} \\ &= C(r,k,\mathcal{G}_{\text{opt},k}) f_{X^k}(c_i^k) N^{-\frac{r+k}{k}} f_{X^k}(c_i^k)^{-1} \left(\int_{\mathbb{R}^k} f_{X^k}(x^k)^{\frac{k}{r+k}} dx^k \right)^{\frac{r+k}{k}} \\ &= \frac{1}{N} D_{CR}, \end{aligned} \tag{7.52}$$

where we used equations 7.39 and 7.49. Thus, for the optimal constrained-resolution quantizer, each cell contributes equally to the mean distortion.

7.4.3 Constrained Entropy

We first relate the (first-order) entropy of the indices to the differential entropy of the source. This derivation is again a straightforward generalization of the scalar case:

$$\begin{aligned} H(I) &= -\sum_{i \in \mathcal{I}} p_I(i) \log(p_I(i)) \\ &= -\sum_{i \in \mathcal{I}} f_{X^k}(c_i^k) V_i \log(f_{X^k}(c_i^k) V_i) \\ &\approx -\int_{\mathbb{R}^k} f_{X^k}(x^k) \log(\frac{f_{X^k}(x^k)}{g_{C^k}(x^k)}) dx^k. \end{aligned} \tag{7.53}$$

[2] The expression of equation 7.50 is a **seminorm**, since it is nonnegative and satisfies the conditions: $\langle b f(\cdot) \rangle_\alpha = |b| \langle f(\cdot) \rangle_\alpha$ and $\langle f(\cdot) \rangle_\alpha + \langle g(\cdot) \rangle_\alpha \geq \langle f(\cdot) + g(\cdot) \rangle_\alpha$.

From equation 7.53 it follows that we can write

$$H(I) \approx -h(f_{X^k} \| g_{C^k}/N) + \log(N), \tag{7.54}$$

where N is the number or centroids. However, as is seen below, this number is not finite for the optimal quantizer, rendering this approach more difficult. Alternatively, we can write

$$H(I) \approx h(X^k) + \mathrm{E}[\log(g_{C^k}(X^k))]. \tag{7.55}$$

From equation 7.55, and the fact that $h(X^k)$ is not affected by the quantizer, we see that a constraint on the first-order index entropy is equivalent to a constraint on $\mathrm{E}[\log(g_{C^k}(X^k))]$:

$$\mathrm{E}[\log(g_{C^k}(X^k))] = \int_{\mathbb{R}^k} f_{X^k}(x^k) \log(g_{C^k}(X^k)) dx^k = b, \tag{7.56}$$

where b is a constant. Assuming that Gersho's conjecture holds, the Lagrange multiplier method can be used to extend the distortion criterion 7.44 with the constraint to obtain

$$\eta = C(r, k, \mathcal{G}_{\mathrm{opt},k}) \int_{\mathbb{R}^k} f_{X^k}(x^k)(g_{C^k}(x^k)^{-\frac{r}{k}} + \lambda' \log(g_{C^k}(x^k)))dx^k, \tag{7.57}$$

which renders the Euler-Lagrange equation

$$g_{C^k}(x^k)^{-\frac{r+k}{k}} + \lambda g_{C^k}(x^k)^{-1} = 0. \tag{7.58}$$

An important consequence is that the *centroid density* $g_{C^k}(x^k)$ *is constant for high-resolution constrained-entropy quantization.* In other words, the centroid density does not depend on the probability density $f_{X^k}(x^k)$.

The solution of the Euler-Lagrange equations, constraint 7.56, and equation 7.55 is

$$g_{C^k}(x^k) = \mathrm{e}^{H(I)-h(X^k)}, \tag{7.59}$$

where we assumed the entropy and differential entropy to have been specified in nats. Thus, the entropy-constrained optimization results in a vector quantizer with a uniform reconstruction point density, which depends only on the constrained index entropy and the differential entropy of the random variable.

Inserting the density of equation 7.59 into the expression for the mean distortion per dimension, 7.44, we obtain for the mean distortion, on a per dimension basis:

$$D_{CE}(r, k, \mathcal{G}_{\mathrm{opt},k}, H(I)) \approx C(r, k, \mathcal{G}_{\mathrm{opt},k})\mathrm{e}^{-\frac{r}{k}\left(H(I)-h(X^k)\right)}. \tag{7.60}$$

It is useful to contrast our results for the constrained-resolution quantizer with that of the constrained-entropy. In the first case, the contribution of each cell to the mean distortion is constant and the cell size varies. In the second case, the size of the cells is constant and the contribution to the mean distortion per cell varies.

Using the Gaussian case, and the fact that it maximizes the differential entropy given the variance, it is possible to create an upper bound for the high-rate rate-distortion function similarly to the scalar case discussed in section 7.3.3. More-over, the same procedure can also be used to design robust vector encoders.

7.4.4 Comparison to Rate-Distortion Function

It is interesting to compare our high-rate results for coding k-dimensional vectors to the order-k rate-distortion function. It is natural to compare the result for constrained entropy to the rate-distortion function first, since the constrained-entropy relation is expressed in terms of rate, here represented as the first-order entropy of the indices, $H(I)$, and differential entropy, $h(X^k)$. Both measures also appear in the Shannon lower bound for the rate-distortion function. We characterize the rate-distortion function with the order-k Shannon lower bound, which is (cf. equation 6.37)

$$R(D) \geq \frac{1}{k} h(X^k) - \frac{1}{2} \log(2\pi \mathrm{e} D), \tag{7.61}$$

where we have written D for the distortion *per dimension*, and where $r = 2$. Inequality 7.61 becomes an equality for the Gaussian case. We can write equation 7.60 as

$$\frac{1}{k} H(I) \approx \frac{1}{k} h(X^k) - \frac{1}{r} \log(\frac{D_{CE}}{C(2, k, \mathcal{G}_{\mathrm{opt},k})}). \tag{7.62}$$

Subtracting equation 7.61 for the special case that $r = 2$ from equation 7.62 results in

$$\frac{1}{k} H(I) - R(D) \leq \frac{1}{2} \log(2\pi \mathrm{e}\ C(2, k, \mathcal{G}_{\mathrm{opt},k})), \tag{7.63}$$

which is an equality for the Gaussian case. As we saw in section 7.3.3, the right-hand side of equation 7.63 is about 0.25 bits for the scalar quantizer (i.e., $k = 1$). This corresponds to the case where the cells are hypercubes. The difference decreases monotonically with increasing dimension.

Assuming Gersho's conjecture is correct, the rate-distortion theorem (see section 6.2.2) tells us that the right-hand side of equation 7.63 should vanish when $k \to \infty$. The high-rate constrained-entropy quantizer then reaches the Shannon lower bound on the rate-distortion function. This implies that the Shannon lower bound must be the rate-distortion function for high rate. The coefficient of quantization $C(2, k, \mathcal{G}_{\mathrm{opt},k})$ must converge to $(2\pi \mathrm{e})^{-1}$ when $k \to \infty$ if the normalized moment of inertia $C(2, k, \mathcal{G}_{\mathrm{opt},k})$, is selected to be the infimum over the set of admissible moments (moments that correspond to cell shapes that can be arranged to cover $\mathbb{R}^k$ without overlap) [35]. For hyperspherical cells, $C(2, k, \mathcal{G})$ also converges to $(2\pi \mathrm{e})^{-1}$ for large k (which is not admissible at finite k). By suitable averaging $C(r, k, \mathcal{G})$ can also be defined for randomly distributed centroids. Interestingly, it can be shown that such a "random lattice" also results in a convergence of $C(r, k, \mathcal{G}_{\mathrm{opt},k})$ to $(2\pi \mathrm{e})^{-1}$ (see problem 7). Naturally, the moment of inertia converges slower to this value for the random lattice than for the spherical cell case.

Next, let us consider the constrained-resolution case. Again, we discuss the result for $k \to \infty$. We rewrite equation 7.51 in the form

$$\frac{r}{k} R = -\log(\frac{D_{CR}}{C(r, k, \mathcal{G}_{\mathrm{opt},k})}) + \log(\langle f_{X^k}(x^k) \rangle_{\frac{k}{r+k}}). \tag{7.64}$$

We can find a bound on the rate. To obtain a simple analysis we make again the assumption that the effect of a fraction of the cells being unbounded can be neglected. This implies a requirement on codebook size that grows very rapidly with increasing dimension, a condition that is not always satisfied. We first write equation 7.64 as

$$\frac{r}{k} R = \quad -\log(\frac{D_{CR}}{C(r, k, \mathcal{G}_{\mathrm{opt},k})}) + \frac{r+k}{k} \log(\int_{\mathbb{R}^k} f_{X^k}(x^k) f_{X^k}(x^k)^{\frac{-r}{k+r}} dx^k). \tag{7.65}$$

Let us consider a density f_{X^k} that has (or can be approximated in the present context with) a finite support range. We note that for sufficiently large k the factor $f_{X^k}(x^k)^{\frac{-r}{k+r}}$ is near unity in the support range of $f_{X^k}(x^k)$. We then write this term in a convenient form,

$$f_{X^k}(x^k)^{\frac{-r}{k+r}} = \mathrm{e}^{\frac{-r}{k+r}\log(f_{X^k}(x^k))}. \tag{7.66}$$

and make a Taylor expansion of the exponent. We note that all terms of the Taylor expansion are positive. Retaining only the first-order term, the rate distortion relation ca be bound by

$$\begin{aligned}
\frac{r}{k}R &= -\log(\frac{D_{CR}}{C(r,k,\mathcal{G}_{\mathrm{opt},k})}) + \frac{r+k}{k}\log(1 - \frac{r}{k+r}\int_{\mathbb{R}^k} f_{X^k}(x^k)\log(f_{X^k}(x^k))dx^k) \\
&< -\log(\frac{D_{CR}}{C(r,k,\mathcal{G}_{\mathrm{opt},k})}) + \frac{r+k}{k}\log(1 + \frac{r}{k+r}h(X^k)) \\
&= -\log(\frac{D_{CR}}{C(r,k,\mathcal{G}_{\mathrm{opt},k})}) + \frac{r}{k}h(X^k),
\end{aligned} \tag{7.67}$$

which is identical to the the constrained-entropy result of equation 7.62. The bound asymptotically becomes an equality with increasing k. Thus, if k is sufficiently large, and if, despite the large k, the effect of boundary cells can be neglected, then the high-rate constrained-resolution quantizer also reaches the Shannon lower bound on the rate-distortion function.

If we analyze our results, we see some seemingly contradictory results. We noted that in proving the rate-distortion theorem we implicitly assume that the codewords have equal probability (since they are in the distortion-typical set). In the constrained-resolution case, assuming that k is sufficiently large, the codewords also obtain identical probability (since the centroid density becomes equal to the data density). In contrast, we use equal cells for the constrained-entropy case and, thus, have unequal codeword probabilities (which we account for by lossless coding). How can these different situations lead to the same relation between distortion and rate? The answer lies, naturally, in the high dimensionality of the data space. In section 6.2.5 we noted that, as a result of equipartition, for high dimension the data density is uniform in its support region. Thus, with increasing k, the sizes of the cells in a constrained-resolution coder become constant in this support region. Similarly, the probabilities of the cells of a constrained-entropy coder also become constant in this support region. In other words, the constrained-resolution and constrained-entropy quantizers approach similarity with increasing dimension.

7.5 Vector Quantization versus Scalar Quantization

In the previous sections, we have obtained expressions for the distortion per dimension for both the constrained resolution and the constrained entropy cases. We can use these expressions as a basis for comparing the performance of optimal vector quantizers and scalar quantizers for a given stationary source, X_i, when the high-rate approximation is valid.

We denote an arbitrary block of k consecutive samples by X_i^k. We make the comparison by simply considering the ratio of the distortion for the scalar quantizer and the

distortion for the optimal vector quantizer and then decomposing the expression into contributions due to the coefficient of quantization, the product of the marginal densities and the joint density, and a "remainder" term depending on the marginal density and the product of the marginal densities. The analysis follows the basic approach described in [36] with additional insights from [32]. We can write the ratio as a multiplication of three factors:

$$\begin{aligned} RD &\equiv \frac{D(\text{dimension} = 1)}{D(\text{dimension} = k)} \\ &= RD_{\text{sf}}(C(r,1,\mathcal{G}_1), C(r,k,\mathcal{G}_{\text{opt},k}))\ RD_{\text{mem}}(f_{X_i^k}, \prod_{i=1}^{k} f_{X_i}) \\ &\qquad RD_{\text{shape}}(f_{X_i}, \prod_{i=1}^{k} f_{X_i}) \end{aligned} \tag{7.68}$$

All factors contribute to the advantage of vector quantization over scalar quantization. The factor $RD_{\text{sf}}(\cdot,\cdot)$ is referred to as the **space-filling advantage** and results from the greater freedom to select cell shapes in higher dimensions. However, it is important to note that the name space-filling advantage is somewhat misleading since it accounts only for the advantage of the optimal cell geometry $\mathcal{G}_{\text{opt},k}$ over that of a k-dimensional hypercubic cell. In a practical constrained-resolution quantizer the cells can be, for example, hyperrectangles; the advantage that a hypercubic cell has over a hyperrectangle cell is not included in the space-filling advantage (it is part of the shape advantage).

The factor $RD_{\text{mem}}(\cdot,\cdot)$ represents the **memory advantage**, which accounts for the dependencies between the samples of the process X_i. The memory advantage factor becomes unity if the vector components in X^k are independent, i.e., if $\prod_{i=1}^{k} f_{X_i}(x_i) = f_{X_i^k}(x_i^k)$.

The remainder factor $RD_{\text{shape}}(\cdot,\cdot)$ is called the **shape advantage**. This advantage includes the advantage that hypercubic cells have over hyperrectangles, and the fact that the k-dimensional centroid density obtained from scalar quantization of a stationary signal is not optimal for hypercubes (the exponential of the density is nonoptimal, see equation 7.48). The shape advantage does not disappear (numerically become unity) when the data are independent. For the constrained-entropy case, the centroid density is uniform and all scalar quantization results in hypercubic cells. Thus, the shape advantage does not exist for the constrained-entropy case.

7.5.1 Constrained Resolution

We start with the constrained-rate case. We consider a rate $\log(N)$ per sample, which corresponds to a codebook of size N for the scalar case and a codebook of size N^k for the vector case. From equation 7.49 we see that the ratio, RD_{CR}, between the distortion

of a one-dimensional and a k-dimensional quantizer can be written as

$$\begin{aligned} RD_{CR} &= \frac{D_{CR}(r,1,\mathcal{G}_1,N)}{D_{CR}(r,k,\mathcal{G}_{\mathrm{opt},k},N^k)} \\ &= \frac{C(r,1,\mathcal{G}_1)\ \langle f_{X_i}\rangle_{\frac{1}{r+1}}}{C(r,k,\mathcal{G}_{\mathrm{opt},k})\ \langle f_{X_i^k}\rangle_{\frac{k}{r+k}}} \\ &= \frac{C(r,1,\mathcal{G}_1)}{C(r,k,\mathcal{G}_{\mathrm{opt},k})}\ \frac{\langle \prod_{i=1}^k f_{X_i}\rangle_{\frac{k}{r+k}}}{\langle f_{X_i^k}\rangle_{\frac{k}{r+k}}}\ \frac{\langle f_{X_i}\rangle_{\frac{1}{r+1}}}{\langle \prod_{i=1}^k f_{X_i}\rangle_{\frac{k}{r+k}}}. \end{aligned} \tag{7.69}$$

The space-filling advantage is thus

$$RD_{CR,\mathrm{sf}} = \frac{C(r,1,\mathcal{G}_1)}{C(r,k,\mathcal{G}_{\mathrm{opt},k})}. \tag{7.70}$$

The space filling advantage is always less than about 1.5 dB (corresponding to 0.25 bits), reaching its maximum value when the vector quantizer dimension approaches infinity. Optimal vector quantization reaches the rate-distortion bound when the dimension goes to infinity, and this result thus coincides with that obtained in the comparison of the scalar quantizer performance and the rate-distortion bound in section 7.3.3.

Often the most significant factor for the constrained-resolution case is the memory advantage factor,

$$RD_{CR,\mathrm{mem}} = \frac{\langle \prod_{i=1}^k f_{X_i}\rangle_{\frac{k}{r+k}}}{\langle f_{X_i^k}\rangle_{\frac{k}{r+k}}}. \tag{7.71}$$

The shape advantage factor is

$$\begin{aligned} RD_{CR,\mathrm{shape}} &= \frac{\langle f_{X_i}\rangle_{\frac{1}{r+1}}}{\langle \prod_{i=1}^k f_{X_i}\rangle_{\frac{k}{r+k}}} \\ &= \frac{\langle f_{X_i}\rangle_{\frac{1}{r+1}}}{\langle f_{X_i}^k\rangle_{\frac{1}{r+k}}}. \end{aligned} \tag{7.72}$$

The shape advantage factor becomes unity (i.e., the advantage vanishes) for the "boxcar" (uniform over finite support region) marginal density function and increases with dimension for other densities. The shape advantage can be up to 2.8 dB for a Gaussian density, more than that for a Laplace density, and again more (up to about 12 dB) for the two-sided gamma density (in each case the advantage increases monotonically with k) [36]. Long "tails" of the density result in a strong shape advantage for vector quantization.

The above description did not explicitly account for the fact that the cells of a scalar quantizer are not hypercubes. We can factor the shape advantage (the remainder factor) into advantages due to removal of so-called **oblongitis** [32], the advantage of hypercubic cell shape over hyperrectangular cell shape, and a remainder term related to density. We call these advantages the **cubic advantage** and **density advantage**:

$$RD_{CR,\mathrm{shape}} = RD_{CR,\mathrm{cubic}} RD_{CR,\mathrm{density}}. \tag{7.73}$$

Let us consider the distortion of a quantizer with the same centroid distribution as a scalar quantizer, but with cubic cells. The inertial profile of such a quantizer is

$C(2,k,\mathcal{G}(x^k)) = C(2,k,\mathcal{G}_{\text{cubic},k}))$. In the general case we obtain, based on equations 7.42 and 7.48,

$$\begin{aligned} D_{CR,\text{cubic}} &= \int_{R^k} C(r,k,\mathcal{G}(x^k)) \prod_{i=1}^{k} (f_{X_i}(x_i) g_{C_i}(x_i)^{\frac{-r}{k}}) dx^k \\ &= C(r,k,\mathcal{G}_{\text{cubic},k}) \int_{\mathbb{R}^k} \prod_{i=1}^{k} (f_{X_i}(x_i) g_{C_i}(x_i)^{\frac{-r}{k}}) dx^k \\ &= C(r,k,\mathcal{G}_{\text{cubic},k}) (\int_R g_{C_i}(x_i)^{\frac{-r}{k}} f_{X_i}(x_i) dx_i)^k \\ &= C(r,k,\mathcal{G}_{\text{cubic},k}) \frac{1}{N^r} (\int_R f_{X_i}(x_i)^{\frac{1}{1+r}} dx_i)^r (\int_R f_{X_i}(x_i)^{\frac{(k+kr-r)}{k+kr}} dx_i)^k \end{aligned} \quad (7.74)$$

From equation 7.74 and equation 7.44 for the case $k = 1$, we see that the advantage of having cubic cells rather than the rectangular cells associated with scalar quantization is

$$RD_{CR,\text{cubic}} = \frac{C(r,1,\mathcal{G}_1)}{C(r,k,\mathcal{G}_{\text{cubic},k})} \frac{\int_R f_{X_i}(x_i)^{\frac{1}{1+r}} dx_i}{(\int_R f_{X_i}(x_i)^{\frac{(k+kr-r)}{k+kr}} dx_i)^k}. \quad (7.75)$$

For the constrained-resolution case the cell shape advantage of an ideal vector quantizer over a scalar quantizer consists of the multiplication of the space-filling and cubic-cell advantages:

$$RD_{CR,\text{cellshape}} = RD_{CR,\text{sf}} RD_{CR,\text{cubic}} \quad (7.76)$$

From the factorization 7.73 it follows that the advantage of vector quantization related to the centroid density is

$$RD_{CR,\text{density}} = \frac{C(r,k,\mathcal{G}_{\text{cubic},k})}{C(r,1,\mathcal{G}_1)} \frac{(\int_R f_{X_i}(x_i)^{\frac{(k+kr-r)}{k+kr}} dx_i)^k (\int_R f_{X_i}(x_i)^{\frac{1}{1+r}} dx_i)^r}{\langle f_{X_i}^k \rangle_{\frac{1}{r+k}}}. \quad (7.77)$$

For the case that $r = 2$ we have that $C(2,k,\mathcal{G}_{\text{cubic},k})) = \frac{1}{12}$ does not depend on dimensionality. The density advantage reduces then to

$$RD_{CR,\text{density}}|_{r=2} = \frac{(\int_R f_{X_i}(x_i)^{\frac{(3k-2)}{3k}} dx_i)^k (\int_R f_{X_i}(x_i)^{\frac{1}{3}} dx_i)^2}{\langle f_{X_i}^k \rangle_{\frac{1}{2+k}}}. \quad (7.78)$$

7.5.2 Constrained Entropy

We now compare vector and scalar quantizers for a stationary process X_i for the constrained-entropy case. We denote the constrained first-order entropy per sample

by H_1. Using equation 7.60 we can write

$$\begin{aligned} RD_{CE} &= \frac{D_{CE}(r,1,\mathcal{G}_1,H_1)}{D_{CE}(r,k,\mathcal{G}_{\text{opt},k},kH_1)} \\ &= \frac{C(r,1,\mathcal{G}_1)\ \mathrm{e}^{-r(H_1-h_1(X_i))}}{C(r,k,\mathcal{G}_{\text{opt},k})\ \mathrm{e}^{-\frac{r}{k}(kH_1-kh_k(X_i))}} \\ &= \frac{C(r,1,\mathcal{G}_1)}{C(r,k,\mathcal{G}_{\text{opt},k})}\ \frac{\mathrm{e}^{rh_1(X_i)}}{\mathrm{e}^{rh_k(X_i)}}. \end{aligned} \tag{7.79}$$

We can draw a number of conclusions from equation 7.79. First, we note that the space-filling advantage, (the advantage that disappears when the vector quantizer is required to use hypercubes as cells) is identical for the constrained-entropy case and the constrained-resolution case.

Since we used the *first-order* index entropy, $H_1(I)$, the constrained-entropy case does have a memory advantage. We note that the differential entropy $kh(X_i)$ and, thus, $h(X_i)$ is determined by the product of the marginal densities $\prod_{i=1}^{k} f_{X_i}$ and that $h(X_i^k)$ is determined by the joint density $f_{X_i^k}$. This means that we can interpret 7.79 without requiring a shape factor. The memory factor is

$$RD_{CE,\text{mem}} = \mathrm{e}^{r(h(X_i)-\frac{1}{k}h(X_i^k))} \tag{7.80}$$

and there is no shape advantage. This is intuitively reasonable since the shape advantage is related to the constraint on variation of the cell sizes that scalar quantizers impose. In a high-rate entropy-constrained quantizer, the density of the centroids is uniform and the constraint has no effect.

Finally, we note that the memory advantage for the constrained-entropy case is a function of the redundancy for a block of length k,

$$\rho_k(X_i) = h(X_i) - \frac{1}{k}h(X_i^k). \tag{7.81}$$

Example 7.5: Scalar quantization with joint index encoding
It is interesting to consider the role of lossless coding for constrained-entropy quantization. First, we consider scalar quantization with joint lossless coding for k indices. We note that this is equivalent to constrained-entropy vector quantization with a cubic cell shape. A conventional k-dimensional constrained-entropy vector quantizer has only a space-filling advantage over such a coder. In the constrained-entropy case, the lossless coding of vectors accounts for the dependencies between the components of the random vector. As a result, the lossless coder generally has high computational or storage complexity. This contrasts with the constrained-resolution vector quantizer, where the variation in the centroid density accounts for the dependencies between the vector components. In other words, in a constrained-resolution vector quantizer the computational complexity of the quantizer generally increases with the vector dimensionality; in a constrained-entropy vector quantizer the increase in the computational complexity of the lossless coder generally dominates.

7.5.3 Vector Quantization, Modeling, and Source Coding

In sections 7.5.1 and 7.5.2, we studied the advantage of vector quantization, of possibly finite dimension, over scalar quantization. It is straightforward to generalize the problem and compare a vector quantizer of higher vector dimension with a vector quantizer of lower dimension. The expressions for this comparison are very similar. In general, performance decreases with decreasing vector dimension, and this decrease can be attributed to contributions resulting from the space filling advantage, the memory advantage, and the shape advantage (consisting of the cubic and density advantages). The shape advantage of higher dimensionality exists for resolution-constrained vector quantization only.

It is common that the computational complexity required for the codebook search (or for the lossless coder) provides a hard constraint on the dimensionality of the vector quantizer. For quantizers that are based on a full search over the codebook, the computational complexity of the quantizer increases exponentially with the dimension of the vector quantizer. For example, if the rate is 1 bit per dimension, then a 10-dimensional vector has a codebook of 1024 entries, but a 20-dimensional vector has a codebook with 1024^2 entries.

In classical vector quantization a full search is performed. To allow a practical implementation of such a source coder, the quantizers must have a dimension that makes the problem computationally tractable. It is then natural to ask *when does the performance of vector quantizers of relatively small dimension approach that of vector quantizers of higher dimensions?* The answer to this question suggests what to do to make a source coder both efficient (i.e., perform close to the rate-distortion limit) and implementable. We first note that the space-filling advantage is fixed for optimal quantizers. However, generalizing the result of sections 7.5.1 and 7.5.2 (see problem 3), we find that the memory advantage of the large-dimensional vector quantizer over that of low-dimensional vector quantizers vanishes if the low-dimensional vectors are independent. A decrease in the dependency (as defined by the memory factor) reduces the memory advantage.

The above reasoning suggests that we can make low-dimensional vector quantizers and scalar quantizers efficient by removing redundancy from the signal and, thus, the memory advantage from a vector quantizer. Naturally, this statement is valid only if we can do so without affecting the form of the distortion criterion. This forms a motivation for introducing **models** or **transforms** in chapters 9 and 10 to reduce the interdependency of variables. The general idea is to transform the original source signal into one or more signals that have low redundancy (defined in equation 2.34) prior to quantization. The inverse transform is then performed after the decoding of the indices.

We have not yet discussed how we can limit the shape advantage of high-dimension quantizers. The shape advantage depends on the density functions of the data vectors. For resolution-constrained quantization it vanishes when the density has a boxcar shape. For entropy-constrained vector quantization it always vanishes. Thus, to limit this factor, entropy-constrained quantization in combination with lossless coding must be used. If a constant rate is required, then a buffer must be introduced and the decrease in bit rate results in an increase in delay.

7.6 Practical High-Rate Quantization

In this chapter we have so-far mainly been concerned with analyzing the asymptotic high-rate behavior of quantizers. The theory provides equations for centroid density and knowledge about optimal cell geometry that can be exploited in the construction of quantizers. We now use these results to obtain practical quantizers. While the quantizers are based on theory that applies to high rate only, it is often found that the resulting quantizers perform well at relatively low (practical) rates.

In a practical situation, one generally does not have knowledge of the probability density that is used in high-rate theory. Thus, we must first obtain an analytic description of the density for the data. It is important to note that it is the resulting density model that captures the properties of the source. The density-estimation techniques described in section 4.3 can be used for this purpose, e.g., [37, 38, 39, 28, 40].

Density models consisting of an additive mixture of components are particularly suitable for quantization. For a mixture model of the density, separate quantizers can be used for each mixture component [38, 39, 28]. Each quantizer can then be designed for the density of a single component, which generally has nice analytic properties (Gaussian components are usually selected). In other words, the quantizers are designed for the generic kernels that make up the mixture, independently of the source properties. The overlap of the component densities is generally associated with a loss of optimality, but in practice the quantizers perform well. Particularly when the quantizers of a mixture quantizers do not require the storage of a codebook, as is the case for high-rate theory based quantizers, they facilitate scalability of the rate. Scalability is useful for systems where the channel capacity varies and is known (so-called informed coding). For the encoding with a mixture quantizer, one must classify a data point as belonging to a certain component and corresponding quantizer. The component index is encoded separately. The optimal bit allocation and centroid densities of mixture quantizers are described in more detail in section 7.6.1.

Once we have the bit allocation and the centroid density for the mixture components, the remaining problems for mixture quantization are to design a practical quantizer that exploits knowledge obtained from high-resolution theory and, for the entropy-constrained case, to design an appropriate lossless coder. Thus, we design the quantizers to exploit knowledge about the optimal distribution of the centroids, as expressed by equations 7.48 and 7.59 and minimize the coefficient of quantization of equation 7.40. For constrained-entropy quantizers, uniform centroid densities were obtained and this suggests the usage of quantizers with regularly-spaced centroids. Such quantizers, so-called lattice quantizers, are discussed in section 7.6.2. Good lattices provide a low coefficient of quantization. Lattice quantizers are also used for constrained-resolution quantization. In this context, companding is often employed to transform a uniform density into a density that approaches the asymptotically optimal constrained-resolution density. However, as we see in section 7.6.3, even with companding, constrained-resolution quantizers often can not provide asymptotically optimal performance in the vector case.

7.6.1 Mixture Quantizers

The principle of a mixture quantizer is to model the density function as a weighted addition of mixture components, and to create a quantizer for each component. In this subsection, we discuss the bit allocation and centroid density of the mixture components.

The mixture model for a density is given by equation 4.16, which we repeat here for convenience:

$$f_{X^k}(x^k) = \sum_{i \in \mathcal{I}_{\text{comp}}} p_I(i) f_{i,X^k}(x^k), \tag{7.82}$$

where we simplified the notation, defined $\mathcal{I}_{\text{comp}}$ as the set of mixture component indices, and defined f_{i,X^k} as the density of component i. In practical mixture quantizers, the components are usually selected to be Gaussian densities. The bit allocations and centroid densities for all components are optimized simultaneously. The optimization can be performed for the constrained-entropy and the constrained-resolution cases. It should be noted that the actual encoding methods are generally inconsistent with the optimization method.

Constrained-Resolution Mixture Quantization

In the constrained-resolution mixture quantizer, the objective is to minimize the mean distortion

$$D = \sum_{i \in \mathcal{I}_{\text{comp}}} p_I(i) D_{\text{comp}_i}, \tag{7.83}$$

where D_{comp_i} is the mean distortion within component i, given the sum of the number of centroids for all component quantizers

$$N = \sum_{i \in \mathcal{I}_{\text{comp}}} N_i, \tag{7.84}$$

where N_i is the number of centroids of the quantizer for component i. Normally we select $\log_2(N)$ to be an integer number of bits.

Using the method of Lagrange multipliers, the extended distortion criterion is

$$\eta = \sum_{i \in \mathcal{I}_{\text{comp}}} (p_I(i) D_{\text{comp}_i}(N_i) + \lambda N_i), \tag{7.85}$$

where we have shown explicitly that D_{comp_i} depends on N_i and where λ is the Lagrange multiplier.

To solve the optimization problem, we must know the function $D_{\text{comp}_i}(N_i)$. It is clear that for each component i, the distortion D_{comp_i} must be minimal for the number of centroids N_i. This relation between N_i and D_{comp_i} is given by equation 7.49, which we write in the present context as

$$D_{\text{comp}_i} = C(r, k, \mathcal{G}) N_i^{-\frac{r}{k}} \langle f_{i,X^k}(x^k) \rangle_{\frac{k}{r+k}}. \tag{7.86}$$

Inserting equation 7.86 into equation 7.85, we obtain

$$\eta = \sum_{i \in \mathcal{I}_{\text{comp}}} \left(p_I(i) C(r,k,\mathcal{G}) N_i^{-\frac{r}{k}} \langle f_{i,X^k}(x^k) \rangle_{\frac{k}{r+k}} + \lambda N_i \right). \tag{7.87}$$

Differentiating equation 7.87 towards N_i and setting the result equal to zero renders the solution

$$\begin{aligned} N_i &= \lambda' \left(p_I(i) \langle f_{i,X^k}(x^k) \rangle_{\frac{k}{r+k}} \right)^{\frac{k}{r+k}} \\ &= \lambda' p_I(i)^{\frac{k}{r+k}} \int_{\mathbb{R}^k} f_{i,X^k}(x^k)^{\frac{k}{r+k}} dx^k, \end{aligned} \tag{7.88}$$

where λ' is a constant.

Using equation 7.88 in equation 7.84 allows us to find λ' for a given N,

$$\lambda' = \frac{N}{\sum_{i \in \mathcal{I}_{\text{comp}}} \left(p_I(i) \langle f_{i,X^k}(x^k) \rangle_{\frac{k}{r+k}} \right)^{\frac{k}{r+k}}}. \tag{7.89}$$

Equation 7.88 then provides the values for N_i. Equation 7.48 can be used to specify the centroid density for each mixture component density model,

$$g_{i,C^k}(x^k) = N_i \frac{f_{i,X^k}(x^k)^{\frac{k}{k+r}}}{\int_{\mathbb{R}^k} f_{i,X^k}(x^k)^{\frac{k}{k+r}} dx^k}. \tag{7.90}$$

The quantizer given by equation 7.90 is optimal for that component. This means it is optimal for a mixture quantizer encoder where the component is specified from extraneous information. In a practical application this situation does not occur. We can classify a realization of X^k as belonging to a mixture component using the maximum-likelihood measure and then use that quantizer. However, this approach implies that a subset of centroids is never used. To obtain better performance, we can, for each realization, try all component quantizers and select the one that provides the lowest distortion. That is, we minimize the average distortion measure of equation 7.85 by minimizing it for each realization of X^k. However, also for this implementation the quantizer design is not consistent with the actual quantization method if overlap of the mixture components occurs.

Constrained-Entropy Mixture Quantization

As in constrained-resolution mixture quantization, the goal of constrained-entropy mixture quantization is to minimize the mean mixture distortion of equation 7.83 under a constraint. Let $\mathcal{I}_i$ be the set of indices of the quantizer of mixture component i and let I_i be the random index variable of component i. The index entropy of the mixture quantizer is then

$$\begin{aligned} H &= -\sum_{i \in \mathcal{I}_{\text{comp}}} \sum_{j \in \mathcal{I}_i} p_{I_{\text{comp}}}(i) p_{I_i}(j) \log(p_{I_{\text{comp}}}(i) p_{I_i}(j)) \\ &= -\sum_{i \in \mathcal{I}_{\text{comp}}} p_{I_{\text{comp}}}(i) \log(p_{I_{\text{comp}}}(i)) - \sum_{i \in \mathcal{I}_{\text{comp}}} p_{I_{\text{comp}}}(i) \sum_{j \in \mathcal{I}_i} p_{I_i}(j) \log(p_{I_i}(j)) \\ &= H(I_{\text{comp}}) + \sum_{i \in \mathcal{I}_{\text{comp}}} p_{I_{\text{comp}}}(i) H(I_i), \end{aligned} \tag{7.91}$$

where $H(I_{\text{comp}})$ is the entropy of the component indices and $H(I_i)$ is the entropy of the quantizer index of component i. In the following, $H_i = H(I_i)$ is considered a variable (the rate associated with component i). The entropy of the component indices, $H(I_{\text{comp}})$ is fixed for a given data model and the constraint can be written as

$$\sum_{i \in \mathcal{I}_{\text{comp}}} p_{I_{\text{comp}}}(i) H_i = b, \tag{7.92}$$

where b is a constant.

Using again the method of Lagrange multipliers, the extended criterion becomes:

$$\eta = \sum_{i \in \mathcal{I}_{\text{comp}}} p_{I_{\text{comp}}}(i) (D_{\text{comp}_i}(H_i) + \lambda H_i), \tag{7.93}$$

where the dependency of the mean component distortion D_{comp_i} on H_i is shown explicitly.

According to equation 7.91, the overall entropy H is the weighted sum of the entropies H_i plus an additional fixed term. To obtain a minimum overall distortion for a given set of component entropies $\{H_i\}_{i \in \mathcal{I}_{\text{comp}}}$, the distortion of each component must be minimized for its given entropy. This implies that the rate-distortion relation for each component is given by equation 7.60, which in the present context takes the form

$$D_{\text{comp}_i}(H_i) = C(r, k, \mathcal{G}) \mathrm{e}^{-\frac{r}{k}\left(H_i - h(f_{i,X^k})\right)}, \tag{7.94}$$

where we replaced "approximately equal" with "equal" for convenience and where, to obtain a simple and nonambiguous notation, we use the density component f_{i,X^k} rather than the random variable as argument of $h(\cdot)$. Inserting equation 7.94 into equation 7.93 renders

$$\eta = \sum_{i \in \mathcal{I}_{\text{comp}}} \left(p_{I_{\text{comp}}}(i) C(r, k, \mathcal{G}) \mathrm{e}^{-\frac{r}{k}\left(H_i - h(f_{i,X^k})\right)} + \lambda p_{I_{\text{comp}}}(i) H_i \right). \tag{7.95}$$

Differentiating towards H_i, setting the result equal to zero and solving for H_i renders

$$H_i = h(f_{i,X^k}) + \lambda'', \tag{7.96}$$

where λ'' is a constant. Comparing this result with equation 7.59 we see that the constant centroid density g_{C^k} for all of the component quantizers is the same. Moreover, we can rewrite equation 7.96 as

$$H_i = h(f_{i,X^k}) + \log(g_{C^k}). \tag{7.97}$$

Note that it follows from equations 7.97 and 7.94 that the distortions D_{comp_i} of all the components are equal. If we insert equation 7.97 into equation 7.91, we see that, to obtain a certain overall index entropy H, we must select the centroid density g_{C^k}, shared by all quantizers, as

$$\log(g_{C^k}) \quad = \quad H - H(I_{\text{comp}}) - \sum_{i \in \mathcal{I}_{\text{comp}}} p_{I_{\text{comp}}}(i) h(f_{i,X^k}). \tag{7.98}$$

As in the case of constrained-resolution, the actual implementation of the mixture quantizer is inconsistent with the design. In the constrained-entropy case it is most natural

to classify the realization first to a component, and then quantize with the selected quantizer. However, as in the constrained-resolution case, this implies that a set of centroids that was assigned finite probability is never used. A more efficient procedure is to minimize, for each realization of X^k, the measure

$$\eta = d(x^k, \mathcal{Q}(x^k)) + \lambda l(x^k), \tag{7.99}$$

where $d(x^k, \mathcal{Q}(x^k))$ is the quantization distortion and $l(x^k)$ is the codeword length associated with an original sample x^k. Equation 7.99 is the unaveraged equivalent of equation 7.93. To obtain best performance it is advantageous if the uniform quantizers of the different components are interlaced. However, we note again, that the design method is not entirely consistent with the quantizer if the mixture components overlap.

7.6.2 Lattice Vector Quantizers

A quantizer of low computational complexity can be obtained by placing the centroids at known fixed intervals in $\mathbb{R}^k$. A lattice is a regular arrangement of points in $\mathbb{R}^k$ that is commonly used for this purpose.

Definition of a Lattice

A lattice $\mathcal{L}$ is a set of points defined by means of a set of n independent vectors in $\mathbb{R}^k$ with $n \leq k$. If this set of vectors is denoted $\{g_i^k\}_{i=1,\cdots,n}$, then the lattice is defined by

$$\mathcal{L} = \{c^k : c^k = \sum_{i=1}^{n} u_i g_i^k, \ \forall u_i \in \mathcal{Z}\}. \tag{7.100}$$

To allow a good description of $\mathbb{R}^k$, the number of vectors g_i^k in the **generating set** is typically k (in other words, $n = k$) and the vectors g_i^k are independent (i.e., nondegenerate). In the following, we assume that $n = k$.

Often, the generating set is written in the form of a matrix, called the **generator matrix**:

$$G = \left[g_1^k \cdots g_n^k\right]^T. \tag{7.101}$$

Considering the typical case where $n = k$, multiplication of the transpose generator matrix by any vector $u^k \in \mathcal{Z}^n$ results in a lattice point (codebook entry) $c_{u^k}^k \in C^k$:

$$c_{u^k}^k = G^T u^k, \quad u^k \in \mathcal{Z}^k. \tag{7.102}$$

To be useful in a practical application, the lattice must be **truncated**. That is, only a finite number of contiguous centroids are used. This is natural for the constrained-resolution case, which has only a finite number of centroids. It is less natural for the constrained-entropy case, for which we showed in section 7.4.3 that the centroid density should be uniform. In practice, a finite lattice is needed also for the constrained-entropy case to facilitate existing codeword assignment methods. When truncation is used, the region inside the truncated lattice is called the **granular region** and the region outside the truncated lattice is called the **overload region**[3].

[3]We note that the discussed low-computational-complexity methods for lattice quantization apply to the granular region; there is no well-accepted method for fast quantization in the overload region.

Example 7.6: Scalar quantizers as a lattice
As an example of a lattice quantizer, consider a sequence of k variables (dimensions), each with a scalar quantizers of unity step size. Such a set of k scalar quantizers is equivalent to a lattice quantizer with the unit vectors as generating set. The generating matrix is thus simply the identity matrix: $G = I$.

From input vector x^k to lattice index u^k

As said, the major motivation for lattice quantization is that one can find the nearest lattice point to an input vector at low computional cost [41, 42]. A generic method [42] to map an input vector $x^k \in R^k$ to a vector $u^k \in Z^k$ labeling a lattice point starts with performing the inverse transformation G^{-T} and rounding to obtain an estimate of u^k:

$$\hat{u}^k = \lfloor G^{-T}(x^k + a^k) \rfloor. \tag{7.103}$$

where a^k is an offset selected to minimize the squared error remaining after rounding (cf. problem 9). It is easy to see that for the scalar quantizer (a trivial one-dimensional lattice), $a^1 = 0.5$. However, in general, the resulting $\hat{u}^k$ does not correspond in the optimal (nearest) lattice point. The nearest point is found by searching over all lattice points within a given radius r of $x^k - G^T\hat{u}^k$ of the origin:

$$u^k_{\text{correction}} = \underset{\{u^k : \|G^T u^k\| < r \wedge u^k \in \mathcal{Z}^k\}}{\operatorname{argmin}} \|(x^k - G^T\hat{u}^k) - G^T u^k\|^2. \tag{7.104}$$

The set $\{u^k : \|G^T u^k\| < r \wedge u^k \in \mathcal{Z}^k\}$ has a small cardinality, thus facilitating a fast search procedure. The final solution is then

$$u^k_{\text{opt}} = \hat{u}^k + u^k_{\text{correction}}. \tag{7.105}$$

For many lattices, faster search procedures exist [41, 43].

From lattice index u^k to codeword; lattice truncation

The mapping from u^k to a codeword is usually nontrivial and depends on the particularities of the quantization method. We first consider the mapping from u^k to the index for a **constrained-resolution lattice quantizer** with two types of truncations. Note, however, that the uniform centroid density of the lattice is not optimal for constrained-resolution quantization, where the centroid density should follow equation 7.48. However, the optimal centroid density becomes flatter with increasing dimension d and increasing distortion power r, making lattices more attractive. We return to this issue in section 7.6.3.

A very simple truncation is that obtained when the truncated lattice is defined by the u^k contained by a "cubic" truncation of u^k: $\{u^k : u^k_i \in \mathcal{U}, \forall_{i=1,\cdots,k}\}$, where $\mathcal{U} = \{0, 1, \cdots, M-1\}$. If we select M to be a power of two, we can define a codeword of $\log_2(M)$ bits for dimension i for any lattice point in $\mathbb{R}^k$ as $\mathsf{c}_i = \mod(u^k_i, M)$, where $\mod(a, b)$ is the a modulo b. The method provides a good encoding for data that lie within the truncated lattice. A two-dimensional example of such a truncated lattice is shown on the left side of figure 7.2. The method maps points outside the parallelogram to points inside the parallelogram. The modulo operation implies that the set of all points that map to the same point inside the parallelogram form an offset sublattice[4] with a

[4] A sublattice of a lattice $\mathcal{C}$ is a lattice that contains a subset of the points of the lattice $\mathcal{C}$

generator matrix $M\,G$. The fixed-length codeword specifying a point of the truncated k-dimensional lattice is then a simple concatenation of the k codewords obtained for the k dimensions. The method is easily generalized to include the case where only M^k is a power of two (and not M). Unfortunately, the present cubic truncation is generally not very attractive, since it does not approximate the shape of most practical data densities.

Exploiting that good cell shapes are close to spherical, an attractive truncation for spherically symmetric densities is obtained by using a lattice truncation shape that is identical to that of the cell shape. We scale up the cell shape by a factor M to obtain the truncation shape. We can construct the codeword as for the case of cubic trunction. That is, we select M to be a power of two and $\mathsf{c}_i = \mod(u_i^k, M)$ forms a $\log_2(M)$ bit codeword for dimension i. The concatenation of the k indices provides an index for the corresponding lattice point in R^k. Each codeword can have been generated by input vectors that have as nearest lattice point a lattice point that is on a sublattice with generator matrix $M\,G$ with a particular offset (remember that M is an integer, G a matrix). Interestingly, no explicit knowledge of the truncation shape is needed at the encoder. At the decoder, we must select the point that is the intersection of the truncated lattice and the offset sublattice.

We now construct the intersection of a truncated lattice and the offset sublattice needed at the decoder. We first construct a decoded lattice index $\tilde{u}^k$ from the codewords (generally $\tilde{u}^k = [\mathsf{c}_1, \cdots, \mathsf{c}_k]^T$) and then compute the corresponding lattice point

$$\tilde{c}^k = G^T \tilde{u}^k \tag{7.106}$$

that may be outside our truncated lattice. Next we determine the sublattice point b^k:

$$b^k = \underset{v^k \in \mathcal{Z}^k}{\operatorname{argmin}} \|\tilde{c}^k + a^k - MG^T v^k\|, \tag{7.107}$$

where the offset a^k is now used to ensure that no lattice points fall on the truncation boundary and where we note that in practice only a small, finite subset of $\mathcal{Z}^k$ needs to be searched. The desired lattice point is then $c^k = \tilde{c}^k - MG^T b^k$.

Example 7.7: Hexagonal lattice truncation
The hexagonal lattice has the generator matrix

$$G = \begin{bmatrix} 1 & 0 \\ \frac{1}{2} & \frac{\sqrt{3}}{2} \end{bmatrix}.$$

Simple truncation of this lattice, such that each dimension of u^2 has elements in the set $\{0, 1, \cdots, 11\}$ leads to the lattice displayed on the left in figure 7.2. The hexagonal truncation, with the same number of points, is shown on the right in figure 7.2. The latter truncation shape is more natural for spherically symmetric densities as commonly used in components of mixture density models.

To obtain good performance, the lattice must be scaled appropriately. The scaling is often based on heuristics. For the constrained-resolution case, the number of points of the lattice is fixed if one constrains the rate. The methods described at the end of section 7.6.3 can be used to obtain an optimal scaling.

So-far, we have discussed the mapping from lattice index to codeword for a lattice quantizer for the constrained-resolution case. However, the uniform density of a lattice

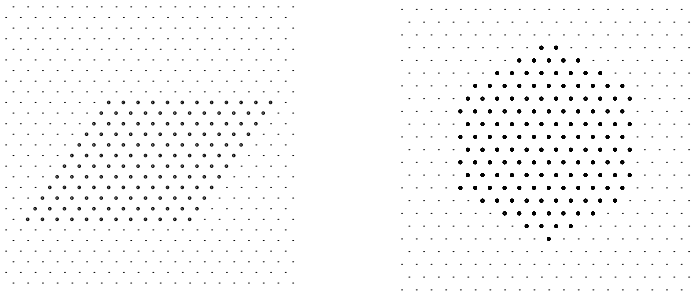

Figure 7.2: Truncated hexagonal lattices with 144 centroids embedded in an untruncated lattice. On the left, straight truncation of u^k. On the right, the truncation into a hexagonal shape that is more useful for spherically symmetric data densities.

quantizer is more suited for **entropy-constrained lattice quantization**. To avoid storage of codewords, the usage of arithmetic coding in combination with a parametric description of the cumulative distribution function is desirable. The codewords can then be computed for each realization of the random variable. However, in practice it is difficult to obtain a parametric description of the cumulative distribution function. A particular method that uses storage is described in [44]. It uses two levels of truncation of the lattice. The first truncation is defined by a subset $\mathcal{A} \subset \mathcal{Z}^k$ that contains the vectors u^k that have a probability above a certain threshold. A lossless code is designed for the subset $\mathcal{A}$ and stored in a table. A second truncation is defined by the subset $\mathcal{B} \subset \mathcal{Z}^k$, with $\mathcal{B} \cap \mathcal{A}$ an empty set, that includes all other vectors u^k that have some practical chance of occurring. All vectors in $\mathcal{B}$ are assigned an identical-length codeword of sufficient length based on the indices u_i^k. The codeword length for the vectors in $\mathcal{B}$ has no significant effect on the mean rate because of their low probability. Other vectors u^k, not part of $\mathcal{A}$ or $\mathcal{B}$ are assigned to a default index, thus ensuring low computational complexity at the expense of a rarely occurring large distortion.

Lattice Quantization Noise

We now consider the quantization noise of a lattice quantizer that is optimal for the squared error criterion ($r = 2$). In particular, we consider properties of the covariance matrix of lattice quantization noise. Most importantly, we show that the noise covariance matrix is a scaled identity matrix for an optimal lattice.

To start we note that the noise covariance matrix determines the quantization coefficient, which equation 7.39 shows to be a normalized measure of distortion. For the squared error criterion, and considering that all cells of a lattice are identical, equation 7.40 can

be written as

$$\begin{aligned} C(k,\mathcal{G}(i)) &= \frac{1}{k}\frac{1}{V_i^{\frac{k+2}{k}}}\int_{\mathtt{V}_i}(x^k-c_i^k)^H(x^k-c_i^k)dx^k \\ &= \frac{1}{k}\frac{1}{V^{\frac{k+2}{k}}}\int_{\mathtt{V}} v^{kH}v^k dv^k \\ &= \frac{1}{k}\frac{1}{V^{\frac{k+2}{k}}}\int_{\mathtt{V}} \mathrm{tr}(v^k v^{kH})dv^k \\ &= \frac{1}{k}\mathtt{V}^{-\frac{2}{k}}\mathrm{tr}\left(\frac{1}{V}\int_{\mathtt{V}} v^k v^{kH}dv^k\right) \\ &= \frac{1}{k}\mathtt{V}^{-\frac{2}{k}}\mathrm{tr}(R_v), \end{aligned} \tag{7.108}$$

where $v^k = x^k - c^k$ is the quantization noise vector and R_v is the covariance matrix of v^k (for minimal distortion, $c_i \in \mathtt{V}_i$ is selected so that $\mathrm{E}[v^k] = 0$),

$$R_v = \frac{1}{V}\int_{\mathtt{V}} v^k v^{kH}dv^k = \mathrm{E}[v^k v^{kH}], \tag{7.109}$$

and where we omitted the subscripts i since all cells have identical geometry.

It is straighforward to prove that, for the ideal spherical cell, R_v is a scaled identity matrix: $R_v = kD_iI$, where D_i is the distortion per dimension of one cell (cf. equation 7.39) of the lattice. In the case of a lattice, we can change the noise covariance matrix, by multiplying all points defining the lattice (i.e., the centroids and the partition) with a **shaping matrix** A. If we constrain A to have a unity determinant,

$$\det(A) = 1, \tag{7.110}$$

then the mean density of the centroids does not change (note that the unit cube is mapped by A onto a region with volume $\det(A)$). The noise covariance matrix is affected by the transformation as

$$R_{v,\mathrm{shaped}} = AR_vA^H, \tag{7.111}$$

where R_v is the covariance matrix for the unshaped lattice. We can then write for the coefficient of quantization

$$\begin{aligned} C(k,\mathcal{G}(i,A)) &= \mathtt{V}^{-\frac{2}{k}}\frac{1}{k}\mathrm{tr}(AR_vA^H) \\ &\geq \mathtt{V}^{-\frac{2}{k}}\det(AR_vA^H)^{\frac{1}{k}} \\ &= \mathtt{V}^{-\frac{2}{k}}\det(R_v)^{\frac{1}{k}}, \end{aligned} \tag{7.112}$$

where we used an equality proven in Appendix D. Thus, a lower bound for the coefficient of quantization is $\mathtt{V}^{-\frac{2}{k}}\det(R_v)^{\frac{1}{k}}$. This lower bound can be attained by selecting A such that $\mathrm{tr}(AR_vA^H) = \det(R_v)^{\frac{1}{k}}$. We achieve the lower if the shaping matrix maps the covariance matrix R_v into a scaled identity matrix:

$$A = \frac{R_v^{-\frac{1}{2}}}{\det(R_v^{-\frac{1}{2}})}, \tag{7.113}$$

where $R_v^{-\frac{1}{2}}$ is a root of R_v^{-1}. The resulting scaled identity matrix form of the covariance matrix, is commonly referred to as a **white** covariance matrix. Conversely, the above result implies that the optimal lattice for any dimension k has a scaled identity matrix as noise covariance matrix,

$$R_v(\text{optimal lattice}) = D_i(\text{optimal lattice})I, \tag{7.114}$$

where D_i(optimal lattice) is the distortion per cell and per dimension of the optimal lattice. Moreover, any lattice that has been subjected to an optimal shaping results in white quantization noise. The fundamental result that R_v(optimal lattice) is white was first obtained by Zamir and Feder [45].

7.6.3 Companding

Equation 7.48 shows that the optimal centroid density for constrained-resolution coding is nonuniform. This implies that straight application of a lattice quantizer to the constrained-resolution case is not optimal. A seemingly obvious modification of the quantizer structure is a method called **companding**. Companding introduces a one-to-one (i.e., invertible) mapping $\mathtt{h} : \mathcal{A}_{X^k} \rightarrow \mathcal{A}_{Y^k}$, that maps the random variable X^k into the random variable Y^k, such that the support $\mathcal{A}_{X^k} \subset \mathbb{R}^k$ of the density of a random variable X^k maps onto a **finite** region of support $\mathcal{A}_{Y^k}$ of a **uniform** quantizer. The region of support of $\mathcal{A}_{Y^k}$ is usually selected to be a hyperrectangle or a hyperellipsoid. Quantization is then performed in $\mathcal{A}_{Y^k}$ and the decoder maps the centroids back to the original domain using the inverse mapping $\mathtt{h}^{-1}$. Companding was originally introduced by Bennett [26] for the scalar case. For historical reasons the mapping $\mathtt{h}$ is often referred to as the **compressor** and the inverse mapping $\mathtt{h}^{-1}$ as the **expander**.

An important side benefit of companding is that the resulting quantizer has no overload region. Thus, the common assumption in high-rate theory that only the cells in the granular region need to be considered is more generally valid, resulting in a more accurate analysis for practical situations. The Gersho conjecture that all cells have the same shape is violated at the edges of $\mathcal{A}_{Y^k}$, but this effect is generally minor compared to the neglect of an overload region.

Upon more careful evaluation, we see that companding is not without its problems for the multi-dimensional case, e.g., [33, 46]. Consider the effect of the mapping $\mathtt{h}^{-1}$ on the partition associated with the lattice. In general, the cell shapes are scaled differently in different dimensions by the inverse mapping. Thus, if the lattice in $\mathcal{A}_{Y^k}$ was selected to have an optimal cell shape (optimal $\mathcal{G}$), this optimality of the shape are lost in the inverse mapping. A white quantization-noise covariance matrix is generally changed into a nonwhite covariance matrix. More-over the manner in which the cell shape is changed varies with the value that X^k takes. The effect of the compander on the cell shapes in the partition for X^k can affect quantization performance significantly. However, it should be noted that despite the discussed nonoptimality, companding-based lattice quantization structures can result in high quality at low computational effort.

The nonoptimality problems associated with multi-dimensional companding can be avoided by exploiting the properties of random codebooks [40]. As has been mentioned before, random codebooks are asymptotically optimal with increasing dimension. Furthermore, the random codebook remains a random codebook upon a mapping $\mathtt{h}^{-1}$. By

using a superposition of conventional lattices, each with a random displacement, this advantage of a random codebook is obtained, while retaining the efficient search procedures associated with the lattices. A disadvantage of this method is that to obtain near-optimality, generally a large number of codebooks is needed.

Below we consider in detail the design of companders. The methods can be exploited for conventional companding-based lattice quantization, or in combination with a random lattice superposition.

Optimal Companding for a Lattice (Squared Error)

Let us consider the effect of a compander $\mathtt{h}$ on the performance of a lattice quantizer. We consider the squared-error case only. We first consider the distortion of the lattice quantizer on the random variable Y^k and then show how this expression must be modified to obtain the distortion for X^k. The overall distortion for Y^k can be written as a sum over the cells:

$$\begin{aligned} D(Y^k, \mathcal{Q}(Y^k)) &= \sum_{i\in\mathcal{I}} \int_{\mathcal{V}_{i,Y^k}} (y^k - c_i^k)^H (y^k - c_i^k) f_{Y^k}(y^k) dy^k \\ &\approx \sum_{i\in\mathcal{I}} f_{Y^k}(c_i^k) \int_{\mathcal{V}_{i,Y^k}} (y^k - c_i^k)^H (y^k - c_i^k) dy^k \\ &\approx \int_{\mathcal{A}_{Y^k}} f_{Y^k}(c^k) \frac{1}{V_{Y^k}} \int_{\mathcal{V}_{Y^k}} v^{kH} v^k dv^k dc^k \qquad (7.115) \\ &= \frac{1}{V_{Y^k}} \int_{\mathcal{V}_{Y^k}} v^{kH} v^k dv^k, \qquad (7.116) \end{aligned}$$

where v^k is the quantization noise, and where we exploited that according to the Gersho conjecture all cells $\mathcal{V}_{i,Y^k}$ are of identical shape and can be denoted as $\mathcal{V}_{Y^k}$ and have volume V_{Y^k}.

Next, we evaluate the overall distortion obtained after the mapping $\mathtt{h}^{-1}$ from $\mathcal{A}_Y$ to $\mathcal{A}_X$. We must describe the mapping in more detail. The mapping $\mathtt{h}^{-1}$ maps a vector y_1^k into $x_1^k = \mathtt{h}^{-1}(y_1^k)$. We consider a mapping $\mathtt{h}$ such that both $\mathtt{h}$ and $\mathtt{h}^{-1}$ are continuous, differentiable functions. Then, for an ϵ with sufficiently small Euclidean norm $\|\epsilon\|$, we have

$$\begin{aligned} x_2^k = \mathtt{h}^{-1}(y_2^k) &= \mathtt{h}^{-1}(y_1^k + \epsilon^k) \\ &\approx \mathtt{h}^{-1}(y_1^k) + A(x^k)\epsilon^k, \qquad (7.117) \end{aligned}$$

where $A(x^k)$ is the Jacobian matrix of the function $\mathtt{h}^{-1}(\cdot)$

$$A_{i,j}(x^k) = \left.\frac{\partial(\mathtt{h}^{-1}(y^k))_i}{\partial(y^k)_j}\right|_{y^k=\mathtt{h}(x^k)}, \qquad (7.118)$$

where $(\mathtt{h}(x^k))_i$ is the i'th component of the vector $\mathtt{h}(x^k)$ and $(x^k)_j$ is the j'th component of the vector x^k. Using this notation, the mapping $\mathtt{h}^{-1}$ changes the distortion per cell at location x^k in $\mathcal{A}_X$ as follows:

$$\frac{1}{V_{Y^k}} \int_{\mathcal{V}_{Y^k}} v^{kH} v^k dv^k \quad \xrightarrow{\mathtt{h}^{-1}} \quad \frac{1}{V_{Y^k}} \int_{\mathcal{V}_{Y^k}} v^{kH} A^H(x^k) A(x^k) v^k dv^k. \qquad (7.119)$$

Using the structure of equation 7.115 (but replacing, for simplicity, $\approx$ with $=$), we can then write the mean squared error for X in the support region $\mathcal{A}_{X^k}$ of f_{X^k} as

$$\begin{aligned} D(X^k, \mathcal{Q}(X^k)) &= \int_{\mathcal{A}_{X^k}} f_{X^k}(x^k) \frac{1}{V_{Y^k}} \int_{\mathcal{V}_{Y^k}} v^{kH} A^H(x^k) A(x^k) v^k dv^k dx^k \\ &= \int_{\mathcal{A}_{X^k}} f_{X^k}(x^k) \frac{1}{V_{Y^k}} \int_{\mathcal{V}_{Y^k}} \operatorname{tr}\left(v^k v^{kH} A(x^k) A^H(x^k)\right) dv^k dx^k \\ &= \int_{\mathcal{A}_{X^k}} f_{X^k}(x^k) \operatorname{tr}\left(\frac{1}{V_{Y^k}} \int_{\mathcal{V}_{Y^k}} v^k v^{kH} dv^k A(x^k) A^H(x^k)\right) dx^k \\ &= \int_{\mathcal{A}_{X^k}} f_{X^k}(x^k) \operatorname{tr}\left(D(Y^k, \mathcal{Q}(Y^k)) A(x^k) A^H(x^k)\right) dx^k \\ &= D(Y^k, \mathcal{Q}(Y^k)) \int_{\mathcal{A}_{X^k}} f_{X^k}(x^k) \operatorname{tr}\left(A(x^k) A^H(x^k)\right) dx^k, \end{aligned} \quad (7.120)$$

where we assumed that the quantization noise vector v^k has a white covariance matrix, i.e., that $R_v = D(Y^k, \mathcal{Q}(Y^k))I$ (which we showed to be true for optimal lattices in section 7.6.2) in $\mathcal{A}_Y$ and where we used the trace properties

$$\begin{aligned} v^{kH} A^H(x^k) A(x^k) v^k &= \operatorname{tr}(A(x^k) v^k v^{kH} A^H(x^k)) \\ &= \operatorname{tr}(v^k v^{kH} A(x^k) A^H(x^k)). \end{aligned} \quad (7.121)$$

Equation 7.120, which separates the effect of companding and quantization in separate distinct factors, is a result originally obtained by Bucklew [47].

Equation 7.120 forms a criterion that must be minimized to find the optimal compander function $\mathtt{h}$. In general, this optimization is a difficult task. We restrict our discussion to **cartesian companding** and restrict it even more by considering only vectors X^k with independent components,

$$f_{X^k} = \prod_{i=1}^{k} f_{X_i^k}(x_i^k), \quad (7.122)$$

where x_i^k is component i of the vector x^k. Such companders have been considered in [48, 39]. It is relevant to remember that the Karhunen-Loève transform, which is discussed in more detail in section 9.4.1, can transform any multi-variate Gaussian distributed vector into a vector with independent components. In cartesian companding the function $\mathtt{h}$ is constrained to be of the form

$$\mathtt{h}(x^k) = [\mathtt{h}_1(x_1^k), \cdots, \mathtt{h}_k(x_k^k),], \quad (7.123)$$

where the $\mathtt{h}_i(x_i^k)$ are scalar functions. It immediately follows that for the cartesian compander the matrix $A(x^k)$ is diagonal, with

$$A_{ii}(x_i^k) = \frac{1}{\dot{\mathtt{h}}_i(x_i^k)}, \quad (7.124)$$

where $\dot{\mathtt{h}}_i$ is the derivative of $\mathtt{h}_i$.

We constrain the function $\mathtt{h}$ to map $\mathbb{R}^k$ onto a hyperrectangle $\mathcal{A}_Y$ that has range $[0, a_i]$ for each dimension i. (Naturally, if all a_i are equal, then the mapping is onto a hypercube.) This implies k constraints of the form

$$\int_{\mathbb{R}^k} \dot{\mathtt{h}}(x_i^k) dx_i^k = \int_{\mathbb{R}^k} \frac{1}{A_{ii}(x_i^k)} dx_i^k = a_i, \ \forall_{i \in \{1, \cdots, k\}}. \quad (7.125)$$

For our purposes, it is convenient to write the constraints 7.125 as

$$\int_{\mathbb{R}^k} \frac{\prod_{j=1}^k f_{X_j^k}(x_j^k)}{f_{X_i^k}(x_i^k)} \frac{1}{A_{ii}(x_i^k)} dx^k = a_i, \ \forall_{i\in\{1,\cdots,k\}}, \tag{7.126}$$

where we exploited the property that (marginal) densities integrate to unity.

Noting that

$$\mathrm{tr}(A(x^k)A^H(x^k)) \quad = \quad \sum_{i=1}^k A_{ii}^2(x_i^k). \tag{7.127}$$

we see that finding the mapping $\mathtt{h}$ that minimizes the criterion 7.120 under the constraints 7.126 is equivalent to finding the functions $A_{ii}(\cdot)$ that minimize

$$\eta = \int_{\mathbb{R}^k} \sum_{i=1}^k \left(\prod_{j=1}^k f_{X_j^k}(x_j^k) A_{ii}^2(x_i^k) + \lambda_i \frac{\prod_{j=1}^k f_{X_j^k}(x_j^k)}{f_{X_i^k}(x_i^k)} \frac{1}{A_{ii}(x_i^k)} \right) dx^k, \tag{7.128}$$

where the λ_i are Lagrange multipliers. The Euler-Lagrange equations are

$$2 \prod_{j=1}^k f_{X_j^k}(x_j^k) A_{ii}(x_i^k) - \lambda_i \frac{\prod_{j=1}^k f_{X_j^k}(x_j^k)}{f_{X_i^k}(x_i^k)} A_{ii}^{-2}(x_i^k) = 0, \ \forall_{i\in\{1,\cdots,k\}}, \tag{7.129}$$

which leads to the solution

$$A_{ii}(x_i^k) = \frac{\lambda_i}{2} (f_{X_i^k}(x_i^k))^{-\frac{1}{3}}. \tag{7.130}$$

Combining this with the constraint 7.126, we obtain:

$$A_{ii}(x_i^k) = \frac{1}{a_i b_i} (f_{X_i^k}(x_i^k))^{-\frac{1}{3}}, \tag{7.131}$$

where we used the notation

$$b_i = \frac{1}{\int_{\mathbb{R}} (f_{X_i^k}(x_i^k))^{\frac{1}{3}} dx_i^k}. \tag{7.132}$$

We thus obtain the cartesian companding functions

$$h_i(x_i^k) = \int_{-\infty}^{x_i^k} \frac{1}{A_{ii}(w_i^k)} dw_i^k = a_i b_i \int_{-\infty}^{x_i^k} (f_{X_i^k}(w_i^k))^{\frac{1}{3}} dw_i^k \tag{7.133}$$

For a companding-based quantizer, the number of centroids is proportional to the volume of the hyperrectangle $\mathcal{A}_Y$. We now determine the set $\{a_i\}_{i\in\{1,\cdots,k\}}$ that minimizes the distortion of equation 7.120 under the constraint that the number of centroids, i.e., the volume of $\mathcal{A}_Y$ is constant,

$$\prod_{i=1}^k a_i = 1. \tag{7.134}$$

Table 7.1: Encoding-decoding operation for constrained-resolution lattice quantizer with cartesian companding.

1. decorrelating transform (the Karhunen-Loève transform of section of 9.4.1) to the input vector to obtain approximately independent components
2. cartesian compander
3. uniform lattice quantizer encoder
4. uniform lattice quantizer decoder
5. inverse cartesian compander
6. inverse of decorrelating transform

Using equation 7.133, 7.127, and 7.120 we find that this is equivalent to minimizing

$$\begin{aligned}\eta &= \int_{\mathbb{R}^k} \sum_{i=1}^{k} \frac{\prod_{j=1}^{k} f_{X_j^k}(x_j^k)}{a_i^2 b_i^2 (f_{X_i^k}(x_i^k))^{\frac{2}{3}}} dx^k + \lambda \prod_{i=1}^{k} a_i \\ &= \sum_{i=1}^{k} \int_{\mathbb{R}^k} \frac{(f_{X_i^k}(x_i^k))^{\frac{1}{3}}}{a_i^2 b_i^2} dx_i^k + \lambda \prod_{i=1}^{k} a_i \\ &= \sum_{i=1}^{k} \frac{1}{b_i^3 a_i^2} + \lambda \prod_{i=1}^{k} a_i. \end{aligned} \tag{7.135}$$

We find the optimal set $\{a_i\}$ by differentiating towards the individual a_i and setting the result to zero. We then find that for the optimal cartesian compander is of the form of equation 7.133 with

$$a_i = \lambda' b_i^{-\frac{3}{2}} = \lambda' (\int_{\mathbb{R}} (f_{X_i^k}(x_i^k))^{\frac{1}{3}} dx_i^k)^{\frac{3}{2}}, \ \forall_{i \in \{1,\cdots,k\}}, \tag{7.136}$$

where λ' is a constant that selects the rate. This result is qualitatively supported by intuition: b_i is large when $f_{X_i^k}(x_i^k)$ is concentrated over a small region of x_i^k. It then follows that dimensions with a large b_i should require only a small support in $\mathcal{A}_Y$.

Table 7.1 summarizes the steps for a typical constrained-resolution lattice quantizer with cartesian companding. In the mixture quantizer case, each of these steps is applied for each component of the mixture quantizer. It should be noted that the decorrelating transform is only properly motivated for the Gaussian data case. It furthermore should be remembered that a constrained-resolution lattice quantizer, even when companding is used cannot attain optimal performance. In practice, however, good performance can be obtained at very low computational complexity, e.g., [39].

7.7 Problems

1. Consider a variable with a density $f_X(x)$, a distortion criterion $d(x, y) = |x - y|$ and a scalar quantizer with step size Δ_i. Assume high-rate. Denote the centroid density as $g_C(x)$.

(a) Derive a relation between the average distortion for a cell and Δ_i.

(b) Derive an expression for the distortion as a function of the centroid density function $g_C(x)$.

(c) Under the constraint that the total number of centroids is N, obtain the optimal centroid density.

2. We evaluate quantitatively the advantages of vector quantization for the constrained-resolution case when the high-rate approximation is valid.

(a) Evaluate and plot the shape advantage for the boxcar, Laplace, and Gaussian densities as a function of the dimension k.

(b) Evaluate and plot the memory advantage for a Gauss-Markov source as a function of the dimension k for the following correlation coefficients: 0.0, 0.5, and 0.9.

3. In this problem, we extend the "vector-quantization advantage" results assuming high-rate and Gaussianity. We consider the constrained-resolution case. Work out the simplest form possible for equations expressing the advantage of a vector quantizer of dimension k over that of dimension m, with $k > m$ in terms of a space-filling, shape, and memory advantage.

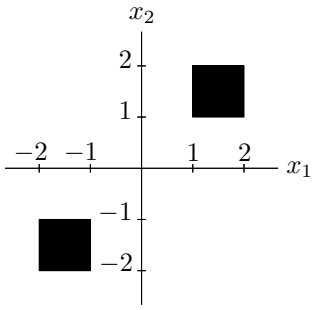

Figure 7.3: The density function of problem 4.

4. Consider the random vector $X = [X_1, X_2]$ of figure 7.3. We want to evaluate the advantage of vector quantization (two-dimensional) over scalar quantization for the case of the squared-error criterion.

(a) Consider constrained-resolution scalar quantization. Draw a *two-dimensional* figure with the optimal quantizer grid for the case that each scalar quantizer gets 3 bits. Clearly show where the centroids are.

(b) Consider constrained-resolution vector quantization. You again have 6 bits total. Draw a figure that shows the location of the centroids for a system with optimal or near-optimal performance.

(c) Draw the optimal lattice for an optimal two-dimensional vector quantizer assuming uniform density.

(d) For a given distortion and at high rate, estimate how many bits less per dimension you need for the two-dimensional vector quantizer as compared to the scalar quantizer, to obtain the same performance with the density of figure 7.3.

5. Consider a cell of dimension k and unity radius. The phenomenon of sphere hardening implies that the fraction of the volume located near the surface increases with k. Compute the fraction of the volume of a sphere located within 0.01 of the surface for $k = 1$, 10, 100 and 1000.

6. Consider the Laplace density $f_X(x) = \frac{a}{2}e^{-a|x|}$ and the absolute diffence distortion measure.

   (a) Find and plot the relation between rate and distortion for high-rate constrained-resolution scalar quantization.

   (b) Find and plot the relation between rate and distortion for high-rate constrained-entropy scalar quantization.

   (c) Compare your results to the rate-distortion function.

7. Prove that $C(r, k, \mathcal{G}) \underset{k\to\infty}{\to} (2\pi\mathrm{e})^{-1}$ for a random codebook.

8. Consider a Gaussian mixture probability-density model with two components, both of unit variance. The first component has probability 0.75 and a mean of 0. The second component has a probability of 0.25 and a mean of 2. We use the squared-error distortion measure.

   (a) Plot the overall probability density.

   (b) Evaluate the differential entropy of this probability density by stochastic integration.

   (c) Plot the Shannon lower bound for the rate-distortion function for the probability density. Is the bound tight?

   (d) Write a program for a constrained-entropy mixture-model based quantizer and run it on artificially generated data for mean squared distortions 0.1 and 0.01. Estimate the resulting practical rate-distortion relation and plot the two points you obtained in the plot of the Shannon lower bound. Comment on the plausibility of your results.

9. Consider the offset a^k of equation 7.103. For a three-dimensional lattice that has as generator matrix the identity matrix I, find the optimal a^k.

10. You must quantize a scalar variable with the absolute error criterion under the constrained-entropy constraint. The high-rate assumptions can be assumed to be valid. You decide to use density model that is a mixture of Laplacian densities, mainly because that allows you to obtain a good estimate of the rate of the quantizer even if you don't know the distribution of the variable.

    (a) Compute the coefficient of quantization, $C(r, k, \mathcal{G}(i))$ for the absolute error criterion for one dimension.

    (b) The rate depends of both the actual distribution and the modelled distribution according to

    $$R = -\sum_{i\in\mathcal{I}} f_Y(c_i)\Delta \log(f_X(c_i)/g_C).$$

    State and explain what f_Y and f_X are (that is, explain which is the actual and which one is the modelled density and why).

(c) Let us assume that you have a correct estimate of $\mathrm{E}[|x|]$ and, thus, of the parameter a for a Laplacian model. Then, starting from the above expression, obtain a relation between R and the distortion D that shows that the rate R does not depend on the actual density (except through $\mathrm{E}[|x|]$).

(d) Consider a Laplacian mixture quantizer. Write down a relevant equation relating the total rate and total distortion. Exploit that the centroid density is the same for all components. (Note that this is independent of the details of the overall density.)

11. Dithering is the addition of random noise, V, to the input variable, X. That is, the variable $X + V$ is quantized. In the following, consider a uniform quantizer with step size Δ and a noise V that is uniformly distributed over $[-\Delta/2, \Delta/2]$. Where needed, make the high-rate assumption that the density $f_X(x)$ varies slowly compared to the quantization step size Δ.

(a) Compute the mean-squared error of the uniform quantizer without dithering.

(b) Compute the mean-squared error (relative to X) of the dithered quantizer output $\mathcal{Q}(X + V)$.

(c) If the decoder knows the noise variable V (possible if pseudo-noise sequence is used), we can subtract the noise at the output to obtain $\hat{X} = \mathcal{Q}(X+V)-V$. Again compute the mean-squared error.

(d) Show that for an ideal quantizer (no dithering, vector quantization) that reaches the Shannon lower bound the quantization error $\hat{X}-X$ is independent of the decoded signal $\hat{X}$. Hint: this problem has nothing to do with dithering; simply find the Shannon lower bound starting from $I(X; \hat{X})$.

(e) The result of 11c suggests that there are cases where dithering does not affect performance (note that the result would also hold for a vector quantizer!). It can be shown that the quantization noise, is not independent of the output in this case. However, the result of 11d indicates then that, in general, dithering cannot reach the Shannon lower bound. Provide an explanation of these seemingly contradictory results.

12. In this problem we consider high-rate entropy-constrained scalar multiple-description coding with two channels (HRECSMDTC). In two-channel multiple-description coding we have two quantizers operating on the same variable. We transmit the index of each variable. Each of the indices is sometimes lost (for example, because a router in the network is overloaded and clears its queue.) If we receive one index we simply use the corresponding centroid as reconstructed value and get a *side* distortion D_s. If we receive both indices we try to combine both into a better decoded value and obtain a *central* distortion D_c. We consider the case where each channel receives an identical rate R and where the variable is a Gaussian with variance 4 and mean 2 and the distortion criterion is the squared-error criterion.

(a) Consider the case where both high-rate scalar quantizers are identical (this is simply transmitting the information twice). Find $D_s(R)$ and $D_c(R)$ (remember that R is the rate of one channel).

(b) Consider the case where the quantization levels of the two high-rate scalar quantizers have an offset $\alpha\Delta$, where Δ is the step size. Find $D_s(R)$ and $D_c(R)$ as a function of α. Find the optimal offset α.

(c) At a given distortion and for the case of no loss of indices, express the advantage of optimal HRECSMDTC over the simple method of 12a in bits.

(d) At a given rate and for the case of no loss of indices, express the advantage of optimal HRECSMDTC over the simple method of 12a in dB.

13. Let $f_G(x)$ be a Gaussian density with unity variance. The density of the random variable X is $f_X(x) = \frac{1}{2}f_G(x) + \frac{1}{2}f_G(x-6)$. We consider the squared error criterion.

(a) Compute the differential entropy of X (be careful!).

(b) Derive a formula for the Shannon lower bound for X.

(c) Is the Shannon lower bound tight? If it is, provide the density of the reconstructed signal and your logic.

(d) You have designed an appropriate high-rate, mixture, scalar constrained-entropy (CE) quantizer with two mixture components. Explain the steps your quantizer makes from the data point coming in to the codeword going into the channel.

(e) Derive the rate-distortion relation of your CE quantizer.

(f) Explain the difference between the rate-distortion function and the performance of your CE quantizer in terms of shape, space-filling, and memory advantages.

14. To improve the understanding of high-rate quantization, we approximate a uniform density

$$f_X(x) = \begin{cases} 1, & x \in (0,1] \\ 0, & \text{elsewhere} \end{cases}$$

by a set of N uniformly spaced Gaussian densities:

$$f_X(x) = \sum_{n=1}^{N} \frac{1}{N\sigma_G} f_G((x - \frac{n}{N})/\sigma_G)$$

where $f_G(x)$ is a unit variance Gaussian density and $\sigma_G = 1/N$. An example for $N = 6$ is shown in figure 7.4. In the following we assume that N=100 (i.e., large) and use the squared error criterion.

(a) In this subproblem we do *not* consider the expansion, but use $f_X(x)$ as defined in the first equation of this problem. Compute the Shannon lower bound for $f_X(x)$. Is this lower bound tight? Is your lower bound close to being tight for high or low rates?

(b) Based on the expansion, design a high-rate, constrained-entropy, Gaussian-mixture constrained-entropy quantizer with a mean distortion $D = 10^{-6}$. Provide all parameters needed to describe the quantizer (such as thresholds, step size(s), whatever) and provide a diagram that clearly shows all processing stages in both encoder and decoder.

(c) Provide an operational rate-distortion relation for the basic, unmodified high-rate, constrained-entropy, Gaussian mixture quantizer. Compare your result with the rate-distortion bound that you found in 14a .

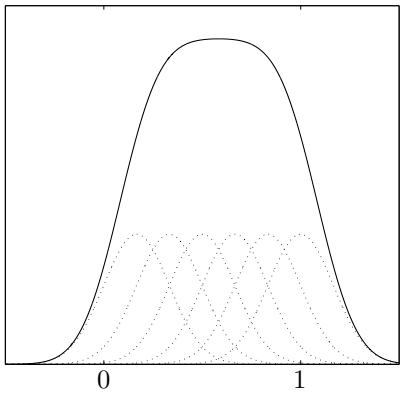

Figure 7.4: The approximation to $f_X(x)$ and its constituent components for $N = 6$ for problem 14 .

(d) For the particular case of this problem, suggest an improved design for the Gaussian mixture quantizer. (No need for any calculations.)

15. Consider a two-dimensional Gaussian density with the identity matrix as covariance matrix and zero mean. The density represents one kernel in a Gaussian mixture model of a probability density function. To create a mixture quantizer, we must design a generic lattice quantizer that applies for each Gaussian kernel separately. We consider the design of a 128-centroid truncated lattice for the Gaussian probability density function.

    (a) Look up the generator matrix for the optimal lattice. Plot a truncated lattice where u_1 and u_2 take the values $\{0, 1, 2, 3\}$. Determine the optimal offset for the lattice for the Gaussian data.

    (b) Using a simple full-search over all centroids, write a program that determines the distortion by stochastic integration. Using this program, search for the optimal scaling of the lattice. Make a scatter plot with an overlay of your best scaled parallelogram-truncated lattice quantizer and list the measured distortion.

    (c) Next consider sublattice based truncation. Determine M so that we get a 16-centroid quantizer. Determine the optimal offset. Write a program that determines the distortion by stochastic integration. Using this program, search for the optimal scaling of the lattice. Make a scatter plot with an overlay of your best scaled parallelogram-truncated lattice quantizer and list the measured distortion.

    (d) For 16 centroids, the rate is 4 bits for the (nonoptimal) constrained-resolution quantizer. Using stochastic integration, determine the best rate possible with an constrained entropy quantizer for the sublattice based truncation.

16. We consider the encoding of the phase of a sinusoid. We model it as a single random variable Φ that is periodic in its amplitude, with period 2π. It has a uniform density of $1/2\pi$ on this interval. We use the squared error criterion.

    (a) Find the Shannon lower bound for the periodic signal.

(b) Argue/prove that the Shannon lower bound is or is not tight.

(c) Derive the reconstruction-point density of the optimal constrained-entropy quantizer. Provide its rate-distortion relation.

(d) Derive the reconstruction-point density of the optimal constrained-resolution quantizer. Provide its rate-distortion relation.

8

Low-Rate Quantization

8.1 Introduction

The lossy encoding of signals is an important aspect of modern communication. Thus-far, we have discussed bounds on the rate given a distortion (or vice-versa) in chapter 6 and the behavior of quantizers when the rate is high in chapter 7. The latter chapter also discussed the design of practical high-rate quantizers. However, it is quite common (e.g., in speech and audio coding) that the high-rate conditions are not satisfied. Under such circumstances, quantizers designed using principles derived from high-rate theory are generally not optimal (although they are often surprisingly good). In this chapter, we describe methods to design practical quantizers that perform well at low rate. These design methods are based on iterative training procedures.

Our discussion of training-based quantization will describe common methods for both constrained-resolution and constrained-entropy training methods. Constrained-resolution quantizers, i.e., quantizers with a finite, given codebook size dominate in existing applications. As was mentioned earlier in section 7.2.2, there are a number of reasons for using constrained-resolution quantizers: an existing infrastructure for fixed-rate, real-time communication, simpler design algorithms, and no requirement for lossless coding. Furthermore, our proof of the rate-distortion theorem in section 6.2.6 shows that, asymptotically with increasing dimension, constrained-resolution quantization can perform at the rate-distortion limit. However, at lower dimensionality, constrained-entropy quantizers will provide a better rate-distortion trade-off. Since there is a clear trend towards both statistical networks and increased computational capacity, constrained-entropy quantizers will likely play an increasingly important role in the future.

The most important topic in this chapter is the **Lloyd algorithm**, a method originally developed at AT&T Bell Laboratories in the nineteen fifties, but published much later [49]. It is also known as the **k-means algorithm**. The Lloyd algorithm, which is ubiquitous in quantization design, is an iterative procedure for improving a constrained-resolution quantizer. The method is applicable to both scalar and vector quantizers. While it was originally developed for the resolution-constrained case, it has been extended to the entropy-constrained case.

We discussed earlier how the rate-distortion theorem (cf. section 6.2.6) suggests that vector quantization can solve all source-coding problems and that this promise cannot be fulfilled because of computational constraints. The computational effort for a full search for the best of all entries in a codebook grows exponentially with the codebook size. One way towards mitigating this problem is to transform the data prior to encoding such that they are more amenable to scalar quantization; two variants on this approach will be discussed in chapters 9 and 10, respectively. Another method of obtaining low-complexity quantization with high fidelity is to reduce the computational requirements of vector quantization. This is generally done by introducing structure into the codebooks and the practical high-rate theory based methods discussed in section 7.6 are an example of that. Structured vector quantization methods that are optimized without high-rate assumptions are discussed in section 8.4 of this chapter. We discuss a number of the most commonly used procedures. It should be remembered that structure in a codebook often implies that the codebooks are not optimal. We also discuss briefly methods for exchanging the computational effort required for vector quantization by large tables. While these methods are optimal, they are not commonly used, since the cost of the required fast accessible memory would be very high.

8.2 Resolution-Constrained Quantization

Practical quantizers for most applications are designed under a resolution constraint: they are designed to minimize mean distortion under the constraint of a given codebook size. Such resolution constrained quantizers form the focus of this section. Entropy-constrained quantizers are discussed in section 8.3.

General design procedures that guarantee optimality of a quantizer for an arbitrary distribution do not exist. As a result, iterative methods that guarantee only local optimality are used instead. We proceed in a manner seen earlier for the EM (section 4.3.4) and Blahut (section 6.5.2) algorithms: we first express the criterion to be minimized (the distortion in this case) as a double minimization. We can then iteratively improve the quantizer by alternating between the two minimizations.

As was described in section 7.2.1, a quantizer is a mapping, $\mathcal{Q}$, of k-dimensional Euclidean space, $\mathbb{R}^k$, onto a countable set of points. It is often convenient to separate the mapping performed by a quantizer into a first mapping $\mathcal{E}$ from the original vector in $\mathbb{R}^k$ to an index, and a second mapping $\mathcal{D}$ from an index to a reconstruction point (centroid) in a codebook $\mathcal{C}^k$. The first mapping is defined by the partition, $\mathtt{V} = \{\mathtt{V}_i\}_{i\in\mathcal{I}}$, and the second mapping is defined by the set of centroids, $\mathcal{C}^k = \{c_i^k\}_{i\in\mathcal{I}}$, where in both cases $\mathcal{I}$ is the set of indices. The minimization of the distortion criterion by optimizing the quantizer can then be written as

$$D_{\min} = \min_{\mathtt{V},\mathcal{C}^k} \mathrm{E}[d(X^k, \mathcal{Q}(X^k))], \tag{8.1}$$

where the minimum is over all allowed partitions and centroid sets. By alternate application of the minimization over $\mathtt{V}$ and $\mathcal{C}^k$, a quantizer is obtained that is locally optimal. We start with a section discussing the optimality conditions, and follow this with a section on quantizer design.

8.2.1 Optimality Conditions

In this section, we discuss the separate optimization of the encoder (the partition) and the decoder (the set of centroids). These conditions can then be used to perform the iterative optimization. We first discuss the encoder, then the decoder.

Optimizing the Encoder

The goal of designing an optimal encoder given the decoder is equivalent to finding a set of cells $\mathtt{V}_i$ that minimizes the mean distortion D of equation 8.1 for a given set of reconstruction points, $\mathcal{C}^k = \{c_i^k\}_{i\in\mathcal{I}}$. Since the distortion can be written as

$$D = \int_{\mathbb{R}^k} f_{X^k}(x^k) d(x^k, \mathcal{Q}(x^k)) dx^k, \tag{8.2}$$

it is clear that D is minimized if $d(x^k, \mathcal{Q}(x^k))$ is minimized for all x^k. That is, optimality of the encoder is ensured if, for an input vector x^k, we select the index i such that

$$i = \underset{m\in\mathcal{I}}{\operatorname{argmin}}\, d(x^k, c_m^k). \tag{8.3}$$

In the following, we will assume that $d(x^k, y^k)$ is a monotonically increasing function of the Euclidean distance[1] $\|x^k - y^k\|$. Most practical distortion measures satisfy this condition, partly because it is usually consistent with the problem at hand, and partly because it makes the quantizer easier to implement.

A minor problem with equation 8.3 is that for $x^k \in \mathbb{R}^k$ the minimum is not always unique, since points exist that are equidistant to two centroids. The nonuniqueness can be eliminated by selecting the index as follows:

$$\begin{aligned} i = \{j \in \mathcal{I} :\ & d(x^k, c_j^k) \leq d(x^k, c_m^k)\ \forall_{m\in\mathcal{I}, m<j}, \\ & d(x^k, c_j^k) < d(x^k, c_m^k)\ \forall_{m\in\mathcal{I}, m>j}\}. \end{aligned} \tag{8.4}$$

The optimal encoder, for given decoder, is then defined by

$$\begin{aligned} \mathtt{V}_i = \{x^k \in \mathbb{R}^k : & d(x^k, c_i^k) \leq d(x^k, c_m^k)\ \forall_{m,i\in\mathcal{I}, m<i}, \\ & d(x^k, c_i^k) < d(x^k, c_m^k)\ \forall_{m,i\in\mathcal{I}, m>i}\}. \end{aligned} \tag{8.5}$$

The cells $\mathtt{V}_i$ of equation 8.5, together with the centroids c_i, define a **nearest-neighbor quantizer** or **Voronoi quantizer**. The cells are sometimes referred to as Voronoi regions (this terminology is more common in the vector quantization case). A Voronoi quantizer is a regular quantizer.

For the scalar case, it is straightforward to find the cell boundaries explicitly. Assuming that the indexing is from left to right, on the real line, the cell boundaries b_i are the points for which

$$d(b_i, c_i) = d(b_i, c_{i+1}). \tag{8.6}$$

For a codebook with $|\mathcal{I}|$ entries, $\{c_i\}_{i\in\mathcal{I}}$, only $|\mathcal{I}| - 1$ boundaries $\{b_1, \cdots, b_{|\mathcal{I}|-1}\}$ are defined.

[1] We remember that distance is the norm of the difference between two points in a Euclidean space.

Example 8.1: Optimal encoder given decoder for the scalar case
Figure 8.1 shows the construction of an optimal encoder given a decoder. The top figure shows a set of reconstruction values, i.e., the decoder, for an interval of $\mathbb{R}$. The bottom figure, shows the optimal encoder for the given decoder. From equation 8.6 it follows that the optimal encoder is simply a partitioning with boundaries in the middle between the reconstruction points (and at the interval ends). This is true independently of the data density.

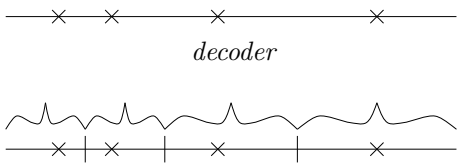

Figure 8.1: Construction of the optimal encoder given a decoder as described in example 8.1. Top: the decoder. Bottom: decoder with the optimal encoder inserted.

For the vector quantization case, the definition of the cell boundaries is in practice much more difficult. Figure 8.2 shows the boundaries for a two-dimensional quantizer. Even for this simple case, the computation of the cell boundaries is not trivial. As a result, the cell boundaries are generally not computed explicitly for the vector case. Instead, the cells are defined simply as the set of points satisfying equation 8.5.

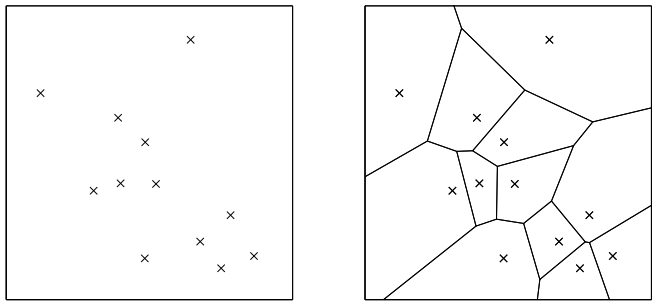

Figure 8.2: Left: a decoder (a set of reconstruction points). Right: the nearest-neighbor (Voronoi) quantizer for this decoder.

Optimizing the Decoder

Next, we turn our attention to the decoder, which is specified by the codebook $\mathcal{C}^k = \{c_i^k\}_{i\in\mathcal{I}}$. Our task is to find the codebook that minimizes D in equation 8.1 given the encoder, i.e., given the set of cells (partition) $\{\mathtt{V}_i\}_{i\in\mathcal{I}}$. Since the cells are fixed, it is

useful to write equation 8.1 in the form

$$\begin{aligned} D_{\min} &= \min_{\mathcal{C}^k} \mathrm{E}[d(X^k, \mathcal{Q}(X^k))] \\ &= \min_{\mathcal{C}^k} \sum_{i \in \mathcal{I}} p_I(i)\, \mathrm{E}[d(X^k, c_i^k) | X^k \in \mathtt{V}_i] \\ &= \sum_{i \in \mathcal{I}} p_I(i) \min_{c_i^k} \mathrm{E}[d(X^k, c_i^k) | X^k \in \mathtt{V}_i]. \end{aligned} \tag{8.7}$$

The encoder (the partition) is fixed, which means that each term in the summation depends on only one codebook entry. We can optimize the terms individually. For a given encoder, the entries of the optimal codebook (the centroids) are given by

$$c_i^k = \operatorname*{argmin}_{y^k \in \mathbb{R}^k} \mathrm{E}[d(X^k, y^k) | X^k \in \mathtt{V}_i]. \tag{8.8}$$

While equation 8.8 is general, it is not very attractive from a computational viewpoint. Fortunately, it is possible to simplify the criterion for the commonly used squared error distortion measure. Noting that c_i^k is deterministic, we have:

$$\begin{aligned} c_i^k &= \operatorname*{argmin}_{y^k \in \mathbb{R}^k} \mathrm{E}[\|X^k - y^k\|^2 | X^k \in \mathtt{V}_i] \\ &= \operatorname*{argmin}_{y^k \in \mathbb{R}^k} \mathrm{E}[\|X^k\|^2 | X^k \in \mathtt{V}_i] - 2y^{kH}\mathrm{E}[X^k | X^k \in \mathtt{V}_i] + \|y^k\|^2 \\ &= \operatorname*{argmin}_{y^k \in \mathbb{R}^k} -2y^{kH}\mathrm{E}[X^k | X^k \in \mathtt{V}_i] + \|y^k\|^2 \\ &= \operatorname*{argmin}_{y^k \in \mathbb{R}^k} \|\mathrm{E}[X^{kH} | X^k \in \mathtt{V}_i]\|^2 - 2y^{kH}\mathrm{E}[X^k | X^k \in \mathtt{V}_i] + \|y^k\|^2 \\ &= \operatorname*{argmin}_{y^k \in \mathbb{R}^k} \|\mathrm{E}[X^k | X^k \in \mathtt{V}_i] - y^k\|^2. \end{aligned} \tag{8.9}$$

Thus, for the squared error distortion measure, the optimal decoder is

$$c_i^k = \mathrm{E}[X^k | X^k \in \mathtt{V}_i]. \tag{8.10}$$

That is, the centroid is the mean of X^k over the cell.

Example 8.2: Optimal decoder given encoder for the scalar case
Figure 8.3 shows the construction of an optimal decoder for the squared error criterion, given an encoder and a density. In the top figure the density is shown, and below that the encoder. In the bottom figure, the centroids, computed according to equation 8.10, are inserted. Note that while the encoder is the same as in figure 8.1, the optimal decoder is not the same as the original decoder of figure 8.1.

8.2.2 The Lloyd Algorithm

The Lloyd training procedure [49] (the k-means algorithm) is based on the optimality conditions discussed in the section 8.2.1. If we use equation 8.5 to optimize the encoder, the mean distortion of the quantizer decreases (or is constant). Similarly, if we use 8.8 to optimize the decoder, the mean distortion of the quantizer decreases (or is constant). Thus, if we optimize the encoder, then optimize the decoder, and repeat these two steps

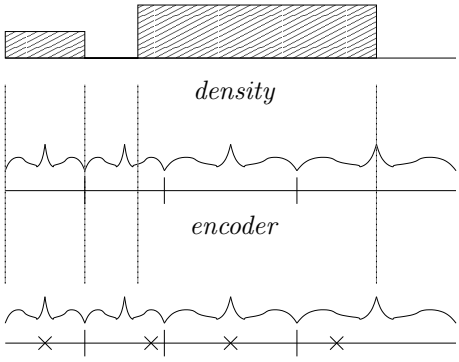

Figure 8.3: Construction of the mean squared error decoder given an encoder as described in example 8.1. Top: the density. Middle: the encoder. Bottom: encoder with the centroids (optimal decoder) inserted.

in alternating order, then the distortion decreases with each optimization step. This iterative procedure is the Lloyd algorithm. In the case of vector quantization, it is often referred to as **generalized Lloyd algorithm** or **GLA**.

Table 8.1 shows the Lloyd algorithm. To obtain the scalar case, simply set $k = 1$. To make the algorithm practical, the boundaries of the cells must be defined explicitly. As was mentioned before, this is generally very complicated in the vector case. Thus, it can be concluded that the algorithm in table 8.1 is useful mainly for the case of scalar quantization.

In our discussion of the optimality conditions and the Lloyd algorithm, we have so-far assumed that the probability density of the random variable to be quantized is known. In most practical situations, this is not the case. Usually, one only has a set of so-called **training data**, and this means that $\mathrm{E}[d(X^k, y^k)|X^k \in V_i^{(q)}]$ cannot be evaluated. A solution is to determine an expression for the density based on the experimental data (we can use the methods discussed in section 4.3, but this approach is not commonly used.

The standard solution to the problem of not knowing the probability density is to use the training data directly. We denote the training data as a set $\mathcal{T}^k$ consisting of points in $\mathbb{R}^k$. We can interpret the training data as being a discrete probability distribution. That is, we have a discrete probability distribution where each training data point has a probability of $1/|\mathcal{T}^k|$, where $|\mathcal{T}^k|$ is the cardinality of the set of training data (i.e., the number of training data). The cells $\mathtt{V}_i$ then become subsets $\mathcal{T}_i^k$ of the training data $\mathcal{T}^k$. If we assume that the decoder, $\mathcal{C}^k$, is known, the sets $\mathcal{T}_i^k$ are

$$\begin{aligned} \mathcal{T}_i^k = \{x^k \in \mathcal{T}^k : d(x^k, c_i^k) &\leq d(x^k, c_m^k)\ \forall_{m,i\in\mathcal{I},m<i}, \\ d(x^k, c_i^k) &< d(x^k, c_m^k)\ \forall_{m,i\in\mathcal{I},m>i}\}. \end{aligned} \tag{8.11}$$

Note that the sets $\mathcal{T}_i^k$ are easily computed in practice. Using the sets $\mathcal{T}_i^k$ as an indication of the distribution f_{X^k}, it is straightforward to estimate the optimal decoder for the

Table 8.1: The Lloyd algorithm for known density function. q is the iteration index, and T a threshold. For clarity, the superscript k has been omitted from the variables.

1. set $q = 1$,
   set $D^{(0)} = \infty$,
   define an initial codebook $\mathcal{C}^{(1)} = \{c_i^{(1)}\}_{i\in\mathcal{I}}$

2. find the optimal encoder:

$$V_i^{(q)} = \{x \in \mathbb{R}^k : d(x, c_i^{(q-1)}) \leq d(x, c_m^{(q-1)}) \ \forall_{m,i\in\mathcal{I}, m<i}, \\ d(x, c_i^{(q-1)}) < d(x, c_m^{(q-1)}) \ \forall_{m,i\in\mathcal{I}, m>i}\}$$

3. find the optimal decoder:

$$c_i^{(q)} = \underset{y\in\mathbb{R}^k}{\text{argmin}}\, \mathrm{E}[d(X, y)|X \in V_i^{(q)}]$$

4. evaluate $D^{(q)} = \sum_{i\in\mathcal{I}} \mathrm{E}[d(X, c_i)|X \in V_i^{(q)}]$;
   stop if $D^{(q-1)} - D^{(q)} < TD^{(q)}$; otherwise set q := q+1 and go to 2

discrete distribution using the relation

$$\mathrm{E}[d(X^k, y^k)|X^k \in V_i^{(q)}] = \frac{1}{|\mathcal{T}_i^k|} \sum_{x^k\in\mathcal{T}_i^k} d(x^k, y^k), \tag{8.12}$$

where $|\mathcal{T}_i^k|$ is the cardinality of the set $\mathcal{T}_i^k$. Having defined a new decoder, we can compute a new set of sets $\{\mathcal{T}_i^k\}_{i\in\mathcal{I}}$. The **discrete Lloyd algorithm** or **discrete GLA** consists of alternatingly optimizing the encoder and decoder . Upon completion of the training, we convert the cell definitions so that they apply to $\mathbb{R}^k$ instead of $\mathcal{T}$. That is, we assume that the discrete distribution leads to a set of centroids that is accurate for the underlying continuous distribution. The complete algorithm is given in table 8.2.

The discrete Lloyd algorithm has several advantages over the basic Lloyd algorithm. First, there is no need to define the cell boundaries, since the mean distortion for a cell is replaced by a sum over a known finite set of data points $\mathcal{T}_i^k$. Second, the probability density of the data does not need to be estimated. Third, the discrete Lloyd algorithm is guaranteed to converge in a finite number of iterations. This last advantage exists since only a finite number of partitions of $\mathcal{T}^k$ into sets $\mathcal{T}_i^k$ is possible, and since the change in the distortion is nonnegative at each iteration.

For the discrete Lloyd algorithm to perform well, the partition of $\mathcal{T}^k$ must provide a reasonable representation of the actual Voronoi region in $\mathbb{R}$. This can only be the case if the number of data points is sufficiently large. It is often suggested that in practice at least 10 points are required for each cell.

Table 8.2: The discrete (generalized) Lloyd algorithm. q is the iteration index, and T a threshold. For clarity, the superscript k has been omitted from the variables.

1. set $q = 1$,
   set $D^{(0)} = \infty$,
   define an initial codebook $\mathcal{C}^{(1)} = \{c_i^{(1)}\}_{i\in\mathcal{I}}$

2. find the sets of data points $\{\mathcal{T}_i^{(q)}\}_{i\in\mathcal{I}}$ such that

$$\begin{aligned}\mathcal{T}_i^{(q)} = \{x \in \mathcal{T} : d(x, c_i^{(q-1)}) &\leq d(x, c_m^{(q-1)})\ \forall_{m,i\in\mathcal{I},\, m<i},\\ d(x, c_i^{(q-1)}) &< d(x, c_m^{(q-1)})\ \forall_{m,i\in\mathcal{I}\, m>i}\}\end{aligned}$$

3. find the optimal decoder:

$$c_i^{(q)} = \underset{y\in\mathbb{R}^k}{\operatorname{argmin}} \sum_{x\in\mathcal{T}_i^{(q)}} d(x, y)$$

4. evaluate $D^{(q)} = \sum_{i\in\mathcal{I}} \sum_{x\in\mathcal{T}_i^{(q)}} d(x, c^{(q)})$
   if $D^{(q-1)} - D^{(q)} < TD^{(q)}$ go to 5; otherwise set q := q+1 and go to 2

5. define the corresponding partition in $\mathbb{R}^k$:

$$\begin{aligned}\mathtt{V}_i = \{x \in \mathbb{R}^k : d(x, c_i^{(q)}) &\leq d(x, c_m^{(q)})\ \forall_{m,i\in\mathcal{I},\, m<i},\\ d(x, c_i^{(q)}) &< d(x, c_m^{(q)})\ \forall_{m,i\in\mathcal{I},\ m>i}\}\end{aligned}$$

The LBG Initialization Procedure

As mentioned before, the Lloyd algorithm is locally optimal, i.e., any small change in the Voronoi regions or in the reconstruction points results in a degradation of performance. However, practical probability distributions often have many such minima. As a result, it is found that the Lloyd algorithm, particularly for the vector case, is sensitive to the initial codebook that is used.

A good design of the initial codebook is, therefore, important. Often, the training procedure is run multiple times, with different initial codebooks, and the best codebook is selected from these multiple runs. An alternative initialization procedure, which generally performs well, and which is commonly used, is the **LBG algorithm**, named after the authors who originally proposed the procedure [50]. (While there is some confusion about naming conventions, we adopt the nomenclature of [24] here.) The LBG algorithm starts from a zero-bit (single-entry) codebook that is the mean of the distribution. The size of the codebook is multiplied by two, by "splitting" the centroid into two centroids: one that is identical to the original centroid, and one that is a small random distance away from the first centroid. This codebook is then subjected to the GLA, and, upon convergence, the codebook is split again, and so-on. Thus, the LBG algorithm generates codebooks of a size that is a power of 2 (which renders codewords of integer length in bits). It is described more formally in table 8.3.

Table 8.3: LBG splitting algorithm for codebook initialization and training (m is a splitting index).

1. set $m = 0$
   set $\mathcal{C}^{(0)} = \{E[X^k]\}$
2. stop if 2^m is desired codebook size
3. generate 2^m random vectors ϵ_i^k, small $\|\epsilon_i^k\|$
   define: $\mathcal{C}^{(m+1)} = \{c_1^k, c_1^k + \epsilon_1^k, c_2^k, c_2^k + \epsilon_2^k, \cdots\}$
4. starting from the $\mathcal{C}^{(m+1)}$ defined in 3 as an initial condition, train $\mathcal{C}^{(m+1)}$ with the discrete Lloyd algorithm
5. set m := m+1 and go to 2

The Equidistortion Correction

Another useful procedure that lowers the chance of obtaining a local optimum with relatively low quantizer performance is based on the equidistortion criterion. The procedure does not guarantee that performance improves, but it often does result in significant performance improvements. The basic idea behind the method is that the equidistortion principle, which was shown to hold at high resolution in section 7.4.2, should be approximately valid at lower resolutions [34]. To satisfy this requirement, the contribution of a cell to the overall distortion is explicitly evaluated for each cell at regular intervals during the iterations. Let us assume that each point in the data base is unique. Then the cell contribution to the overall distortion can be written as

$$D_i = \sum_{x^k \in \mathcal{T}_i^k} d(x^k, c_i^k). \tag{8.13}$$

The cells with maximum and minimum D_i are selected and if the ratio of their distortion exceeds a threshold, the cell with the lowest contribution to the distortion is removed, and the cell with the highest distortion is split. We refer to this method as the **equidistortion correction**. The algorithm is formalized in table 8.4. The algorithm in table 8.4 should be inserted between steps 4 and 2 of table 8.2.

8.2.3 Quantization Error and the Lloyd Algorithm

It is useful to have quantitative information about the nature of the quantization error. We know from rate-distortion theory (cf. section 6.4.2) that, at least when the Shannon lower bound is tight, the quantization error and the source, x^k, are dependent; we furthermore saw that the quantization error is independent of the reconstructed variable $\mathcal{Q}(x^k)$. These relations are reflected in the outcome of the Lloyd algorithm for the squared error criterion: the quantization error and the reconstructed variable are uncorrelated while the source and the quantization error have nonvanishing correlation. That the correlation between the reconstructed variable and the quantization error vanishes is easily derived by using the optimal decoder condition (equation 8.10):

$$\mathrm{E}[(\mathcal{Q}(X^k) - X^k)\mathcal{Q}(X^k)^T] = \sum_{i \in \mathcal{I}} p_I(i)\mathrm{E}[c_i^k - X^k |\, X^k \in \mathtt{V}_i] c_i^{kT} = 0. \tag{8.14}$$

Table 8.4: Equidistortion correction that is to be inserted in the discrete generalized Lloyd algorithm. T_e is a threshold.

1. consider codebook: $\mathcal{C}^{(q)} = \{c_i^k\}_{i\in\mathcal{I}}$
2. find cells with maximum and minimum contributions to mean distortion;

$$i_{\min} = \underset{i\in\mathcal{I}}{\operatorname{argmin}}\, D_i$$

$$i_{\max} = \underset{i\in\mathcal{I}}{\operatorname{argmax}}\, D_i$$

3. if $\frac{D_{i_{\max}}}{D_{i_{\min}}} > T_i$ then:
   (a) generate random vector ϵ^k, $|\epsilon^k|$ small
   (b) replace $c^k_{i_{\min}}$ of codebook with $c^k_{i_{\max}} + \epsilon^k$

The correlation matrix for quantization error then satisfies

$$\begin{aligned}
&\mathrm{E}[(\mathcal{Q}(X^k) - X^k)(\mathcal{Q}(X^k) - X^k)^T] \\
&\quad = -\mathrm{E}[(\mathcal{Q}(X^k) - X^k)X^{kT}] + \mathrm{E}[(\mathcal{Q}(X^k) - X^k)\mathcal{Q}(X^k)^T] \\
&\quad = -\mathrm{E}[(Q(X^k) - X^k)X^{kT}]. \qquad (8.15)
\end{aligned}$$

8.3 Entropy-Constrained Quantization

The strong trend towards statistical networks has made training methods for entropy-constrained quantizers more important. In this section, we extend the methods developed in section 8.2 for the resolution-constrained case to apply to the entropy-constrained case. The principles of the approach presented in this section were first introduced in [51].

As was discussed before, we can consider a quantizer to consist of a first mapping $\mathcal{E}$, defined by the partition $\mathtt{V} = \{\mathtt{V}_i\}_{i\in\mathcal{I}}$, from the original vector in $\mathbb{R}^k$ to an index, and a second mapping $\mathcal{D}$, defined by a codebook, $\mathcal{C}^k = \{c_i^k\}_{i\in\mathcal{I}}$, from an index to a centroid from a codebook $\mathcal{C}^k$. In the constrained-entropy case, the minimizaton of the distortion criterion, $D = \mathrm{E}[d(X^k, \mathcal{Q}(X^k))]$, with respect to $\mathtt{V}$ and $\mathcal{C}^k$ must now be performed under an entropy constraint on the quantization indices. This constraint can be written as

$$\begin{aligned}
R &= -\sum_{i\in\mathcal{I}} p_I(i)\log(p_I(i)) \\
&= -\sum_{i\in\mathcal{I}} \int_{\mathtt{V}_i} f_{X^k}(x^k)dx^k \log(\int_{\mathtt{V}_i} f_{X^k}(x^k)dx^k), \qquad (8.16)
\end{aligned}$$

where R is the given rate. Let us define an equivalent codeword length for each cell[2]:

$$l_{\mathtt{V}}(i) = -\log(\int_{\mathtt{V}_i} f_{X^k}(x^k)dx^k). \qquad (8.17)$$

[2]We recall from chapter 5 that such noninteger rates per symbol can be approached by means of arithmetic coding.

The entropy constraint can then be written as

$$R = \mathrm{E}[l_{\mathtt{V}}(i(X^k))], \tag{8.18}$$

where the dependency of i on X^k is made explicit. Using the method of Lagrange multipliers to minimize distortion under the entropy constaint, the objective is then to find the $\mathcal{V}$ and $\mathcal{C}^k$ that minimize the extended criterion

$$\eta \quad = \quad \mathrm{E}[d(X^k, \mathcal{Q}(X^k))] + \lambda \mathrm{E}[l_{\mathtt{V}}(i(X^k))], \tag{8.19}$$

where λ is a Lagrange multiplier and where we omitted the constant term $-\lambda R$. A local minimum can be found by alternating the minimization over $\mathcal{V}$ with that of $\mathcal{C}^k$. It is seen from equaton 8.19 that the Lagrange multiplier assigns a relative weight to the distortion and the rate. That is, with increasing λ we can expect the minimization to result in a rate-distortion pair with lower rate and higher distortion.

Even the iterative minimization of equation 8.19 is generally not straightforward. The minimization over the encoder, i.e., over the partitions $\mathcal{V}$, is complicated by the dependency of $l_{\mathtt{V}}$ on the partition. This problem can be eliminated if we consider the results of section 5.2.1 to realize that the set of codeword lengths $\mathcal{L}_{\mathtt{V}} = \{l_{\mathtt{V}}(i)\}_{i\in\mathcal{I}}$ minimizes the expected codeword length $\mathrm{E}[l(i(X^k))]$ that permits unique decoding for a given partition $\mathcal{V}$ and density $f_{X^k}(x^k)$. The minimum can then be written as

$$\eta_{\min} \quad = \quad \min_{\mathcal{V},\mathcal{C}^k,\mathcal{L}} \ \mathrm{E}[d(X^k, \mathcal{Q}(X^k))] + \lambda \mathrm{E}[l(i(X^k))], \tag{8.20}$$

where $\mathcal{L} = \{l(i)\}_{i\in\mathcal{I}}$ is a set of codeword lengths that facilitates unique decoding given the encoder. A locally optimal constrained-entropy quantizer can then be found by iteratively optimizing the partition (encoder), set of centroids (decoder), and set of codeword lengths.

8.3.1 Optimality Conditions

In this section, we specify three optimality conditions to minimize the expression of equation 8.20 iteratively. We first consider the optimal encoder, then the optimal set of codeword lengths, and finally the optimal decoder.

By considering equation 8.20, we see that the optimal encoder given the decoder and the set of codeword lengths is obtained by minimizing $d(x^k, \mathcal{Q}(x^k)) + \lambda l(i(x^k))$ for all x^k. Proceeding as for the constrained-resolution case (cf. section 8.2), we obtain as optimal encoder $\mathcal{V} = \{\mathtt{V}_i\}_{i\in\mathcal{I}}$ for a given decoder $\mathcal{C}^k = \{c_i^k\}_{i\in\mathcal{I}}$ and set of codeword lengths $\mathcal{L} = \{l(i)\}_{i\in\mathcal{I}}$:

$$\begin{aligned} \mathtt{V}_i = \{x^k \in \mathbb{R}^k : d(x^k, c_i^k) + \lambda l(i) \quad &\leq \quad d(x^k, c_m^k) + \lambda l(m) \ \forall_{m,i\in\mathcal{I},m<i}, \\ d(x^k, c_i^k) + \lambda l(i) \quad &< \quad d(x^k, c_m^k) + \lambda l(m) \ \forall_{m,i\in\mathcal{I},m>i}\}. \end{aligned} \tag{8.21}$$

Next, we consider the optimal set of codeword lengths, given the encoder and decoder. This task is trivial, since we introduced this optimization to facilitate the optimization of the partition $\mathcal{V}$ by decoupling the set of codeword lengths $\mathcal{L}$ from the partition. The optimal $\mathcal{L}$ is $\mathcal{L}_{\mathtt{V}}$, and the individual $l_{\mathtt{V}}(i)$ are given by equation 8.17. We note that the optimal set of codeword lengths does not depend on the decoder.

Finally, we consider the optimal decoder given the set of codeword lengths and the encoder. The optimal decoder $\mathcal{C}^k$ for a given encoder $\mathcal{V}$ is based on the minimization

$$\begin{aligned} D_{\min} &= \min_{\mathcal{C}^k} \mathrm{E}[d(X^k, \mathcal{Q}(X^k))] + \lambda \mathrm{E}[l(i(X^k))] \\ &= \min_{\mathcal{C}^k} \sum_{i \in \mathcal{I}} p_I(i)\, \mathrm{E}[d(X^k, c_i^k) | X^k \in \mathtt{V}_i] \\ &= \sum_{i \in \mathcal{I}} p_I(i) \min_{c_i^k} \mathrm{E}[d(X^k, c_i^k) | X^k \in \mathtt{V}_i]. \end{aligned} \tag{8.22}$$

Thus, the optimal decoder given the encoder for the constrained entropy case is identical to that for the constrained-resolution case:

$$c_i^k = \operatorname*{argmin}_{y^k \in \mathbb{R}^k} \mathrm{E}[d(X^k, y^k) | X^k \in \mathtt{V}_i]. \tag{8.23}$$

Having defined the optimality conditions we can now proceed with deriving a modified Lloyd algorithm that applies to the constrained-entropy case.

8.3.2 The Constrained-Entropy Lloyd Algorithm

A constrained-entropy generalized Lloyd algorithm (constrained-entropy GLA) is obtained by selecting a λ, i.e., a particular weighting of distortion versus rate, and iterating over the three the optimality conditions described in section 8.3.1. This leads to a particular rate and distortion (R, D) pair. To obtain a desired rate or distortion, one has to try different values of λ. For the case $\lambda = 0$, the algorithm reduces to the constrained-resolution GLA.

As in the constrained-resolution case, the probability density function is generally unknown, and even when it is known, the description of the partition is difficult to accomplish for the multi-dimensional case. In practice, the constrained-entropy GLA is used for a given set of training data $\mathcal{T}^k$. As for the constrained-resolution case, usage of the training data as probability distribution eliminates the need for knowing the probability density, the need for defining the cell boundaries and guarantees convergence in a finite number of iterations.

The discrete version of the constrained-entropy GLA algorithm is obtained by replacing the continuous density with the discrete data distribution. As before, we define the cells as subsets $\mathcal{T}_i^k$ of the database $\mathcal{T}$. For good performance, all cells need to be populated sufficiently (typically 10 or more data points per cell). For the discrete distribution given by the training data we can compute the optimal codeword length as

$$l_{\mathtt{V}}(i) = -\log(p_{\mathcal{I}}(i)) = -\log(P(\mathcal{T}_i^k)) = -\log(\frac{|\mathcal{T}_i^k|}{|\mathcal{T}^k|}). \tag{8.24}$$

The conversion to the discrete data distribution of the optimal encoder and decoder conditions (equations 8.21 and 8.23) are identical to the corresponding conversions for the constrained-resolution case. The resulting discrete constrained-entropy GLA is given in table 8.5.

It is interesting to note that the constrained-entropy GLA is based on a fixed codebook size. The size of the codebook forms a constraint on the performance of the algorithm.

At least in principle, best performance should be obtained with the largest codebook. Naturally, for a given codebook size, the constraining effect of the codebook size becomes increasingly important with increasing rate.

As for the constrained-resolution GLA, the constrained-entropy GLA is sensitive to the configuration of the initial codebook. While the LBG algorithm can be used for initialization, this is not natural for the entropy-constrained case. Instead, we can use the high-resolution results to create a good initialization procedure. We know that optimal high-rate entropy-constrained quantizers are uniform. This implies that a simple truncated uniform hypercube lattice forms a reasonable starting point for an entropy-constrained quantizer.

Table 8.5: The discrete constrained-entropy GLA for given λ. The value of λ determines the rate versus distortion trade-off. q is the iteration index, and T a threshold. For clarity, the superscript k has been omitted.

1. select λ
   set $\eta^{(0)} = \infty$,
   define an initial codebook $\mathcal{C}^{(0)} = \{c_i^{(0)}\}_{i\in\mathcal{I}}$
   define an initial set of codeword lengths $\mathcal{L}^{(0)} = \{l^{(0)}(i)\}_{i\in\mathcal{I}}$
   set $q = 1$

2. find the sets of data points $\{\mathcal{T}_i^{(q)}\}_{i\in\mathcal{I}}$ such that

$$\begin{aligned}\mathcal{T}_i^{(q)} = \{x \in \mathcal{T} : d(x, c_i^{(q-1)}) + \lambda l^{(q-1)}(i) &\leq d(x, c_m^{(q-1)}) + \lambda l^{(q-1)}(m)\ \forall_{m,i\in\mathcal{I},\, m<i},\\ d(x, c_i^{(q-1)}) + \lambda l^{(q-1)}(i) &< d(x, c_m^{(q-1)}) + \lambda l^{(q-1)}(m)\ \forall_{m,i\in\mathcal{I}\, m>i}\}\end{aligned}$$

3. find the optimal set of codeword lengths:

$$l^{(q)}(i) = -\log(p_I(i)) = -\log(\frac{|\mathcal{T}_i^{(q)}|}{|\mathcal{T}|}),\ \forall_{i\in\mathcal{I}}$$

4. find the optimal decoder:

$$c_i^{(q)} = \underset{y\in\mathbb{R}^k}{\text{argmin}} \sum_{x\in\mathcal{T}_i^{(q)}} d(x, y)$$

5. evaluate $\eta^{(q)} = \sum_{i\in\mathcal{I}} \sum_{x\in\mathcal{T}_i^{(q)}} d(x, c_i^{(q)}) + \lambda l^{(q)}(i)$
   if $\eta^{(q-1)} - \eta^{(q)} < T\eta^{(q)}$ go to 6; otherwise set $q := q + 1$ and go to 2

6. define the corresponding partition in $\mathbb{R}^k$:

$$\begin{aligned}\mathtt{V}_i = \{x \in \mathbb{R}^k : d(x, c_i^{(q)}) + \lambda l^{(q)}(i) &\leq d(x, c_m^{(q)}) + \lambda l^{(q)}(m)\ \forall_{m,i\in\mathcal{I},\, m<i},\\ d(x, c_i^{(q)}) + \lambda l^{(q)}(i) &< d(x, c_m^{(q)}) + \lambda l^{(q)}(m)\ \forall_{m,i\in\mathcal{I},\ m>i}\}\end{aligned}$$

8.4 Structured Vector Quantizers

In a vector quantizer, the transmitted index is selected by searching through a codebook. If an exhaustive search must be performed, the computational complexity of this search depends exponentially on the codeword length. The size of the required training data base grows similarly rapidly with a general vector quantizer design. Thus, despite the rapid advances in hardware technology, there is a strong incentive for using structured vector quantizers that have reduced computational and training data requirements. In this section, a brief overview of some of the most important techniques will be given. Most of the descriptions will focus on constrained-resolution quantizers, which are most commonly used. For a more complete overview we refer to [24].

Most structured vector quantizers fall into the class of **product code vector quantizers**. The essence of this approach is to divide up one large task of high complexity into a number of smaller tasks of much lower total complexity. This is generally accomplished by structuring the codebook so that the search for the codebook entries with the lowest distortion becomes equivalent to several searches through two or more sub-codebooks of much smaller size. The index into the structured codebook can then be seen as the product of the indices into the sub-codebooks (the codeword becomes the concatenation of the codewords for each sub-codebook). We note that a structured codebook involving a search through two codebooks of 1000 entries each has an effective codebook size with 10^6 entries, but with a complexity that is a factor 500 lower. In certain cases (e.g., the gain-shape and mean-removed quantizers) independent or sequential searches through the sub-codebooks result in the optimal product codebook entry, but in general a sequential or independent search through the sub-codebooks (the only way to obtain the computational advantage) does not necessarily result in the best combination of indices in the sub-codebooks. In addition, the structure generally means that the codebook is nonoptimal.

8.4.1 The Tree-Structured Quantizer

A tree-structured quantizer combines a number of vector quantizers, each of relatively low complexity. Let us first consider the encoding with a tree quantizer. We consider a symmetric tree (this is consistent with constrained resolution). We start with defining a codebook for the root node, $\mathcal{C}_0^k = \{c_i^k\}_{i\in\mathcal{I}_0}$, where the subscript indicates the tree depth. We define a unique codebook at tree depth one for each index selected at tree depth zero, and we indicate this by writing the depth-one codebooks as $\mathcal{C}_1^k(i_0)$. At tree depth two, we have codebooks $\mathcal{C}_2^k(i_0, i_1)$, and so on. The quantizer at depth zero renders an index i_0 defined as

$$\begin{aligned} i_0 &= \operatorname*{argmin}_{i\in\mathcal{I}_0} d(x^k, c_i^k) \\ &= \mathcal{E}(x^k|\mathcal{C}_0^k), \end{aligned} \tag{8.25}$$

where we introduced the notation $\mathcal{E}(x^k|\mathcal{C}_0^k)$ for the mapping $\mathbb{R}^k$ onto the set of codebook indices $\mathcal{I}_0$ that is associated with finding the nearest neighbor in the codebook $\mathcal{C}_0^k$. If we define the initial residual error vector to be the input vector, $e_0^k = x^k$, then we can set up a recursion of the form

$$e_m^k = e_{m-1}^k - \mathcal{Q}(e_{m-1}^k|\mathcal{C}_{m-1}^k(i_0, i_1, \cdots, i_{m-2})), \tag{8.26}$$

with

$$i_l = \mathcal{E}(e_l^k | \mathcal{C}_l^k(i_0, \cdots, i_{l-1})), \tag{8.27}$$

where $\mathcal{Q}(e_m^k | \mathcal{C}_m^k(i_0, i_1, \cdots, i_{m-1}))$ is the mapping from $\mathbb{R}^k$ to $\mathcal{C}_m^k(i_0, i_1, \cdots, i_{m-1})$. We assume for simplicity that all codebooks at a particular level l have identical cardinality. For a codebook of depth m, the composite index will be a concatenation of $m+1$ indices: $i_0 \cdots i_m$. In practical applications, the distortion criterion used for the sub-codebooks of the tree-structured codebook is generally set to be identical for all sub-codebooks.

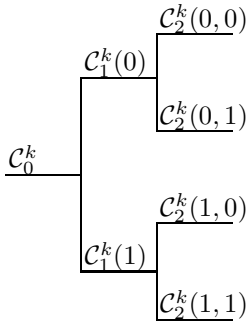

Figure 8.4: A binary tree-structured vector quantization codebook.

The training of a tree-structured codebook is straightforward. For the root node one applies the Lloyd algorithm. Next, consider a particular codebook at the node associated with index i_0. To train this codebook, we must first encode all data with codebook $\mathcal{C}_0^k$. We select all training data associated with i_0, subtract the centroid, and use this as the training data set for $\mathcal{C}_1^k(i_0)$. We continue this procedure for nodes at greater depths.

8.4.2 The Multi-Stage and Split Vector Quantizers

The multi-stage vector quantizer can be seen as a constrained form of the tree-structured vector quantizer. The multi-stage vector quantizer is a tree-structured vector quantizer where the codebooks are dependent only on the depth of the quantizer in the tree and not on the indices of the lower-depth quantizers. Thus, for a multi-stage quantizer we have that $\mathcal{C}_m^k(i_0, i_1, \cdots, i_{m-1}) = \mathcal{C}_m^k$. To be more specific, in figure 8.4 we would have $\mathcal{C}_1^k(0) = \mathcal{C}_1^k(1) = \mathcal{C}_1^k$ and $\mathcal{C}_2^k(0,0) = \mathcal{C}_2^k(0,1) = \mathcal{C}_2^k(1,0) = \mathcal{C}_2^k(1,1) = \mathcal{C}_2^k$.

The training of a multi-stage codebook must be modified accordingly. For the root node one again applies the Lloyd algorithm. To train the codebook $\mathcal{C}_1^k$, one computes the residual error $e_1 = e_0 - \mathcal{Q}(e_0 | \mathcal{C}_0^k)$ for the entire data base. This method is continued for greater depths.

As with the tree-structured codebook, the error criterion used for the sub-codebooks is generally the same. However, the **split** vector quantizer can be interpreted as a special case of the multi-stage vector quantizer with a distortion criterion that varies with the depth. At each depth into the tree, the distortion criterion considers only a sub-vector of the input vector. Thus, in the split vector quantizer, each vector is split

into sub-vectors, and these sub-vectors are quantized separately. If the sub-vectors are relatively independent, split vector quantizers can perform well. This has been shown, for example, for quantization of (a transform of the) linear-prediction coefficients [52]. To make the sub-vectors relatively independent, transformations are useful.

8.4.3 The Gain-Shape Quantizer

In the gain-shape vector quantizer, the input vector x^k (here assumed to be real) is approximated by $\hat{x}^k = \lambda c^k$, where λ is a scalar from a codebook of scalar gains, and where c^k is a vector from a codebook of shape vectors of unit Euclidean norm. Usually, a squared error distortion criterion is used:

$$d(x^k, \lambda c^k) = (x^k - \lambda c^k)^T W (x^k - \lambda c^k), \tag{8.28}$$

where W is a real and symmetric ($W = W^T$) weighting matrix. Often, the weighting matrix varies from one quantization operation to the next.

The most common procedure for gain-shape quantization is outlined as:

1. Obtain an expression for the optimal gain, given an arbitrary shape.
2. Select the best shape vector from the shape codebook assuming the optimal gain.
3. Select the best gain value from the gain codebook for the selected shape vector.

We now go through these steps. Let us consider a particular input vector x^k and a shape vector c^k. The optimal gain that minimizes $d(x^k, \lambda c^k)$ for a given c^k is

$$\begin{aligned}
\lambda_{\text{opt}}(c^k) &= \underset{\lambda \in \mathbb{R}}{\text{argmin}} \ (x^k - \lambda c^k)^T W (x^k - \lambda c^k) \\
&= \underset{\lambda \in \mathbb{R}}{\text{argmin}} \ -2\lambda c^{kT} W x^k + \lambda^2 c^{kT} W c^k \\
&= \underset{\lambda \in \mathbb{R}}{\text{argmin}} \ \lambda^2 - 2\lambda \frac{x^{kT} W c^k}{c^{kT} W c^k} \\
&= \underset{\lambda \in \mathbb{R}}{\text{argmin}} \ (\lambda - \frac{x^{kT} W c^k}{c^{kT} W c^k})^2 \\
&= \frac{x^{kT} W c^k}{c^{kT} W c^k}.
\end{aligned} \tag{8.29}$$

Equation 8.29 can now be used for the second step in the quantization procedure, the quantization of the shape, assuming the optimal gain. Let $\mathcal{C}_{c^k} = \{c_i^k\}_{i \in \mathcal{A}}$ be the shape codebook with index set $\mathcal{I}_{c^k}$. We then see that the optimal shape index assuming the optimal gain is

$$\begin{aligned}
i_c &= \underset{i \in \mathcal{I}_{c^k}}{\text{argmin}} \ (x^k - \lambda_{\text{opt}}(c_i^k) c_i^k)^T W (x^k - \lambda_{\text{opt}}(c_i^k) c_i^k) \\
&= \underset{i \in \mathcal{I}_{c^k}}{\text{argmax}} \ \frac{(x^{kT} W c_i^k)^2}{c_i^{kT} W c_i^k}.
\end{aligned} \tag{8.30}$$

Next we quantize the gain. Let the gain codebook be $\mathcal{C}_\lambda = \{\lambda_i\}_{i\in\mathcal{I}_\lambda}$ with index set $\mathcal{I}_\lambda$. Using the same reasoning as in equation 8.29, the index i_λ of the optimal gain codebook entry is then

$$\begin{aligned} i_\lambda &= \underset{i\in\mathcal{I}_\lambda}{\operatorname{argmin}} \ (x^k - \lambda_i c^k)^T W (x^k - \lambda_i c^k) \\ &= \underset{i\in\mathcal{I}_\lambda}{\operatorname{argmin}} \ (\lambda_i - \lambda_{\text{opt}}(c^k))^2. \end{aligned} \tag{8.31}$$

Thus, if the selected shape vector is c^k, straightforward quantization of the optimal gain for this vector using the squared error criterion minimizes the overall distortion.

In the special case that W does not vary, which means that c_i^k can be normalized such that $|c_i^{kT} W c_i^k| = 1$, and that the gain codebook is symmetric around zero, it is possible to find the optimal entry in the product codebook with only two sequential searches. To make this possible, the shape codebook must satisfy $|c_i^{kT} W c_i^k| = 1$ for all entries. In this case, it follows that

$$\begin{aligned} \min_{i\in\mathcal{I}_\lambda, j\in\mathcal{I}_{c^k}} d(x^k - \lambda_i c_j^k) &= \min_{i\in\mathcal{I}_\lambda, j\in\mathcal{I}_{c^k}} (x^k - \lambda_i c_j^k)^T W (x^k - \lambda_i c_j^k) \\ &= \min_{i\in\mathcal{I}_\lambda, j\in\mathcal{I}_{c^k}} -2\lambda_i c_j^{kT} W x^k + \lambda_i^2 \\ &= \min_{i\in\mathcal{I}_\lambda} (\lambda_i^2 - 2\lambda_i \max_{j\in\mathcal{I}_{c^k}} (|c_j^{kT} W x^k|)). \end{aligned} \tag{8.32}$$

From equation 8.32 it is seen that the product codebook $\mathcal{C}_\lambda \mathcal{C}_{c^k}$ is optimally searched if we first search the shape by maximizing the criterion $c_j^{kT} W x^k$ and then search the gain codebook by minimizing the criterion in equation 8.32 (which is equivalent to the final expression in equation 8.31).

The gain-shape quantizer is quite common in the area of speech coding, e.g., [53, 54], where the dynamic range of the signal often varies greatly. Unfortunately, in these gain-shape quantizers W varies with the speech, and the product codebook entry found by the usual sequential search in the sub-codebooks is thus not guaranteed to be the best.

8.4.4 The Mean-Removed Vector Quantizer

The mean-removed quantizer is somewhat similar to the gain-shape quantizer in philosophy. In the mean-removed quantizer, the mean of the input vector is computed first and then subtracted from the data vectors. Let 1^k denote the k-dimensional vector with all k elements being 1: $1^k = [1, 1, \cdots, 1]^T$. Furthermore, $\mathcal{C}_\mu$ is the codebook for the mean value and $\mathcal{C}_{c^k}$ is the codebook for the mean-removed vector[3]. We restrict the mean-removed vector codebook to be zero mean, i.e., $1^{kT} c^k = 0$, and we denote the

[3]For notational convenience, we select directly from the codebooks $\mathcal{C}_\mu$ and $\mathcal{C}_{c^k}$ rather than from index sets as in section 8.4.3.

mean of x^k by $\mu_x = \frac{1}{k}1^{kT}x^k$. For a simple squared error criterion we then have that

$$\begin{aligned}
&\min_{\mu\in\mathcal{C}_\mu, c^k\in\mathcal{C}_{c^k}} d(x^k, \mu 1^k + c^k) \\
&= \min_{\mu\in\mathcal{C}_\mu, c^k\in\mathcal{C}_{c^k}} (x^k - \mu 1^k - c^k)^T (x^k - \mu 1^k - c^k) \\
&= \min_{\mu\in\mathcal{C}_\mu, c^k\in\mathcal{C}_{c^k}} -2\mu x^{kT} 1^k - 2x^{kT} c^k + 2\mu 1^{kT} c^k + k\mu^2 + c^{kT} c^k \\
&= \min_{\mu\in\mathcal{C}_\mu, c^k\in\mathcal{C}_{c^k}} -2k\mu\mu_x + k\mu^2 - 2x^{kT} c^k + 2\mu_x 1^{kT} c^k + c^{kT} c^k \\
&= \min_{\mu\in\mathcal{C}_\mu, c^k\in\mathcal{C}_{c^k}} k(\mu - \mu_x)^2 + (x^k - \mu_x 1^k - c^k)^T (x^k - \mu_x 1^k - c^k), \qquad (8.33)
\end{aligned}$$

where we exploited that $1^{kT}c^k = 0$. Equation 8.33 shows that an independent quantization of the mean and the residual shape vector $x^k - \mu_x 1^k$ results in the optimal entry in the product codebook.

8.5 Unstructured Quantizers: Fast Search Methods

We saw that the motivation for structured quantizers is that they reduce computational and storage requirements. In this section, we show that it is possible to reduce the computational requirements of unstructured nearest-neighbor (Voronoi) quantizers over that required by a full search. Unfortunately, this generally results in an increase of the storage requirements, but in certain cases this increase may be manageable.

8.5.1 Neighbor Descent

Neighbor descent methods employ a table that stores for each Voronoi cell the indices for the neighboring Voronoi cells. (The determination of the set of neighboring cells is a nontrivial task that we will not discuss.) Using this table, methods can be designed that, on average, find the entry in a codebook that minimizes distortion very fast. An algorithm to find the nearest centroid c_i^k for a vector x^k is:

1. Select a cell index j to start the iteration procedure.
2. Compute $d(c_n^k, x^k)$ for all neighbors of j and select the centroid n that minimizes $d(c_n^k, x^k)$.
3. If $d(c_n^k, x^k) < d(c_j^k, x^k)$ set $j = n$ and go to 2, else set $i = j$ and terminate.

A variation on this algorithm makes it computationally faster [55]:

1. Select a cell index j to start the iteration procedure.
2. Compute $d(c_n^k, x^k)$ for neighbors until one is found for which $d(c_n^k, x^k) < d(c_j^k, x^k)$ or until all neighbors have been tested.
3. If no suitable neighbor is found set $i = j$ and terminate, else set $j = n$ and go to 2.

Neighbor descent algorithms have two drawbacks. The first drawback is that the time to find the best-matching codebook entry varies with the input vector. In practice, one limits the number of comparisons to a reasonable number, which results almost always in the determination of the best codebook entry and which guarantees that the codebook entry is at least close in performance to the best codebook entry. A more serious drawback is the requirement for the table storing the indices to all the neighboring cells. The large computational requirement for this table is generally not an issue, since it is done only once. However, the very large storage requirement for the table is often prohibitive and forms the main reason why these methods are not commonly used in practical applications.

8.5.2 The k-Dimensional Tree

Whereas vector quantizers require a large computational effort for a full search, scalar quantizers can be searched rapidly. These facts immediately suggest a simple fast search procedure. In this procedure, space is partitioned with scalar quantizers and codebook vectors are searched within each partition. To be more precise: during the design procedure one partitions $\mathbb{R}^k$ with k scalar quantizers into hypercubes, which we will refer to as **buckets**. Then one creates a table that lists for each bucket the indices for the cells of the vector quantizer that have a nonzero intersection with that bucket. The fast search procedure for the optimal codebook entry c_i^k for a vector x^k then proceeds as follows: *i*) determine the bucket in which x^k falls, *ii*) perform an exhaustive search over all codebook entries whose cells have nonzero intersection with the bucket.

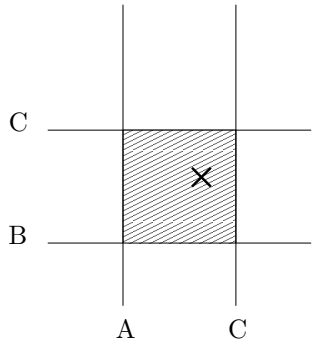

Figure 8.5: The k-d tree of depth 4 for example 8.3.

The k-dimensional tree (e.g., [56]), usually referred to as k-d tree, is created using the same principle of partitioning space as discussed above. The k-d tree method partitions the space one hyper-plane at a time. The position relative the hyper-plane is easy to find: take the inner product of the vector normal to the plane and the difference between the input vector and any vector on the plane and determine its sign. The first step of the k-d tree search method for vector quantization is often a principle-component analysis to make the partitioning by the k-d tree more efficient. Good placement of the hyper-planes is a nontrivial problem [56], but even when the placement of the hyper-planes is not optimized, the method will function.

Example 8.3: A k-d tree of depth four
An example of a k-d tree with depth four is given in figure 8.5 (the input vector is given by "x"). As a first step, we split the space ($\mathbb{R}^2$ in this case) with the hyper-plane A. In this case, the input vector is found to be to the right of hyper-plane A. The next partition is B; the input vector is found to be above this hyper-plane. We perform the same operations with the hyperplanes C and D and thus find that the point is in the cross-hatched bucket that forms a terminal branch of this k-d binary tree of depth four. To exploit this in a vector quantizer covering $\mathbb{R}^2$, one looks up in the table which Voronoi regions intersect with this particular bucket and performs an exhaustive search over the corresponding centroids (codebook entries) to find the nearest entry in the codebook.

8.6 Problems

1. Figure 8.6 displays the beginning of the Lloyd algorithm for a scalar random variable with a simple density and four quantization levels. The top of the figure shows the density, which consists of two regions of identical uniform density. The center of the "X" denotes decoded values, and the vertical bars denote the cell boundaries. The squared error criterion is used.

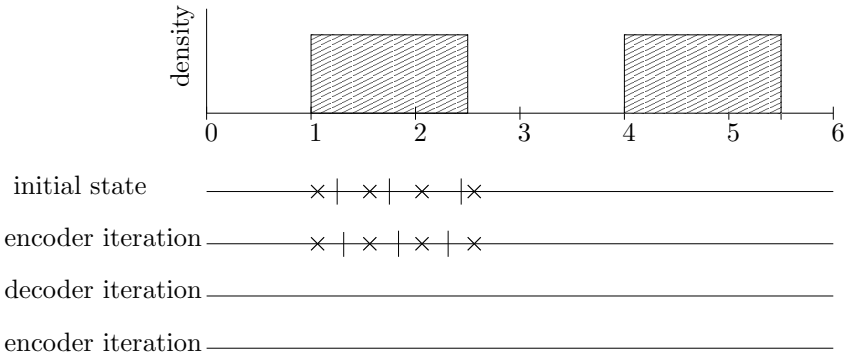

Figure 8.6: Density function and initial Lloyd iterations for problem 1.

   (a) Copy the picture and continue the iterative process in a *qualitative* manner. Continue your iterations until the changes are not visible in your picture.

   (b) Has your solution converged exactly?

   (c) In what sense is the solution optimal?

   (d) How would your answer under 1b change if you were to optimize given a data base and not given a density function?

   (e) Suggest optimal centroids and cell boundaries for the illustrated density distribution. Show that they satisfy the optimality conditions.

2. You are to encode two processes consisting of independent identically distributed samples. The marginal densities are shown in figure 8.7.

(a) Write down the optimality conditions and based on these find the codebooks that are likely the optimal 2-bit (4 codebook entries) scalar constrained-resolution quantizers.

(b) Compute the index entropy for the 2-bit constrained resolution quantizer for your design for quantizer A and quantizer B.

(c) Make a codeword assignment for both cases with a fixed number of bits per sample (trivial).

(d) Make a codeword assignment that minimizes the *average* number of bits per sample. (The codeword assignment is on a per-sample basis.)

(e) Sometimes codewords are assigned to blocks of indices to reduce the average bit rate. Would this help in case A and/or case B?

3. Let X_i be a process consisting of iid samples, each uniformly distributed on $[0, 1]$. Define an N-level uniform quantizer $\mathcal{Q}(\cdot)$ on $[0, 1]$ as

$$\mathcal{Q}(x) = (k + 1/2)/N, \quad k/N \leq x < (k + 1)/N.$$

Thus, $[0, 1]$ is divided into N equal intervals. Application of the quantization to X_i yields a new process $\mathcal{Q}(X_i)$ and the quantization-error process $\epsilon_i = \mathcal{Q}(X_i) - X_i$.

(a) Find the probability density function for ϵ_i.

(b) Find $E(\epsilon_i)$, $E(\epsilon_i^2)$, and $E(\epsilon_i Q(X_i))$.

(c) Find the covariance $E[(\epsilon_j - E[\epsilon_j])(\epsilon_i - E[\epsilon_i])]$ for the process ϵ_i.

(d) Find the spectral density for the process ϵ_i.

4. You are supposed to design a 2-bit scalar quantizer that uses a squared error criterion. Consider the experimental data $\{0, 0, 1, 2, 2, 6, 6, 6, 8, 8, 8\}$. You use as initial condition for your codebook $\{0, 1, 2, 3\}$.

(a) Apply the Lloyd algorithm to train the above codebook with the given data.

(b) In what sense is the solution of the Lloyd algorithm optimal?

(c) Compute the total error for the solution you found above. Is the initial condition a "good" one and why? Can you define a better one?

5. Determine a practical expression for an optimal decoder for the absolute error criterion, $d(x^k, y^k) = \|x^k - y^k\|$.

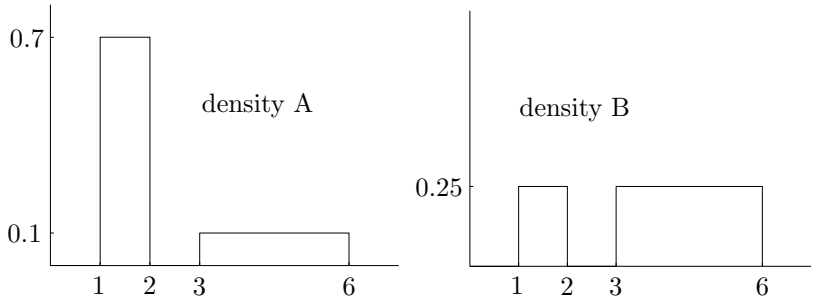

Figure 8.7: Marginal density functions.

6. Consider a scalar Gaussian density with unit variance and zero mean. Write a program to generate 10000 data.

   (a) Compute the Shannon lower bound for a rate of 4 bits per sample.

   (b) Design a 16-entry codebook by drawing 16 points from the density.

   (c) Using the Lloyd algorithm optimize the random 4-bit codebook. Plot the centroids and cell bounds.

   (d) Compare the experimental performance of your random and trained codebooks with the bound on distortion; provide reasons for why you do not reach the bound.

   (e) Under the (optimistic) assumption that high-resolution approximations hold, design an entropy-constrained codebook for a 4-bit entropy.

   (f) Use a practical Huffman code to encode the indices of the entropy-constrained codebook. Compare its experimental performance with the other methods.

7. Consider a two-dimensional density $f_{XY}(x, y) = f_X(x)f_X(y)$ where $f_X(x)$ is Gaussian with unit variance. Write a program to generate 100000 data.

   (a) Use the constrained-resolution Lloyd algorithm with LBG initialization to find a set of 16 centroids. Plot the centroids and measure the distortion.

   (b) Use the constrained-entropy Lloyd algorithm with LBG initialization to plot the empirical rate-distortion function between 0 and 5 bits. Obtain separate curves for codebooks with 16, 32, 64, 128, and 256 entries. Plot the centroids for the 64-entry codebook for an approximately 3-bit rate.

   (c) Use a practical Huffman code to encode the indices of your 4-bit entropy-constrained (size 64) and resolution-constrained codebooks and estimate the resulting true mean bit rate. Comment on their performance. Can you do better?

8. Consider again the density function in figure 7.3. Generate a data base of 10000 points according to this density.

   (a) Starting with a codebook with 4 reconstruction points in each cluster, use the Lloyd algorithm to optimize the codebook. Make a plot showing the centroids.

   (b) Starting with a codebook with all 8 reconstruction points in the top right cluster, use the Lloyd algorithm to optimize the codebook. Make a plot showing your centroids.

   (c) Starting with a codebook with all 8 reconstruction points in the top right cluster, use the Lloyd algorithm with an equidistortion correction to optimize the codebook. Make a plot showing your centroids.

   (d) Use the LBG algorithm to design a codebook with 8 reconstruction points. Make a plot showing the centroids.

   (e) Comment on your results.

9. Consider an iid Gaussian process with unit variance and and a squared-error distortion criterion.

(a) Write a program that generates data according to this process and make a scatter plot containing 10000 data points for two-dimensional vectors.

(b) Using the Lloyd algorithm, design vector quantizers of fixed rate 2 (remember that rate refers to bits per dimension) with dimensions 1, 2, 3, 4, and 5.

(c) Using the entropy-constrained Lloyd algorithm, design vector quantizers of mean rate 2 (remember that rate refers to bits per dimension) with dimensions 1, 2, 3, 4, and 5.

(d) Plot the rate-distortion function for a Gaussian iid process; plot the distortions for your vector quantizer implementations in the rate-distortion plot (evaluate over a test database).

10. Consider a first-order Gauss-Markov process with unit variance and with correlation coefficient 0.9 and a squared-error distortion criterion.

(a) Write a program that generates data according to this process and make a scatter plot containing 10000 data points for two-dimensional vectors.

(b) Using the Lloyd algorithm, design vector quantizers of rate 4 with dimensions 1, 2, 3, 4, and 5.

(c) Evaluate and plot the rate-distortion function for the process.

(d) Plot the distortions for your vector quantizers in the rate-distortion plot (evaluate over a test database).

11. Consider problem 10.

(a) Using the constrained-entropy Lloyd algorithm, design constrained-entropy vector quantizers of rate (close to) 4 with dimensions 1, 2, 3, 4, and 5. Explain your choice of codebook size.

(b) Plot the distortions for your constrained-entropy vector quantizers in the rate distortion plot.

12. Provide several reasons for not using the partitioning obtained by a k-d tree directly as Voronoi regions for a vector quantizer.

13. With a random number generator, we want to generate a noise signal that has the same covariance matrix as a Lloyd algorithm optimized quantizer. The generated noise is of the form $\alpha X + \beta W$, where X is the random vector to be quantized, and W is a random vector. Find the covariance matrix for W, and describe a method to generate W.

9

Transforms and Filter Banks

9.1 Introduction

In chapter 7, we saw that scalar[1] quantization is a computationally simple procedure that can perform within 0.25 bit of the rate-distortion bound under certain conditions, assuming high rate. Most important is the condition that the variables to be quantized are independent. This condition forms a powerful argument to design a coding system that makes the data independent by means of invertible mappings. The forward mapping can then be applied prior to quantization and encoding and the inverse mapping can be applied to the decoded values. Most practical source coders include such mappings. The mappings can roughly be divided into block transforms and filter banks on the one hand, and prediction techniques on the other hand. In this chapter, we discuss the block transforms and filter banks, while prediction is discussed in chapter 10.

Block transforms and filter banks are popular in image coding and audio coding. Their application is less common in speech coding. Examples of coders that use linear block transforms are JPEG for images [57] and MPEG-2 [1] in audio coding.

In the fore-mentioned practical implementations, the transforms are fixed. If the statistics are known, coding performance benefits, in general, from adapting the transform to the statistics of the input signal. Since this adaptation can be done before the actual signal vector is known, we refer to this adaptation as **a-priori adaptation**.

Many recent coding methods adapt the transform to the actual signal samples. We call this **a-posteriori adaptation**. These methods are not subject to the above-mentioned motivation for transforms. Rather it is more natural to consider the transform adaptation to be part of the codebook structure. In an example of a-posteriori adaptation, for a sequence of vectors that is to be encoded, a particular transform can be selected from a dictionary of transforms for each vector in the sequence. In the case of a-posteriori transform adaptation, the encoded information generally consists of information about the transform, and a set of conventional scalar quantization indices. Naturally, a trade-

[1] For the case of entropy-constrained quantization, our reasoning leading to the desirability of independence assumes that the constraint is on the *first-order* entropy. This additional assumption of scalar character is also motivated by the strong demands on computation and storage by lossless coders.

off exists between the information required for the transform specification and for the indices. Because of their structure, it is often difficult to relate the coding efficiency of methods with a-posteriori transform adaptation to results in information theory. Furthermore, the adaptation generally implies that these transforms are **nonlinear** operators[2].

This chapter is organized as follows. For a proper understanding of transforms and filter banks, we first discuss the principles of linear transforms that map $\mathbb{C}^k$ onto $\mathbb{C}^m$, and their interpretation as basis expansions and frame expansions. These principles are then extended to the discrete and continuous Hilbert spaces and the discrete Hilbert space is used to define filter banks. Next, we discuss some common examples of non-adapting transforms (the DFT and the DCT transforms). We then move on to a-priori adaptation, in a section that mostly focuses on the the Karhunen-Loève transform. This section also explains the relation between the DFT, DCT, and the Karhunen-Loève transform. We end the chapter with a section on methods for a-posteriori adaptation of the transform.

9.2 Fundamentals of Linear Transforms

In this section, we discuss linear transforms from a coding perspective. We start with transforms that map a k-dimensional complex vector space $\mathbb{C}^k$ onto itself, then move on to more general transforms that map $\mathbb{C}^k$ onto $\mathbb{C}^m$ with $m \geq k$ (the transformed vector is embedded in an m-dimensional complex vector space), and end with a brief discussion of transforms in discrete Hilbert space.

9.2.1 Transforms that Map $\mathbb{C}^k$ onto Itself and Bases

The coding of random vectors using an invertible linear transform is illustrated in figure 9.1. The forward transform is performed on the realization vector x^k using a $k \times k$ matrix U and the resulting vector y^k is coded and decoded. The resulting quantized vector $\mathcal{Q}(y^k)$ is then converted to $\mathcal{Q}_{U^{-1}}(x^k)$ by the inverse transformation, where the subscript on the quantizer operator indicates the linear transform after quantization.

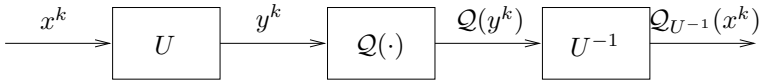

Figure 9.1: A transform coder using an invertible $k \times k$ matrix U.

Square-Matrix Transformations and Bases

In the coding scheme of figure 9.1, the quantizers operate on a vector y^k in $\mathbb{C}^k$ that is obtained by the linear transform

$$y^k = Ux^k, \tag{9.1}$$

[2]Linearity of an operator implies that scaling the input is equivalent to scaling the output and that summing inputs is equivalent to summing outputs. The latter property generally fails in a-posteriori adaptation.

where U is a square, invertible matrix.

Let us compare the matrix multiplication with the definition of a standard inner product. We write the inner product between the k-dimensional column vectors x^k and z^k as

$$\langle x^k, z^k \rangle \equiv \sum_{l=1}^{k} z_l^{k*} x_l^k, \tag{9.2}$$

where * indicates complex conjugate. We see that the inner product has a simple relation to the matrix multiplication:

$$y_n^k = \langle x^k, U_n^H \rangle, \tag{9.3}$$

where ^{H} indicates the conjugate transpose or **Hermitian transpose** and where U_n^H is the n'th column of U^H.

The transform of equation 9.1 (or, alternatively, the inner product of equation 9.3) results in the representation of the vector x^k in a new **basis** in $\mathbb{C}^k$. This can be seen as follows. The inverse transform is of the form

$$x^k = U^{-1} y^k. \tag{9.4}$$

Since U is invertible, the columns of U^{-1} must span $\mathbb{C}^k$. In other words, the columns of U^{-1} must form a basis of $\mathbb{C}^k$. Equation 9.4 can be interpreted as an expansion of x^k into these basis vectors. Each of the basis vectors is multiplied by a component of y^k. This **basis expansion** interpretation of the inverse transform (equation 9.4) can be made more explicit as follows:

$$x^k = \sum_{n=1}^{k} y_n^k U_n^{-1} = \sum_{n=1}^{k} \langle x^k, U_n^H \rangle \; U_n^{-1}, \tag{9.5}$$

where U_n^{-1} is the n'th column of U^{-1}.

Distortion-Invariant Transforms and Orthonormal Bases

In our source-coding context, the purpose of the transform U is to make scalar quantizers or low-dimensional vector quantizers more efficient while using a given distortion criterion. However, the distortion is in general not the same when measured with the same method on x^k and $y^k = Ux^k$. Transforms that do not affect the distortion are particularly convenient to use. In the following, we will determine under what conditions the transform is distortion invariant for the class of distortion criteria that are of the form

$$d(y^k, \mathcal{Q}(y^k)) = d(\|y^k - \mathcal{Q}(y^k)\|), \tag{9.6}$$

where $\|\cdot\|$ indicates the Euclidean norm, based on the standard inner product: $\|y^k\| = \sqrt{y^{kH} y^k}$. This set of criteria includes the distortion criteria given by equation 7.12.

The requirement that the Euclidean norm of the quantization error is invariant with respect to the inverse transformation U^{-1} can be written as

$$\|y^k - \mathcal{Q}(y^k)\| = \|x^k - U^{-1}\mathcal{Q}(y^k)\| = \|U^{-1}(y^k - \mathcal{Q}(y^k))\|, \tag{9.7}$$

which results in the simple requirement that

$$U^{-H}U^{-1} = I. \tag{9.8}$$

Multiplying equation 9.8 by U^H on the left and U on the right, we obtain

$$I = U^H U. \tag{9.9}$$

A matrix U satisfying equation 9.9 preserves the Euclidean norm and is called **unitary**. The unitarity condition implies that the columns of U are orthonormal and form an orthonormal basis:

$$\langle U_n, U_m \rangle = \delta_{nm}, \tag{9.10}$$

where δ_{nm} is the Kronecker delta.

In the present case we are more interested in the columns of U^H rather than the columns of U, since the columns of U^H are used for the inner product of equation 9.3. Fortunately, for a square matrix U, if $UU^H = I$ then also[3] $U^H U = I$ and the columns of U^H also form an orthonormal basis:

$$\langle U_n^H, U_m^H \rangle = \delta_{nm}, \tag{9.11}$$

Equation 9.9 shows that for a unitary transform $U^H = U^{-1}$ and that

$$x^k = U^H y^k. \tag{9.12}$$

The inverse mapping 9.12 can be interpreted as an **orthonormal basis expansion**:

$$x^k = \sum_{n=1}^{k} y_n^k U_n^H = \sum_{n=1}^{k} \langle x^k, U_n^H \rangle \; U_n^H. \tag{9.13}$$

Equation 9.13 is equation 9.5 for the specific case of a unitary matrix U.

Example 9.1: A simple orthonormal basis

Figure 9.2 illustrates a simple basis in $\mathbb{R}^2$. Let us use this as the expansion basis. That is

$$U_1^H = \frac{1}{\sqrt{2}} \begin{bmatrix} -1 \\ 1 \end{bmatrix}$$

and

$$U_2^H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 1 \end{bmatrix}.$$

The transform matrix is thus

$$U = \frac{1}{\sqrt{2}} \begin{bmatrix} -1 & 1 \\ 1 & 1 \end{bmatrix}.$$

It is easily verified that equality 9.9 holds for this matrix.

[3] In general, for a square matrix U the left and the right inverse are the same: if $CU = I$ and $UD = I$, then $C = C(UD) = (CU)D = D$.

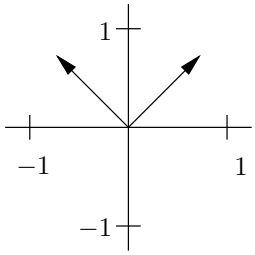

Figure 9.2: A simple orthonormal basis in $\mathbb{R}^2$.

9.2.2 Linear Transforms that Map $\mathbb{C}^k$ onto $\mathbb{C}^m$ and Frames

We now remove the restriction that the matrix U has to be square. Instead of assuming invertibility, we assume that the rows of U span the space of the input vectors and that the distortion criterion is of the form given by equation 9.6. The transform is thus

$$y^m = Ux^k, \tag{9.14}$$

where m does not have to equal k. Again, this transform can be interpreted as computing the inner products of the complex conjugates of the row vectors of U and the vector x^k.

Since we required that the rows of U span the input vector space, we must have $m \geq k$ and we must have that Ux^k is zero only if x^k is zero. The latter result and the linearity of the mapping imply that the mapping from $\mathbb{C}^k$ to $\mathbb{C}^m$ is one-to-one: if both $z^m = Ux^k$ and $y^m = Ux^k$ then also $z^m - y^m = 0$ and thus $z^m = y^m$. It is important to note that the columns of U span a k-dimensional subspace of $\mathbb{C}^m$.

Since the forward mapping is one-to-one, it is invertible, suggesting it may be useful for coding. The corresponding coding system is illustrated in figure 9.3. In practice, there are some problems. First, the output of the quantization operation $\mathcal{Q}(y^m)$ is not constrained to be in the column space of U. Second, the inverse of U, denoted by $\tilde{U}$, is, in general, not unique. In a coding system, it is natural to use the particular inverse transform that minimizes the mean distortion associated with quantization of the output. It is straightforward to find such a transform for the mean squared-error distortion measure and that is what we will do in this section.

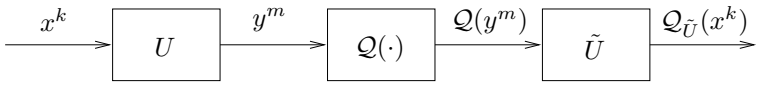

Figure 9.3: A transform coder using an $m \times k$ matrix U.

We introduce a **dimension redundancy** in the transform domain variables if $m > k$. This type of redundancy, which is often simply referred to as "redundancy" should not be confused with the redundancy rate defined in equations 2.34 and 3.10. The dimension redundancy is generally undesirable when scalar quantization is used, since we must quantize more variables than we started with. However, dimension-redundant representations are useful in certain cases. For example, this type of redundancy can be used to counter the loss of information packets in a packet network. Also, in the more general setting of Hilbert space, it can be shown that a signal expansion in terms

of smoothly windowed complex exponentials is possible only if the representation is redundant (cf. section 9.3.3).

Projections, the Pseudo-Inverse, and Quantization

In the mapping of equation 9.14, y^m is a weighted addition of the k columns of the matrix U. That is, Ux^k must be within a k-dimensional subspace of $\mathbb{C}^m$ spanned by the columns of U. We call this subspace the **column space of** U or the **image of** U. We already know that any point Ux^k in the image of U has a one-to-one correspondence with a point in C^k. This means that there is an inverse $\tilde{U}$ such that for any $x^k \in \mathbb{C}^k$

$$x^k = \tilde{U}Ux^k. \tag{9.15}$$

It is convenient to separate $\mathbb{C}^m$ into the image-of-U subspace and its orthogonal complement. An arbitrary vector y^m in $\mathbb{C}^m$ can thus be decomposed into a component in the image of U, denoted as y^m_U and a component orthogonal to that, which is denoted as $y^m_{U\perp}$,

$$y^m = y^m_U + y^m_{U\perp}. \tag{9.16}$$

The inverse mapping for points in the image of U is given: it is the inverse of the forward mapping U. However, if we require only that $\tilde{U}$ is an inverse, then the linear mapping for the orthogonal complement of the image of U is arbitrary. Let us consider the particular inverse $U^\sharp$ for which $U^\sharp y^m_{U\perp} = 0$. We then have for an arbitrary y^m in $\mathbb{C}^m$ that

$$\begin{aligned} UU^\sharp y^m &= UU^\sharp y^m_U + UU^\sharp y^m_{U\perp} \\ &= UU^\sharp y^m_U \\ &= y^m_U. \end{aligned} \tag{9.17}$$

Thus, $UU^\sharp$ is the **projection** of y^m onto the column space of U. Similarly, it is seen that $I - UU^\sharp$ is the projection of y^m onto the complement of the column space of U,

$$\begin{aligned} (I - UU^\sharp)y^m &= y^m - y^m_U \\ &= y^m_{U\perp}. \end{aligned} \tag{9.18}$$

Given the significance of the inverse $U^\sharp$ such that $U^\sharp y^m_{U\perp} = 0$, it is useful to find a convenient expression for this inverse. We know that there exists a unique x^k such that $y^m_U = Ux^k$ and thus that $U^H y^m_U = U^H Ux^k$. By definition, the columns of U are orthogonal to the orthogonal complement of U, that is $U^H y^m_{U\perp} = 0$. We thus have the following conditions:

$$U^H y^m_U = U^H Ux^k, \tag{9.19}$$

$$U^H y^m_{U\perp} = 0. \tag{9.20}$$

Summing these conditions gives us

$$U^H y^m = U^H Ux^k. \tag{9.21}$$

The inverse we are looking for, the so-called **pseudo-inverse**, is then

$$U^\sharp = (U^H U)^{-1} U^H. \tag{9.22}$$

$U^\sharp$ indeed satisfies both $U^\sharp Ux^k = x^k$ and $U^\sharp y^m_{U\perp} = 0$.

Example 9.2: Finding the pseudo-inverse through minimization
The pseudo-inverse is also found from a minimization criterion. To simplify the derivation, we assume that x^k and the transform U are real, but this restriction is not necessary. Consider the vector y^m that is

$$y^m = Ux^k + \epsilon^m,$$

where ϵ^m is a remainder vector. We define the pseudo-inverse as the transform mapping $\mathbb{R}^m$ onto $\mathbb{R}^k$ that minimizes the Euclidean norm of ϵ^m for a given y^m. This inverse mapping is defined if we express the optimal x^k in terms of y^m. The square of the norm is

$$\epsilon^{mH}\epsilon^m = y^{mH}y^m - 2x^{kH}U^H y^m + x^{kH}U^H U x^k.$$

Differentiating with respect to the vector x^k and setting the result to zero results in

$$0 \quad = \quad -2U^H y^m + 2U^H U x^k,$$

which means that

$$x^k = (U^H U)^{-1} U^H y^m.$$

We can now turn to the task of finding the inverse transform that minimizes, in general, the mean squared-error $\mathrm{E}[\|X^k - \mathcal{Q}_{\tilde{U}}(X^k)\|^2]$ where quantization is performed on the random vector $Y^m = UX^k$ with a squared-error criterion. We want an inverse $\tilde{U}$ such that

$$\mathrm{E}[\|X^k - \mathcal{Q}_{\tilde{U}}(X^k)\|^2] \quad = \quad \mathrm{E}[\|\tilde{U}(Y^m - \mathcal{Q}(Y^m))\|^2] \tag{9.23}$$

is minimized, where $\|\cdot\|$ is the Euclidean norm. We decompose the error into the orthogonal components $Y^m - \mathcal{Q}(Y^m)_U$ and $\mathcal{Q}(Y^m)_{U\perp}$ (recall that since $Y^m = UX^k$ we have that $Y^m = Y^m_U$). We assume that the covariance of the quantization error $y^m - \mathcal{Q}(y^m)$ is a scaled identity matrix (that is, we assume an "ideal" quantizer; it was shown in section 7.6.2 that optimal lattice quantizers satisfy this condition). We can then write equation 9.23 in the following form:

$$\mathrm{E}[\|X^k - \mathcal{Q}_{\tilde{U}}(X^k)\|^2] \quad = \quad \mathrm{E}[\|\tilde{U}(Y^m - \mathcal{Q}(Y^m)_U)\|^2] + \mathrm{E}[\|\tilde{U}\mathcal{Q}(Y^m)_{U\perp}\|^2]. \tag{9.24}$$

The first term on the right-hand side of equation 9.24 is fixed; the second is minimized if $\tilde{U}\mathcal{Q}(y^m)_{U\perp} = 0$, which is true for $\tilde{U} = U^\sharp$. In other words, the pseudo-inverse is optimal in a mean squared-error sense for the transform, quantization, inverse-transform operation sequence shown in figure 9.3.

Frames in $\mathbb{C}^k$

Let us consider again an $m \times k$ matrix U, whose rows span $\mathbb{C}^k$. If these row vectors have finite norm, as they normally do, then they form a frame. The row vectors of U, or, equivalently, the column vectors of U^H form a **frame**, if the so-called **frame condition** holds:

$$Ax^{kH}x^k \leq x^{kH}U^H U x^k \leq Bx^{kH}x^k, \quad \forall_{x^k \in \mathbb{C}^k}, \tag{9.25}$$

where $A > 0$ and $B < \infty$ are scalars[4]. A and B are called the **frame bounds**. The condition that $A > 0$ implies that the frame vectors (the columns of U^H) must span $\mathbb{C}^k$. In $\mathbb{C}^k$, $B < \infty$ states that the vector norms of the frame vectors are finite. Whenever $A = B$, then the frame is called **tight**.

It is useful to relate the inequality 9.25 to the eigenvalues of the $k \times k$ matrix $U^H U$. We start with noting that the $k \times k$ matrix $U^H U$ is positive definite and that $U^\sharp U^{\sharp H}$ is its inverse. We can then write $U^H U = V^H \Lambda V$, where Λ is a diagonal matrix with the eigenvalues along the diagonal and V is unitary. It follows that if the frame bounds are tight (not to be confused with the frame itself being tight) then the frame bounds are simply the lowest and the highest eigenvalues of $U^H U$.

It is now easily proven that, if the column vectors of U^H form a frame with frame bounds A and B, then the column vectors of $U^\sharp$ form a frame with frame bounds $1/B$ and $1/A$. This follows immediately from the relation of the frame bounds with the eigenvalues of $U^H U$:

$$\begin{aligned} U^\sharp U^{\sharp H} &= (U^H U)^{-H} \\ &= (V^H \Lambda V)^{-H} \\ &= V^H \Lambda^{-H} V. \end{aligned} \tag{9.26}$$

Thus, the eigenvalues of $U^\sharp U^{\sharp H}$ fall between $1/B$ and $1/A$. This implies that $1/B$ and $1/A$ form frame bounds for the rows of $U^{\sharp H}$ (and the columns of $U^{\sharp *}$ and $U^\sharp$).

We have now shown that the columns of $U^\sharp$ form a frame with frame bounds $1/B$ and $1/A$. This frame is called the **dual frame** of the frame associated with the columns of U^H. We sometimes use the nomenclature **analysis frame** for the columns of U^H and **synthesis frame** for the columns of $U^\sharp$. The names analysis and synthesis frames are related to their function. The analysis frame is used to analyze the input x^k, by means of inner products. This is obvious when we rewrite equation 9.14 as

$$y_n^m = \langle x^k, U_n^H \rangle. \tag{9.27}$$

We use the synthesis frame to resynthesize the original vector (in a coding system: a quantized version there-of). This becomes clear when we write the inverse transform in the form of a **frame expansion**,

$$x^k = U^\sharp y^m = \sum_{l=1}^{m} y_l^m U_l^\sharp = \sum_{l=1}^{m} \langle x^k, U_l^H \rangle U_l^\sharp, \tag{9.28}$$

where the $U_l^\sharp$ are the columns of $U^\sharp$. Note again, that the frame expansion is equivalent to the inverse mapping from the frame coefficients $\langle x^k, U_l^H \rangle$ to the vector x^k.

The operation Ux^k is sometimes referred to as a frame operation. U can then be interpreted as the frame operator associated with the frame formed by the columns of U^H.

Example 9.3: The duality of the dual frame
When using $U^{\sharp H}$ for the analysis, we have

$$y_n^m = \langle x^k, U_n^\sharp \rangle.$$

[4]Our usage of the notation A and B for the frame bounds is motivated by the near-universal usage of this notation in the literature.

Furthermore, the generalized inverse of $U^{\sharp H}$ is

$$\begin{aligned}(U^{\sharp H})^{\sharp} &= (U^{\sharp}U^{\sharp H})^{-1}U^{\sharp} \\ &= ((U^H U)^{-1}U^H U(U^H U)^{-1})^{-1}(U^H U)^{-1}U^H \\ &= U^H.\end{aligned}$$

The corresponding synthesis is thus

$$x^k = \sum_{l=1}^{m} \langle x^k, U_l^{\sharp} \rangle U_l^H.$$

Contrasting these equations with equations 9.27 and 9.28 illustrates the duality between the frames formed by the columns of U^H and $U^{\sharp}$, respectively.

From the frame condition it follows immediately that for a tight analysis frame with frame bound A we have

$$U^H U = AI. \tag{9.29}$$

The corresponding synthesis frame has a frame bound of $1/A$. Furthermore, in the case of a tight frame, the pseudo-inverse reduces to the following simple form:

$$\begin{aligned}U^{\sharp} &= (U^H U)^{-1}U^H \\ &= \frac{1}{A}U^H.\end{aligned} \tag{9.30}$$

Equation 9.30 implies that **a tight frame is its own dual, except for a constant**. In other words, for a tight frame the analysis and synthesis frames are identical except for a constant.

In closing the discussion on frames in $\mathbb{C}^k$, we note that the relation between the frame bounds of a frame and its dual do not carry over to the columns of an arbitrary generalized inverse $\tilde{U}$. For finite dimensionality and inverses with finite coefficients, the columns of $\tilde{U}$ do form a frame, however.

Example 9.4: An orthonormal basis is a tight frame with $A = 1$
Let U be a $k \times k$ matrix with orthonormal row vectors. In other words, the row vectors form an orthonormal basis:

$$UU^H = I.$$

It immediately follows that

$$U^H U = I$$

and, therefore, that

$$x^{kH}U^H U x^k = x^{kH}x^k$$

for any x^k. Comparing this to the frame conditions (equation 9.25), we see that an orthonormal basis corresponds to a frame with frame bound $A = B = 1$.

Example 9.5: A simple tight frame
Figure 9.4 illustrates a simple frame in $\mathbb{R}^2$. The frame forms the rows of U:

$$U_1^H = \begin{bmatrix} \frac{\sqrt{3}}{2} \\ \frac{1}{2} \end{bmatrix},$$

$$U_2^H = \begin{bmatrix} -\frac{\sqrt{3}}{2} \\ \frac{1}{2} \end{bmatrix},$$

and

$$U_3^H = \begin{bmatrix} 0 \\ -1 \end{bmatrix}.$$

We note that U^H is a 2×3 matrix,

$$U^H = \begin{bmatrix} U_1^H \, U_2^H \, U_3^H \end{bmatrix}$$

and that

$$U^H U = \frac{3}{2} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}.$$

The frame bound is thus $\frac{3}{2}$ and $U^\sharp = \frac{2}{3} U^H$ is an inverse of U.

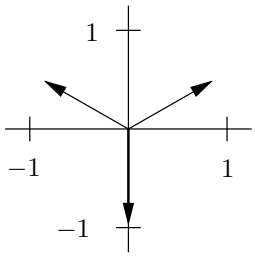

Figure 9.4: A simple tight frame in $\mathbb{R}^2$.

Example 9.6: Construction of a tight frame from two orthonormal bases
Consider a $k \times k$ transform matrix D and a $k \times k$ transform matrix E, such that $D^H D = I$ and $E^H E = I$. Thus, the rows of D and the rows of E each form an orthonormal basis of $\mathbb{C}^k$. We can construct a frame by joining the bases corresponding to D and E. The new frame consists of the rows of F in

$$F = \begin{bmatrix} D \\ E \end{bmatrix}.$$

We then have that

$$F^H F = D^H D + E^H E = I + I = 2I.$$

Thus, the rows of F form a tight frame with frame bound 2. We can construct frames with higher redundancy by using more orthonormal bases.

Example 9.7: Multiple-description coding
In packet networks, the bit stream is transmitted in the form of discrete packets, each containing a header and a payload (the actual bit stream). If the network operates under real-time constraints and at high loads, packets are often lost. An obvious method to increase robustness is to transmit all packets twice. It is natural to throw away one packet if both packets arrive. Using frames we can do better than that.

Let each packet contain k coded samples of the signal, with each packet repeated twice. We can write this in a convenient manner using frame notation. Let x^k

represent the data of one packet. The data of the two packets then form a frame $y^{2k} = Ux^k$ with

$$U = \begin{bmatrix} I \\ I \end{bmatrix}.$$

The first k components of $\mathcal{Q}(y^{2k})$ are encoded in the first packet, and the second k components in the second packet.

We use three decoders. If we receive both packets we use the **central** decoder, if we receive one packet we use the appropriate **side** decoder. The first side decoder is activated if we receive only the first packet (the first k samples of $\mathcal{Q}(y^{2k})$) and it implements $\mathcal{Q}_{U^{-1}}(x_i) = \mathcal{Q}(y_i)$ for $i = 1, \cdots, k$. Similarly, second side decoder implements $\mathcal{Q}_{U^{-1}}(x_i) = \mathcal{Q}(y_{i+k})$ for $i = 1, \cdots, k$.

Next, we consider the central decoder. We note that the pseudo-inverse for U is in this case

$$U^\sharp = \frac{1}{2}U^H.$$

We set up the encoding so that the quantization errors of the components of y^{2k} are independent. This can be accomplished by adding a known noise vector to the input vector to the quantizer, y^{2k}, and subtracting the same noise vector after decoding, a process called "dithering". Let the quantization error be $v^{2k} = y^{2k} - \mathcal{Q}(y^{2k})$. The mean squared quantization , for the case that both descriptions are received, is

$$\mathrm{E}[\|X^k - \mathcal{Q}(X^k)\|^2] = \frac{1}{4}\mathrm{E}[\|V^{2k}\|^2] = \frac{1}{2}\mathrm{E}[\|V^k\|^2],$$

where V^k contains only the first k components of V^{2k}. Thus, by quantizing and encoding the same samples twice with independent quantization error, we reduced the mean squared error by a factor two (a gain of about 3 dB) for the case that both descriptions are received.

In the above case, we assumed that the quantizers generate random quantization noise and exploited that for the multiple-description encoding. It is more advantageous to exploit the deterministic nature of a quantizer. Let us consider the case of high-rate entropy-constrained scalar quantization. We have seen in chapter 7 that the optimal quantizer for this case is uniform. It is natural to interleave the quantization levels of the first and second packets. That is, the centroids for the second quantizer are set midway between the centroids of the first quantizer. It is easily shown that also for this case $U^\sharp = \frac{1}{2}U^H$ is optimal. The interleaving of the two scalar quantizers is equivalent to doubling the number of levels and, thus, to adding one bit to the encoding. This implies that, if both packets are received, then this method reduces the mean squared quantization error for decoding two packets by a factor four (6 dB) over the simple method of transmitting the same packet twice.

Norm-Preserving Linear Transforms that Map $\mathbb{C}^k$ onto $\mathbb{C}^m$ Correspond to Tight Frames

Let us again consider figure 9.3. Ideally, the distortion criterion minimized during the quantization in $\mathbb{C}^m$ should equal to the distortion in $\mathbb{C}^k$, even when $m > k$. For the squared-error criterion, this corresponds to preservation of the Euclidean norm under the inverse transform, which is impossible. Indeed, equality of $y^{mH}y^m$ and $y^{mH}\tilde{U}^H\tilde{U}y^m$

leads to the condition

$$\tilde{U}^H \tilde{U} = I, \tag{9.31}$$

which can, in general, not be satisfied since this matrix equation has m^2 components, while there are only $k \times m$ components in the matrix U.

It is, however, possible to preserve the Euclidean norm when mapping from $\mathbb{C}^k$ to $\mathbb{C}^m$ for $k \leq m$. If we allow a scaling A, then we obtain the condition

$$U^H U = A\, I, \tag{9.32}$$

which is the condition that the rows of U form a tight frame.

Equation 9.32 implies for the squared-error distortion that

$$\begin{aligned} A\|x^k - Q_{U^H}(x^k)\|^2 &= \|Ux^k - UQ_{U^H}(x^k)\|^2 \\ &= \|y^m - \mathcal{Q}(y^m)_U\|^2. \end{aligned} \tag{9.33}$$

That is, for a tight frame, the squared-error component that falls within the image of U is preserved. If the rows of U form a tight frame, and if the performance of the quantizer is uniform over all dimensions of the m dimensional space, then minimizing the mean distortion in the m-dimensional space is equivalent to minimizing the mean distortion in the k-dimensional space.

9.2.3 Frames in Hilbert Space and Filter Banks

So-far the most general transforms we have discussed are those where the original vector is in $\mathbb{C}^k$ and the transformed vector is in $\mathbb{C}^m$ ($m \geq k$). We interpreted these transforms in terms of analysis and synthesis frames. The resulting theory is useful in the coding of vectors. We now generalize our discussion so that it becomes applicable to the coding of processes (signals). For processes, we want to use filter banks in place of the matrix transforms we used for the coding of random vectors. Indeed, filter banks are commonly used to enhance the efficiency of source coding. To allow this transition to processes and filter banks, we generalize frame theory to the discrete Hilbert space $\ell^2(\mathcal{Z})$. Because of its similarity, we also define frame theory for the continuous Hilbert space, $\mathcal{L}^2(\mathbb{R})$. We first discuss our previous results in the Hilbert-space setting without generalizing the corresponding derivations and then discuss the interpretation of our results in terms of filter banks.

Frames in Hilbert Space

We start with introducing a notation that is commonly used when defining frames in the Hilbert space setting. In $\ell^2(\mathcal{Z})$, we deal with vectors of infinite dimensionality and write these as discrete functions of the index[5]. For example, $\psi_k(n)$ is the value of function ψ_k at sample n. Alternatively, in certain cases it is more convenient to say that the component n of the vector ψ_k is $\psi_k(n)$. We define the inner product of the discrete-time

[5]To avoid a confusing notation, we do not write the discrete time indices as subscripts.

functions ψ_l and f as

$$\langle f, \psi_l \rangle = \sum_{n=-\infty}^{\infty} \psi_l^*(n) f(n). \tag{9.34}$$

In the case of $\mathcal{L}^2(\mathbb{R})$, the functions are continuous and the inner product is defined by an integral:

$$\langle f, \psi_m \rangle = \int_{\mathbb{R}} \psi_m^*(t) f(t) dt. \tag{9.35}$$

In $\ell^2(\mathcal{Z})$, a frame consists of a set of functions, denoted as $\{\psi_m\}$ (where m is a countable function index with finite or infinite range) for which the frame condition holds:

$$A\langle f, f \rangle \leq \sum_{m \in \mathcal{M}} |\langle f, \psi_m \rangle|^2 \leq B\langle f, f \rangle, \quad \forall_{f \in \ell^2(\mathcal{Z})}. \tag{9.36}$$

The same relation holds for $\mathcal{L}^2(\mathbb{R})$. As before, if $A = B$, then the frame is called tight.

As in the finite-dimensionality case, the forward transform is implemented by inner products with the analysis frame. This forms the **frame operation** in the Hilbert-space setting:

$$\mathcal{U}f(m) = \langle f, \psi_m \rangle, \tag{9.37}$$

where $\mathcal{U}$ is the **analysis frame operator**. The frame condition forms a necessary and sufficient condition for the frame operator to be invertible (on the image of $\mathcal{U}$) and have a bounded (i.e., finite) inverse. The frame condition thus guarantees **completeness** of the frame.

We also define a conjugate frame operator, $\mathcal{U}^H$:

$$\mathcal{U}^H g(n) \quad = \textstyle\sum_m g(m)\, \psi_m^*(n). \tag{9.38}$$

The conjugate frame operator will be used in section 9.2.4.

The notation $\mathcal{U}f(m)$ indicates the inner product of f with ψ_m. These inner products are often referred to as **coefficients**, since they form the coefficients that multiply the dual frame functions in the frame expansion of the original function (signal),

$$\begin{aligned} f(n) &= \sum_{m \in \mathcal{M}} \langle f, \psi_m \rangle\, \psi_m^\sharp(n) \\ &= \sum_{m \in \mathcal{M}} \mathcal{U}f(m)\, \psi_m^\sharp(n) \\ &= \mathcal{U}^\sharp \mathcal{U} f(n). \end{aligned} \tag{9.39}$$

The inverse frame operator $U^\sharp$ results in the frame expansion that corresponds to the inverse mapping from the frame coefficients $\langle f, \psi_l \rangle$ to the function $f(n)$. Its existence is guaranteed by the frame condition.

For the tight frame case, $A = B$, the frame is its own dual except for a factor equal to the frame bound:

$$\psi_m^\sharp = \frac{1}{A}\, \psi_m. \tag{9.40}$$

Before continuing, let us summarize the correspondences between the finite-dimensionality and more general Hilbert-space cases. The set of all analysis-frame functions ψ_m corresponds to the set of row vectors of the finite-dimensionality transform matrix U. The set of all synthesis frame functions $\psi_m^\sharp$ corresponds to the columns of the pseudo-inverse $U^\sharp$.

While the notation introduced for this section is generally used in the continuous and discrete Hilbert space cases, the matrix notation we used before is often helpful for understanding the discrete Hilbert space $\ell^2(\mathcal{Z})$ case. Naturally, the matrices are then of infinite dimensionality, and we have to be careful with the consequences there-of. Practical cases where we use such matrices will be discussed in section 9.3.4.

Perfect-Reconstruction Filter Banks

It is convenient in source coding if the signal can be processed with a finite delay. A natural method to accomplish this is to use frame functions that are constructed from a finite set of finite-support frame functions by repeating a standard time-shift operation. That is, the frame functions can be described by a doubly indexed set of functions $\phi_{j,m}$ such that

$$\phi_{j,m}(n) = \phi_j(n - mM), \;\; j = 1, \cdots, J, \;\; m \in \mathcal{Z}, \tag{9.41}$$

where the time shift is set to M samples and where ϕ_j has finite support (is nonzero for only a finite set of samples) to ensure the finite-delay property. The index m indicates time, while the index j is often associated with frequency, scale, or a within-block time location. This type of doubly indexed frame is sometimes called a **uniform filter bank (UFB) frame** [58], a convention we use here as well. The term "uniform" refers to the uniform time interval M when increasing the time index m in $\phi_{j,m}(n)$.

The structure given in equation 9.41 is particularly useful since the dual of this frame has the same structure. That is, if an analysis frame is a UFB frame, then so is the synthesis frame. Let S_M be a time-shift operator such that $S_M f(n) = f(n + M)$. We then have

$$\begin{aligned} f(n+M) &= \sum_{j,m} \langle S_M f, \phi_{j,m} \rangle \, \phi_{j,m}^\sharp \\ &= \sum_{j,m} \langle S_M f, \phi_{j,m+1} \rangle \, \phi_{j,m+1}^\sharp \\ &= \sum_{j,m} \langle S_M f, S_M \phi_{j,m} \rangle \, \phi_{j,m+1}^\sharp \\ &= \sum_{j,m} \langle f, \phi_{j,m} \rangle \, \phi_{j,m+1}^\sharp . \end{aligned} \tag{9.42}$$

Thus, to shift the signal by M samples, we simply move the coefficients by one time index with respect to the synthesis frame functions. For this to be true for all $f(n) \in \ell^2(\mathcal{Z})$, we must have that $\phi_{j,m+1}^\sharp = S_M \phi_{j,m}^\sharp$.

While the naming convention of the UFB frame clearly connects it to a filter bank, we have not yet discussed this relation explicitly. Let us denote a time-reversal by $\breve{x}(n) = x(-n)$. Let us define $g_j(m)$ as the convolution of $f(n)$ with $\breve{\phi}_j^*$, down-sampled

by a factor M. Then, the following relation holds between this down-sampled filter output and the inner products between $f(n)$ and $\breve{\phi}_{j,m}(n)$:

$$\begin{aligned} g_j(m) &= \sum_n f(mM-n)\,\breve{\phi}_j^*(n) \\ &= \sum_n f(mM-n)\,\phi_j^*(-n) \\ &= \sum_n f(mM+n)\,\phi_j^*(n) \\ &= \sum_n f(mM+n)\,\phi_{j,m}^*(n+mM) \\ &= \langle f, \phi_{j,m}\rangle. \end{aligned} \tag{9.43}$$

Thus, we conclude that the down-sampled convolution of $f(n)$ with a filter with impulse response $\breve{\phi}_j^*$ is equivalent to taking the inner product with the time-reversed frame functions. In other words, computing the coefficients of the frame expansion is equivalent to filtering with a filter bank and then down-sampling by a factor M, as is illustrated in figure 9.5. To allow reconstruction, we must have that $J \geq M$. If $M = J$, then M samples of $f(n)$ correspond to M coefficients $g_j(m)$ and the filter bank is **critically sampled**. If $J > M$, then the filter bank is **over-sampled**.

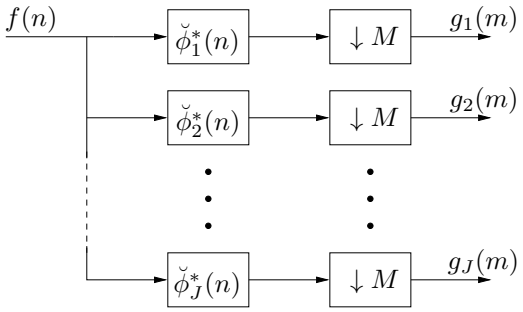

Figure 9.5: Analysis filter bank equivalent of the analysis-frame operator for a UFB analysis frame. For $J > M$ the filter bank is over-sampled and for $J = M$ the filter bank is critically sampled.

Example 9.8: A simple analysis filter bank
The above discussion shows that performing the same linear transform on sequential blocks of a signal can be interpreted as a filter bank. As an example, we consider a signal $s(n)$. The signal is divided into sequential blocks of M samples, thus creating a sequence of M-dimensional vectors; each vector is transformed by multiplying with an $M \times M$ matrix A. Following the above reasoning, these operations are equivalent to a critically-sampled filter bank of the form shown in figure 9.5, with filters $\breve{\phi}_j^*(n),\ j \in \{1, \cdots, M\}$, with discrete filter impulse responses

$$\breve{\phi}_j^*(n) = \begin{cases} A_{j,M-n}, & n = 0, \cdots, M-1 \\ 0, & \text{elsewhere.} \end{cases}$$

The (synthesis) frame expansion can also be interpreted as a filter bank. This is seen as follows. Let $g_j^{\uparrow M}(m)$ be the factor-M upsampling of the signal $g_j(m)$ (insert $M-1$

zeros between each sample). Then we have that

$$\begin{aligned} f(n) &= \sum_{j,m} g_j(m)\, \phi^\sharp_{j,m}(n) \\ &= \sum_{j,m} g_j(m)\, \phi^\sharp_j(n - mM) \\ &= \sum_{j,m} g_j^{\uparrow M}(m)\, \phi^\sharp_j(n - m) \\ &= \sum_{j,m} g_j^{\uparrow M}(n - m)\, \phi^\sharp_j(m). \end{aligned} \tag{9.44}$$

The expansion of $f(n)$ in the synthesis frame functions $\phi^\sharp_{j,m}$ is equivalent to filtering $g_j^{\uparrow M}(m)$ with $\phi^\sharp_j(n)$ and summing over the filter outputs, as is shown in figure 9.6.

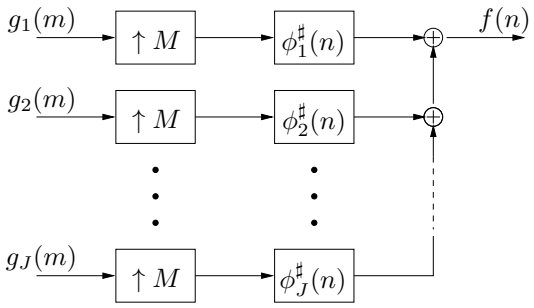

Figure 9.6: Synthesis filter bank equivalent of the synthesis-frame expansion for a UFB analysis frame.

We have seen that UFB frames correspond to perfect-reconstruction filter banks. Critically-sampled filter banks correspond to analysis and synthesis frames that form a basis of $\ell^2(\mathcal{Z})$, whereas over-sampled filter banks correspond to redundant analysis and synthesis frames. Furthermore, it follows from these correspondences that if the UFB frames are tight, then the impulse responses of the analysis and synthesis filter banks are identical except for conjugation and a scaling factor. If the frame is an orthonormal basis, then the scaling factor is unity.

So-far we have ignored several issues. First of all, in a practical implementation, filters must be causal and this means that the analysis-synthesis process introduces delay. If the analysis and synthesis frame functions have finite support, it is straightforward to add delay to the analysis and synthesis filter banks such that each is causal. Perfect reconstruction is maintained, except for this delay. A more serious issue is that we have not discussed how to create an analysis-synthesis frame pair. In finite dimensionalities, the dual frame can be obtained using the pseudo-inverse of the corresponding matrix representation. This procedure usually is not practical for the $\ell^2(\mathcal{Z})$ case. The so-called frame algorithm can often be applied and we will discuss this algorithm in section 9.2.4. However, many common analysis-synthesis frame pairs have been found with procedures specific to the particular frame. In section 9.3, a number of such well-known pairs will be discussed. Below, we provide examples with a simple ad-hoc design method that exploits the properties of tight frames and an implementation there-of.

The performance of a filter bank for a particular application is determined by the behavior of the individual signals $g_i(m)$. The performance is often commonly analyzed in terms of the severity of the well-known **aliasing** (frequency folding) phenomenon for these signals. Aliasing is a function of frequency-band selectivity and the down-sampling factor M and will be discussed in more detail in section 9.3.3.

Example 9.9: Simple perfect-reconstruction filter-bank design method Consider a set of orthonormal functions $\{\phi_j(n)\}_{j\in\{1,\cdots,J\}}$ that have support in the range (are nonzero only for) $0,\cdots,J-1$. That is, the vectors $\{[\phi_j(0),\cdots,\phi_j(J-1)]\}_{j\in\{1,\cdots,J\}}$ are orthonormal. The design method starts with selecting an appropriate set of such orthonormal vectors. Upon selecting such an orthonormal set, we can construct an orthonormal basis $\mathcal{A}_0$ for the Hilbert space $\ell^2(\mathcal{Z})$ by displacing the functions by integer multiples of J: $\mathcal{A}_0 = \{\phi_j(n-lJ)\}_{j\in\{0,\cdots,J\},l\in\mathcal{Z}}$. Computing the coefficients for the orthonormal basis $\mathcal{A}_0$ corresponds to a block-by-block transform of the signal. We note that $\mathcal{A}_1 = \{\phi_j(n-lJ-1)\}_{j\in\{1,\cdots,J\},l\in\mathcal{Z}}$ also forms an orthonormal basis for $\ell^2(\mathcal{Z})$. From the reasoning used in example 9.6, we know that the joint set $\mathcal{A}_0+\mathcal{A}_1$ forms a tight frame. It is easy to generate many different tight frames in this manner. For example, the set $\mathcal{A}_0+\cdots+\mathcal{A}_{J-1}$ corresponds to the analysis filter bank (of a perfect-reconstruction filter bank) where the filter outputs are not down-sampled. The corresponding synthesis filter banks are constructed by conjugation and division by the oversampling factor. It is simple to construct perfect-reconstruction filter banks with any desired down-sampling rate. The performance of the perfect-reconstruction filter banks constructed this way depends, naturally, on the selection of the orthonormal set of vectors $\{[\phi_j(0),\cdots,\phi_j(J-1)]\}_{j\in\{1,\cdots,J\}}$. An implementation of the design method is illustrated in example 9.10.

Example 9.10: Simple three-band perfect-reconstruction filter bank Following the method outlined in example 9.9, let us design a simple filter bank with three bands, where the output from each filter retains the original sampling rate. We start with constructing a vector $\{[\phi_j(0),\cdots,\phi_j(J-1)]\}_{j\in\{1,\cdots,J\}}$ for each band. We need three vectors, one for each band, and so $J=3$. The low-pass and high-pass filters are centered at $\omega=0$ and at $\omega=\pi$, respectively. Their center frequency differs by π, and it is reasonable to make them differ only by this frequency displacement, which corresponds to a modulation with $e^{j\pi n}=(-1)^n$. In other words, the sign of the vectors corresponding to the high-pass and low-pass filters should be the same for odd components of the vectors and be the opposite for even components of the vectors, or vice versa. The power spectrum of the vector corresponding to the low-pass filter should be maximum at $\omega=0$, which is true if we select the components to be symmetric, positive, and with the largest value in the center. We then arrive at the two orthonormal vectors $[1/2,\sqrt{1/2},1/2]$ and $[1/2,-\sqrt{1/2},1/2]$. The third vector must be orthonormal to both and has to be $[\sqrt{1/2},0,-\sqrt{1/2}]$. The filter bank impulse responses are obtained by simply time-reversing the vectors

$$\begin{aligned} H_1(z) &= \frac{1}{2}+\frac{1}{2}\sqrt{2}z^{-1}+\frac{1}{2}z^{-2} \\ H_2(z) &= -\frac{1}{2}\sqrt{2}+\frac{1}{2}\sqrt{2}z^{-2} \\ H_3(z) &= \frac{1}{2}-\frac{1}{2}\sqrt{2}z^{-1}+\frac{1}{2}z^{-2}. \end{aligned}$$

The transfer functions of these three filters are shown in figure 9.7. Resynthesis from the signals that form the output of the analysis filter bank is obtained by filtering the three signals with the filters

$$\begin{aligned} H_1^\sharp(z) &= \frac{1}{3}(\frac{1}{2} + \frac{1}{2}\sqrt{2}z^{-1} + \frac{1}{2}z^{-2}) \\ H_2^\sharp(z) &= \frac{1}{3}(\frac{1}{2}\sqrt{2} - \frac{1}{2}\sqrt{2}z^{-2}) \\ H_3^\sharp(z) &= \frac{1}{3}(\frac{1}{2} - \frac{1}{2}\sqrt{2}z^{-1} + \frac{1}{2}z^{-2}). \end{aligned}$$

and adding. The factor 1/3, which corrects the signal power, corresponds to the inverse of the frame bound.

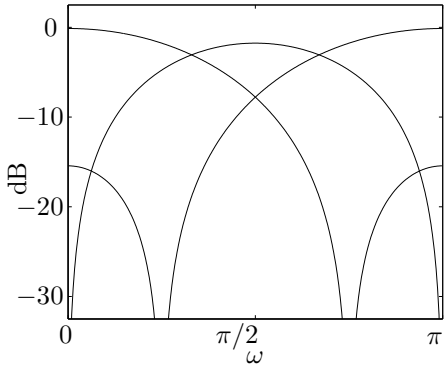

Figure 9.7: The transfer functions for the perfect-reconstruction filter bank of example 9.10.

9.2.4 The Frame Algorithm

It is often the case that an analysis filter bank that corresponds to a frame is given, but that an associated synthesis (inverse) filter bank is not known. While the synthesis filter bank is not unique, we prefer to find the one that corresponds to the dual frame and, thus, to a minimum-squared-error reconstruction. The **frame algorithm** described in this subsection is an iterative procedure to compute the dual frame. The first iteration often provides a useful approximation to the inverse or even the exact inverse.

Let $g = \mathcal{U}f$ be the result of the frame operation on f. We define the frame algorithm by the estimate $f_{(m)}$ of f at iteration m:

$$f_{(m)} = \rho\mathcal{U}^H g + (Id - \rho\mathcal{U}^H\mathcal{U})f_{(m-1)}, \tag{9.45}$$

where ρ is a scalar relaxation parameter, Id is the identity operator, and $f_{(0)} = 0$. The estimation error at iteration m is then

$$f - f_{(m)} = (Id - \rho\mathcal{U}^H\mathcal{U})(f - f_{(m-1)}) = (Id - \rho\mathcal{U}^H\mathcal{U})^m f. \tag{9.46}$$

By selecting the optimal value for ρ, the estimation error can be bounded as follows:

$$\begin{aligned} \|f - f_{(m)}\| &= \min_\rho \|(Id - \rho\mathcal{U}^H\mathcal{U})^m f\| \\ &\leq \min_\rho \max(|1 - \rho A|, |1 - \rho B|)^m \|f\|. \end{aligned} \tag{9.47}$$

The values A and B form the minimum and maximum eigenvalues of the operator $U^H U$ a, which are precisely the frame bounds (we have actually shown this for the $\mathbb{C}^k$ case).

Minimizing the bound on the estimation error results in $\rho = \frac{2}{B+A}$ and the error becomes

$$\|f - f_{(m)}\| = (\frac{B-A}{B+A})^m \|f\|. \tag{9.48}$$

Thus, with the proper selection of ρ, the frame algorithm converges.

The first-iteration estimate of f by the frame algorithm is the expansion $\rho U^H g = \rho \sum_j g_j \psi_j$, which implies that ρU^H is an approximation to the inverse operator. This approximation corresponds to a synthesis filterbank with impulse responses that are the time-reversed impulse reponses of the analysis filterbank, scaled by ρ. Moreover, we see from equation 9.48 that the relative error is bound by the factor $\frac{B-A}{B+A}$. Note that the first-iteration estimate immediately provides the dual frame for $A = B$, which corresponds to a tight frame.

9.3 Non-Adaptive Transforms and Filter Banks

Non-adaptive transforms (including filter banks) are commonly used in source coding. However, it is not immediately obvious that such transforms are useful for this purpose. Let us briefly revisit the arguments for using transforms in source coding. It is important in practical source coding systems to have manageable computational and storage complexity and this means that scalar quantizers or low-dimensional vector quantizers are desirable. In section 7.5, we saw that such quantizers perform, at least asymptotically at high rate, most efficiently if the variables to be encoded are independent. A natural question is then if we can make data samples independent with a transform that do not depend on signal properties.

In general, to make signal data independent, its statistics must be known. However, the discrete-time Fourier transform and the discrete-time cosine transform can convert all discrete stationary Gaussian signals into independent coefficients. Finite-dimensional approximations by means of succesive application of the discrete Fourier transform (DFT) and the discrete cosine transform (DCT) have been found to be powerful tools in many source-coding applications.

The undesirable effects resulting from the approximation of the discrete-time transforms by sequential application of the finite-dimensionality DFT or DCT to successive signal segments can be minimized by using smooth windows and overlap-add synthesis. Thus, the DFT is generalized to the Gabor transform, and the DCT to the modulated lapped transform (MLT). As will be seen in this section, smooth windows require oversampling for the Gabor transform (more coefficients are needed to describe the signal after the transform), but not for the MLT.

We start this section with showing how the Fourier transform can make certain data independent. We then discuss the DFT and DCT and their application to signals. We conclude the section with a discussion of the MLT and the Gabor transform.

9.3.1 The Fourier Transform and Stationary Gaussian Signals

For Gaussian coefficients, statistical independence is equivalent to being uncorrelated. This means that a transform that decorrelates the data is sufficient for asymptotically optimal source coding with a scalar quantizer or a low-dimensional vector quantizer. For stationary Gaussian signals, such a transform is the Fourier transform.

Let $X_w(\mathrm{e}^{j\omega})$ denote the discrete-time Fourier transform of the wide-sense stationary, discrete-time signal $X(n)$ windowed by a suitable window w and let $R(n-m) = \mathrm{E}[X(n)X(m)]$ be the autocorrelation of this signal. Then we have

$$\begin{aligned}
&\mathrm{E}[X_w(\mathrm{e}^{j\omega_1})X_w^*(\mathrm{e}^{j\omega_2})] \\
&= \mathrm{E}[\sum_{n\in\mathcal{Z}} w(n)X(n)\mathrm{e}^{-j\omega_1 n} \sum_{m\in\mathcal{Z}} w(m)X(m)\mathrm{e}^{j\omega_2 m}] \\
&= \sum_{m\in\mathcal{Z}}\sum_{n\in\mathcal{Z}} w(n)w(m)R(n-m)\mathrm{e}^{-j\omega_1 n+j\omega_2 m} \\
&= \sum_{m\in\mathcal{Z}}\sum_{l\in\mathcal{Z}} w(l+m)w(m)R(l)\mathrm{e}^{-j\omega_1(l+m)+j\omega_2 m} \\
&= \sum_{m\in\mathcal{Z}} \mathrm{e}^{-j(\omega_1-\omega_2)m}w(m)\sum_{l\in\mathcal{Z}} R(l)\mathrm{e}^{-j\omega_1 l}w(l+m) \qquad (9.49)
\end{aligned}$$

Let $R(\mathrm{e}^{j\omega})$ be the discrete-time Fourier transform of $R(n)$ and let $R(n)$ decay sufficiently rapidly. Let $w_S(t)$ denote a square window of support S. Then, for sufficiently large support S, we have that the term $\sum_{l\in\mathcal{Z}} R(l)\mathrm{e}^{-j\omega_1 l}w(l+m)$ equals $R(\mathrm{e}^{j\omega_1})$ for almost all values of m within the support region of $w_S(m)$. Furthermore, the factor $\sum_{m\in\mathcal{Z}} \mathrm{e}^{-j(\omega_1-\omega_2)m}$ approaches a Dirac delta function[6] with increasing support S. We write this asymptotic behavior as

$$\mathrm{E}[X_w(\mathrm{e}^{j\omega_1})X^*(\mathrm{e}^{j\omega_2})] \underset{S\to\infty}{\to} \delta(\omega_1-\omega_2)\, R(\mathrm{e}^{j\omega_1}). \qquad (9.50)$$

The asymptotic behavior of 9.50 indicates that the frequency components of stationary Gaussian signals are uncorrelated. It is straightforward to derive an equivalent result for continuous-time functions using the Fourier transform (see problem 11).

In practical applications, we have to use finite-block transforms in the form of the DFT and DCT, which are discussed in the next section. We will see that the finite size of these transforms, which corresponds to a finite window $w(m)$ degrades the performance. We note, however, that having a smooth window and an associated high frequency resolution increases the sharness of the term $\sum_{m\in\mathcal{Z}} \mathrm{e}^{-j(\omega_1-\omega_2)m}w(m)$ in equation 9.49 and, therefore, the effective decorrelation.

9.3.2 Unwindowed Discrete Fourier and Cosine Transforms

The DFT and DCT are useful for the encoding of sequences of vectors. These vector sequences can be subsequent blocks of a one-dimensional signal and the sequential application of the DFT or DCT to these blocks can then be interpreted as a filter bank.

[6] The delta function and the associated symbolic integrals are defined under the theory of distributions (e.g., [59])

As we will see in this section, a finite-size DFT or DCT is an imperfect tool for decorrelating a wide-sense stationary signal. The k-dimensional DFT and DCT decorrelate wide-sense stationary, periodic sequences with a period k. In the case of the DFT, the periodic sequence is simply a repetition of the vector, whereas in the DCT the relation between the periodic sequence and the k-dimensional input vector is somewhat more complicated.

If the input vector is long compared to the decay of the sample correlations, then the DFT and the DCT generally perform an approximate decorrelation that is useful for coding. Even if this is the case, the block boundaries between successive vectors are often the source of significant perceptual distortion in image and audio coding (often referred to as "blocking effects"). A natural solution to these problems is the usage of smoothly windowed frame functions. However, such functions introduce additional theoretical challenges and we postpone their discussion until section 9.3.3.

The Discrete Fourier Transform

The discrete Fourier transform (DFT) is the multiplication of a vector x^k by a matrix U of the form

$$U_{\mathcal{F}} = \frac{1}{\sqrt{k}} \begin{bmatrix} \mathcal{W}^0 & \mathcal{W}^0 & \mathcal{W}^0 & \cdots & \mathcal{W}^0 \\ \mathcal{W}^0 & \mathcal{W}^{-1} & \mathcal{W}^{-2} & \cdots & \mathcal{W}^{-(k-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathcal{W}^0 & \mathcal{W}^{-(k-1)} & \mathcal{W}^{-2(k-1)} & \cdots & \mathcal{W}^{-(k-1)(k-1)} \end{bmatrix}, \tag{9.51}$$

where we used the so-called **twiddle factor**

$$\mathcal{W} = \mathrm{e}^{\frac{j2\pi}{k}}. \tag{9.52}$$

The matrix multiplication displayed here is, of course, not a very clever way of implementing the DFT in terms of computational effort. It is useful for our purposes, however; for the computational aspects of the DFT we refer to the extensive literature on the fast Fourier transform (FFT) (e.g., [60]).

It is easily seen that the rows of $U_{\mathcal{F}}$ form an orthonormal basis in $\mathbb{C}^k$. The inverse matrix is thus simply the transpose conjugate of $U_{\mathcal{F}}$ and we have $U_{\mathcal{F}}^H U_{\mathcal{F}} = I$. By extending the basis vectors with zeros, and then creating additional vectors by shifts as described in section 9.2.3, a basis for $\ell^2(\mathcal{Z})$ can be constructed. The associated frame operator can be interpreted either as a filter bank or as a block-by-block transformation.

The above discussion shows that the DFT leads to straightforward transforms and inverse transforms in both $\mathbb{C}^k$ and $\ell^2(\mathcal{Z})$. Next, we examine whether the DFT is useful for decorrelating signals. For this, let us consider its operation on a zero-mean k-dimensional random vector X^k that is transformed into $Y^k = U_{\mathcal{F}} X^k$. (Note that we can always convert the random vector to be zero mean.) It is convenient for our purpose to write the transform in the form

$$Y_m^k = \sum_{l=0}^{k-1} \mathcal{W}^{-lm} X_l^k. \tag{9.53}$$

We now want to see what the correlations of the components of the random vector y^k are:

$$\begin{aligned} \mathrm{E}[Y_n^k Y_m^{k*}] &= \mathrm{E}[\sum_{l=0}^{k-1} \mathcal{W}^{-ln} X_l^k \sum_{q=0}^{k-1} \mathcal{W}^{qm} X_q^{k*}] \\ &= \sum_{l=0}^{k-1}\sum_{q=0}^{k-1} \mathcal{W}^{-ln+qm} \mathrm{E}[X_l^k X_q^{k*}]. \end{aligned} \tag{9.54}$$

So-far the DFT does not seem to lead to a decorrelation. Looking for inspiration at the derivation of equation 9.50 we see that we would like to write $\mathrm{E}[X_l^k X_q^{k*}] = R(l,q) = R_{l-q}$, i.e., the correlation should depend only on the difference $l-q$. This is true if the sequence is a segment of a periodic and wide-sense stationary sequence. This implies that any realization of X^k extends periodically:

$$x_{n+mk}^k \equiv x_n^k, \; n \in \{0,1,\cdots,k-1\}, \; m \in \mathcal{Z}. \tag{9.55}$$

These extensions are shown in figure 9.8. The extensions allow us to decorrelate the vectors (and also define values for the correlations for $l-q > k$). We can continue the derivation in equation 9.54 to obtain

$$\begin{aligned} \mathrm{E}[Y_n^k Y_m^{k*}] &= \sum_{l=0}^{k-1}\sum_{q=0}^{k-1} \mathcal{W}^{-ln+qm} R_{l-q} \\ &= \sum_{p=0}^{k-1}\sum_{q=0}^{k-1} \mathcal{W}^{-pn+q(m-n)} R_p \\ &= \sum_{q=0}^{k-1} \mathcal{W}^{q(m-n)} \sum_{p=0}^{k-1} \mathcal{W}^{-pn} R_p \\ &= \delta_{mn} \sum_{p=0}^{k-1} \mathcal{W}^{-pn} R_p. \end{aligned} \tag{9.56}$$

Equation 9.56 shows that if we assume that the vectors x^k extend periodically, then $\mathrm{E}[Y_n^k Y_m^{k*}]$ is diagonal and the components of Y^k are decorrelated.

Let us consider qualitatively the consequences of our periodicity assumption. The periodic extension assumption results in a **circulant** autocorrelation matrix. Circulant matrices are Toeplitz (constant along the diagonals) matrices, where each row (column) forms a cyclically shifted version of the neighboring rows (columnns). The DFT performs a better approximation to decorrelation of a block of samples of a stationary signal if the differences between the circulant autocorrelation matrix and the actual autocorrelation matrix are minor. This is more likely with increasing dimensionality since R_i is generally a decreasing function of $|i|$.

Example 9.11: Circulant autocorrelation matrices
For an example with $k = 5$, the circulant autocorrelation matrix would be of the form

$$R \equiv \mathrm{E}[x^5 x^{5T}] = \begin{bmatrix} R_0 & R_1 & R_2 & R_2 & R_1 \\ R_1 & R_0 & R_1 & R_2 & R_2 \\ R_2 & R_1 & R_0 & R_1 & R_2 \\ R_2 & R_2 & R_1 & R_0 & R_1 \\ R_1 & R_2 & R_2 & R_1 & R_0 \end{bmatrix},$$

where each row is a circular shift of the next row. In contrast, the "true" autocorrelation function corresponding to a segment of a stationary signal would be of the form

$$R = \begin{bmatrix} R_0 & R_1 & R_2 & R_3 & R_4 \\ R_1 & R_0 & R_1 & R_2 & R_3 \\ R_2 & R_1 & R_0 & R_1 & R_2 \\ R_3 & R_2 & R_1 & R_0 & R_1 \\ R_4 & R_3 & R_2 & R_1 & R_0 \end{bmatrix}.$$

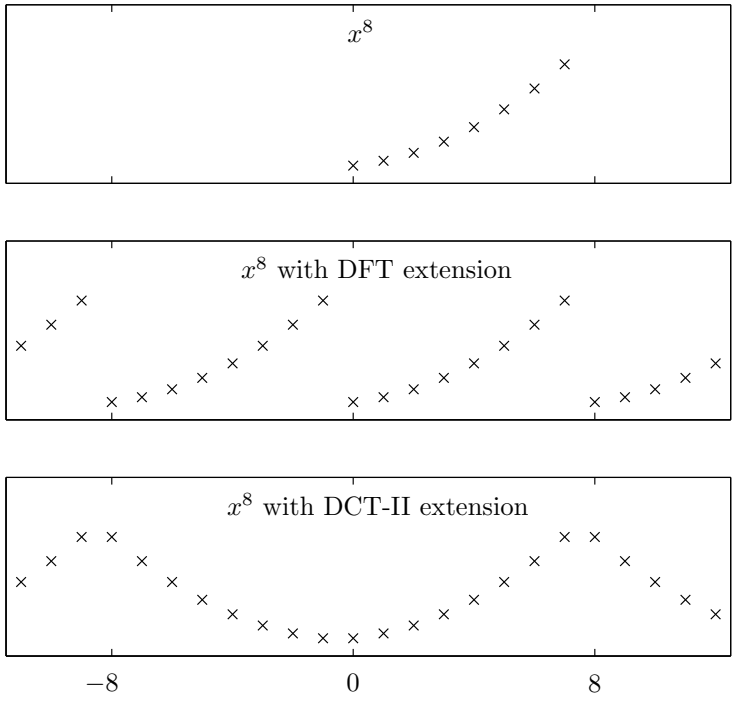

Figure 9.8: An eight-dimensional vector, x^8, and the extensions assumed for DFT and DCT decorrelation.

The Discrete Cosine Transform

Like the DFT, the DCT has decorrelating properties. However, because of the symmetry assumptions made, the DCT results in a better decorrelation for a given block size than the DFT for most practical signals, e.g., [61, 62]. As a result, the DCT is more commonly used in practical source coding applications than the DFT.

A number of different flavors of the DCT exist, each with different symmetries. We base

our discussion on the so-called DCT-II transform:

$$y_i = c(i)\sqrt{\frac{2}{k}}\sum_{n=0}^{k-1} x_n \cos(\frac{(2n+1)i\pi}{2k}), \tag{9.57}$$

$$x_i = \sqrt{\frac{2}{k}}\sum_{n=0}^{k-1} c(n)y_n \cos(\frac{(2n+1)i\pi}{2k}), \tag{9.58}$$

where $c(0) = 1/\sqrt{2}$ and $c(i) = 1$, $i \in \{1, \cdots, k-1\}$.

Again, the derivation of equation 9.50 can be adapted to the transform at hand, and again we have to make an additional modification. The basis vectors of the DCT-II exhibit a $2k$ periodicity and a symmetry. The assumed extensions are

$$\begin{aligned} x_{n+2k} &= x_n, \\ x_{2k-1-n} &= x_n. \end{aligned} \tag{9.59}$$

This extension is also shown in figure 9.8. The resulting decorrelation can be written as

$$\mathrm{E}[Y_m Y_n^*] = \delta_{mn} c(m)^2 \sum_{i=0}^{k-1} R_i \cos((2i+1)m\pi). \tag{9.60}$$

From the symmetries of the DCT-II, we can see two reasons why this DCT should perform better than the DFT in the decorrelation of a block of samples for a stationary signal: *i*) there is no jump at the extension *ii*) the correlations across the extension are averaged over different left and right extensions. Thus, the autocorrelation matrix corresponding to DCT-II extension is generally more similar to the true autocorrelation of the stationary signal than that corresponding to a DFT extension.

9.3.3 Windowed Discrete Fourier and Cosine Transforms

As was seen in the previous section, the DFT and DCT can decorrelate a stationary sequences only approximately, because of boundary effects. We noted in section 9.3.1 that smooth windows will aid the effective decorrelation. Furthermore, in practical coding it is often useful to resolve time-frequency events in signals, and the boxcar (square) window often performs poorly in this respect [7]. Thus, smoothly tapered windows are commonly used in the context of filter banks and source coding.

Analysis-Only Windowing versus Windowed Frame Functions

A simple and commonly used method for introducing windowing to block transforms uses windowing only during the analysis operation. That is, the first step of the transform is to create a sequence of overlapping windowed signals segments. The windows are selected such that they sum to one. The second step is to perform the DCT or DFT on every windowed segment. The inverse transform does not contain a windowing operation and consists of performing the inverse DCT (or DFT) and summing the

[7] Rectangular windows correspond to a $\frac{1}{\omega^2}$ decay of the signal power in the frequency domain; this is slow compared to tapered windows and leads to an infinite product of time and frequency variance.

resulting segments. Since the windows corresponding to the segments sum to one, the summation of the reconstructed segments results in the original signal. The filter interpretation of this method is shown in figure 9.9. Note that the synthesis is an expansion in unwindowed basis functions.

While the analysis-only windowing method is straightforward and provides perfect reconstruction, it has a number of disadvantages:

1. The approach is inherently oversampled since the segments overlap.
2. Quantization (or other) operations in the transform domain result in discontinuities in the time domain, since the synthesis functions are not tapered.
3. The transform does not preserve energy even when the basis used is orthonormal. That is, the set of functions $\{w(n-mM)\phi_j(n-mM)\}_{m\in\mathcal{Z},j=1,\cdots,M}$ forms a frame (which makes reconstruction possible) but not a tight frame.

To avoid these problems, we consider in the following sections only methods where the window is built into the frame functions. The filtering interpretation of this method is illustrated in figure 9.10, whereas the transform interpretations are used and discussed in detail in sections 9.3.4 and 9.3.5. The frame functions are smoothly windowed cosines or complex exponentials. As was mentioned before, the combination of both smooth windowing and critical sampling exists for DCT based transforms but not for the Gabor transform.

Aliasing in Windowed Fourier and Cosine Transforms

As is shown in figure 9.5, the analysis filter bank equivalent of the analysis-frame operator is a UFB that contains a down-sampling operation. It is well-known that if a signal is not band-limited to half the sampling frequency, aliasing results.

In the case of the windowed Fourier and cosine transforms, the bandwidth of a particular channel j is determined by the bandwidth of the window. Let us define the sampling rate of the original signal $f(n)$ to be 1. For a critically sampled filter bank, the sampling rate of the coefficient signals $g_j(m)$ is then $\frac{1}{M}$. The bandwidth of the square window with support M is much wider than $\frac{1}{2M}$, resulting in severe aliasing. As a result, the critically sampled Gabor transform, which must use a square window (shown below), suffers from severe aliasing.

Aliasing can be reduced by two methods: *i*) by reducing the down-sampling factor to less than M *ii*) by increasing the support and smoothness of the window. Reduction of the down-sampling can be applied to both windowed cosines and windowed complex exponentials (Gabor transforms). The technique of increasing the support and smoothness of the window is preferred, since it does not lead to an increase in the number of coefficients per unit time. However, as was mentioned before and as will be shown in section 9.3.5, this is not possible for a critically sampled Gabor filter bank.

The effects of aliasing are generally most severe if nonlinear operations (such as quantization and adaptive filtering, where the filter depends on the input) are performed on the individual coefficients channels $g_j(m)$ under the assumption that they represent the signal within the corresponding frequency band. For linear operations, the processing of the unaliased component is unaffected by the aliasing and the overall impact is generally

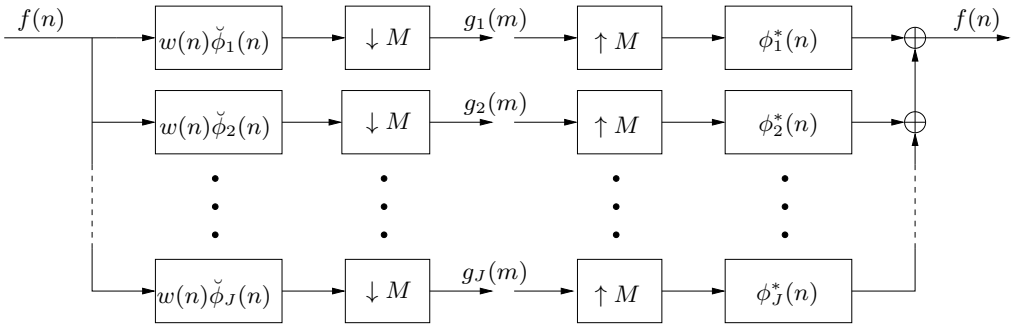

Figure 9.9: Analysis-synthesis filter bank using windowing for analysis only. The window $w(n)$ satisfies $\sum_{m\in\mathcal{Z}} w(n-Mm) = 1$ and the functions $\{\phi_j(n)\}_{j=1,\cdots,M}$ form a (discrete Fourier or discrete cosine) orthonormal basis over the support of the window.

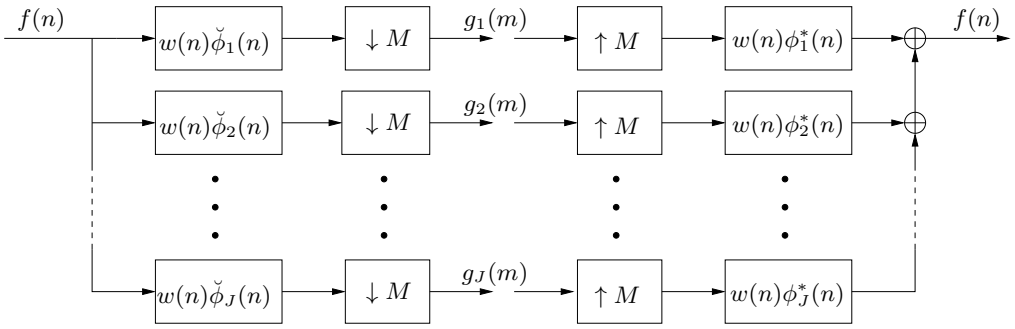

Figure 9.10: Analysis-synthesis filter bank with windowed functions $\{w(n-mM)\phi_j(n-mM)\}_{m\in\mathcal{Z},j=1,\cdots,M}$ of n that form a tight frame in the Hilbert space $\ell^2(\mathcal{Z})$. The figure assumes $w(n)$ is symmetric. The functions $\phi_j(n)$ are complex exponentials or cosines. The MLT and Gabor transform are described by this figure.

less. This is the case, for example, for performing filtering of the original signal $f(t)$ in the frequency domain by means of Fourier transforms.

Finally, we note that usage of windows that are too smooth can also lead to problems. Windows that are too smooth result in an effective total frequency-domain support that is less than the bandwidth of the original signal, which makes it difficult to recover the original signal with finite-precision computing machines. The range of smoothness where neither aliasing or oversmoothing is a problem increases with the oversamping rate.

9.3.4 The Modulated Lapped Transform

The combination of smooth windowing and critical sampling is difficult to achieve but attractive for many applications. The modulated lapped transform (MLT) is based on

the DCT and falls into the class of smoothly windowed, critically sampled filter banks. In the following, we provide a derivation of a particular MLT. For a more in-depth discussion of lapped transforms we refer to [63].

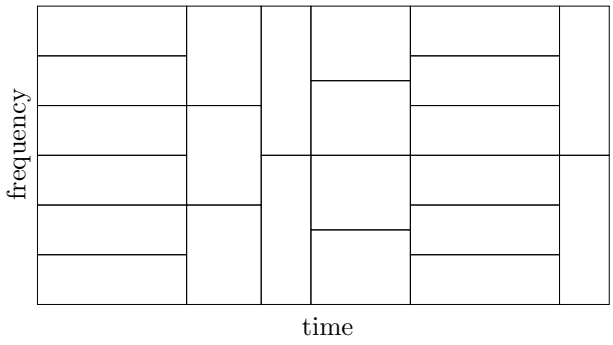

Figure 9.11: A time-frequency tiling obtained with a lapped transform.

In lapped transforms, the discrete time signal is described by means of a basis of $\ell^2(\mathcal{Z})$ that is usually constructed from a set of cosines multiplied by a set of windows. Often, these windows are uniformly spaced translates of a basic finite-support window, but the windows may also have unequal time support. Such a situation is illustrated schematically in figure 9.11, which shows the regions where each basis function dominates the signal energy, providing a so-called time-frequency **tiling**. The description of the MLT by means of the simple matrix multiplication method of section 9.2.1 (albeit with a square matrix of infinite dimensionality) provides good insight into its operation:

$$U = \begin{bmatrix} & & \ddots & & & & & & \\ \cdots & 0 & G_{\text{left}} & G_{\text{mid}} & G_{\text{right}} & 0 & 0 & 0 & \cdots \\ \cdots & 0 & 0 & 0 & G_{\text{left}} & G_{\text{mid}} & G_{\text{right}} & 0 & \cdots \\ & & & & & & \ddots & & \end{bmatrix}. \tag{9.61}$$

The J rows of the matrix $G = [G_{\text{left}}\, G_{\text{mid}}\, G_{\text{right}}]$ define a set of basis functions, each roughly corresponding to a frequency band, from which the complete set of basis functions is obtained by time shifts of J samples. In the following, we will consider the case where the tails on the left and the right have equal support T ($T < J$). The rows of the $J \times T$ matrices G_{left} and G_{right} form the "tails" of this set of basis functions, and the $J \times (J - 2T)$ matrix $\mathcal{G}_{\text{mid}}$ can be thought of as the "body" of the basis functions.

Our aims are to *i*) make U unitary and *ii*) make each row of G the multiplication of a smooth window and a cosine function. The unitarity condition implies $U^H U = I$. Since the matrix U is square, we can write the unitarity condition also as $UU^H = I$ (see the argument leading up to equation 9.11). This form of the unitarity condition is more convenient since it allows an interpretation in terms of inner products of the basis functions (the rows of U). The condition $UU^H = I$ implies the following conditions for the submatrices:

$$G_{\text{left}}G_{\text{left}}^H + G_{\text{mid}}G_{\text{mid}}^H + G_{\text{right}}G_{\text{right}}^H = I, \tag{9.62}$$

$$G_{\text{right}}G_{\text{left}}^H = 0. \tag{9.63}$$

Condition 9.63 is commonly referred to as the **orthogonality of tails**.

Next, we make the windowing explicit by factoring. We describe the windowing with a diagonal $T \times T$ matrix W containing only positive, real numbers along the diagonal. We write the matrices G in terms of W and modulation matrices $\mathcal{G} = [\mathcal{G}_{\text{left}}\, \mathcal{G}_{\text{mid}}\, \mathcal{G}_{\text{right}}]$:

$$G_{\text{left}} = \mathcal{G}_{\text{left}} W, \tag{9.64}$$
$$G_{\text{mid}} = \mathcal{G}_{\text{mid}}, \tag{9.65}$$
$$G_{\text{right}} = \mathcal{G}_{\text{right}} \breve{W}, \tag{9.66}$$

where $\breve{W}$ is the matrix W with its columns (or rows) reversed.

The orthogonality of the tails (equation 9.63) can be written as

$$\mathcal{G}_{\text{left}} W \breve{W} \mathcal{G}_{\text{right}}^H = 0. \tag{9.67}$$

The diagonal windowing matrix $\breve{W}W$ has an even symmetry along its diagonal. Thus, the orthogonality of the tails is satisfied if the rows of $\mathcal{G}_{\text{left}}$ have **even symmetry** and the rows of $\mathcal{G}_{\text{right}}$ have **odd symmetry** or vice versa.

We impose on W the **power complementarity** condition,

$$WW + \breve{W}\breve{W} = I, \tag{9.68}$$

which implies that

$$\frac{1}{2}I - WW = \breve{W}\breve{W} - \frac{1}{2}I \tag{9.69}$$

must have odd symmetry along the diagonal. The odd and even symmetry of the tails and the fact that $\frac{1}{2}I - WW = \breve{W}\breve{W} - \frac{1}{2}I$ has odd symmetry results in the following useful properties:

$$\mathcal{G}_{\text{left}}(\frac{1}{2}I - WW)\mathcal{G}_{\text{left}}^H = 0, \tag{9.70}$$
$$\mathcal{G}_{\text{right}}(\frac{1}{2}I - \breve{W}\breve{W})\mathcal{G}_{\text{right}}^H = 0. \tag{9.71}$$

Using equations 9.70 and 9.71, we see that equation 9.62 can be written as

$$\begin{aligned} I &= \mathcal{G}_{\text{left}} WW \mathcal{G}_{\text{left}}^H + \mathcal{G}_{\text{mid}} \mathcal{G}_{\text{mid}}^H + \mathcal{G}_{\text{right}} \breve{W}\breve{W} \mathcal{G}_{\text{right}}^H \\ &= \frac{1}{2}\mathcal{G}_{\text{left}} \mathcal{G}_{\text{left}}^H + \mathcal{G}_{\text{mid}} \mathcal{G}_{\text{mid}}^H + \frac{1}{2}\mathcal{G}_{\text{right}} \mathcal{G}_{\text{right}}^H. \end{aligned} \tag{9.72}$$

To summarize, to obtain unitarity of U we must have that

1. $\mathcal{G}$ satisfies 9.72;
2. the tails of $\mathcal{G}_{\text{right}}$ and $\mathcal{G}_{\text{left}}$ must have opposite symmetry.

For the simple case where $J = T$, i.e., where we have only the tails $\mathcal{G}_{\text{left}}$ and $\mathcal{G}_{\text{right}}$ and no body $\mathcal{G}_{\text{mid}}$, the first condition reduces to $\mathcal{G}\mathcal{G}^H = 2I$, i.e., the rows of $\sqrt{\frac{1}{2}}\mathcal{G}$ are orthonormal.

It is easy to check that both conditions are satisfied by the so-called DCT-IV functions:

$$\mathcal{G}_{in} = \sqrt{\frac{2}{J}} \cos(\frac{(2i+1)}{4J}(2n - J + 1)\pi). \tag{9.73}$$

Over the interval $[0, \cdots, J-1]$, these functions display even symmetry around $(J-1)/2$. Over the interval $[J, 2J-1]$ they are anti-symmetric around $J + (J-1)/2$.

Before concluding this section, it is useful to reflect on the smoothness properties of the MLT. The MLT was motivated by the need for having smoothly windowed functions. However, the anti-symmetries used in our construction have in some circumstances a similar effect as discontinuities. That is, if an original signal that does not have this odd symmetry is modeled with an MLT, then quantization errors introduce some of this odd symmetry and this may be perceptually undesirable.

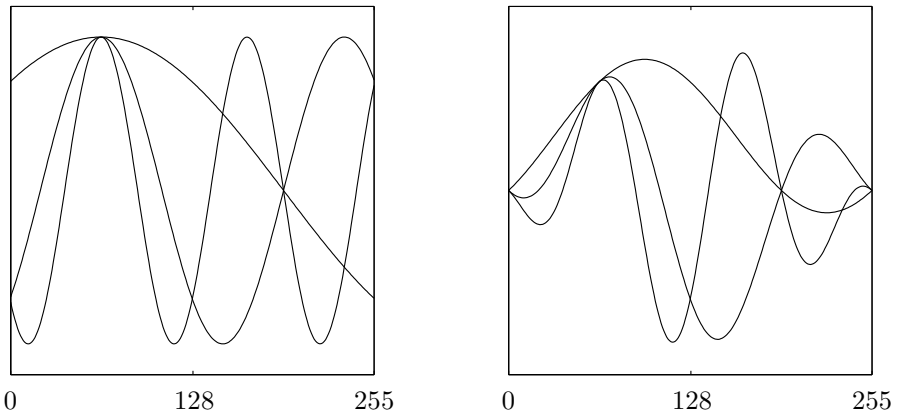

Figure 9.12: **Left**: the first three rows of the matrix $\mathcal{G}$ of example 9.12 for $J = 128$. The left tail is symmetric around $63\frac{1}{2}$ and the right tail is anti-symmetric around $191\frac{1}{2}$. **Right**: the first three MLT basis functions ϕ_{00}, ϕ_{10}, and ϕ_{20} for example 9.12, again for $J = 128$.

Example 9.12: A practical MLT implementation
In the tails-only case, the relation between G and $\mathcal{G}$ can be written as

$$G = \mathcal{G} \begin{bmatrix} W & 0 \\ 0 & \breve{W} \end{bmatrix}.$$

where the rows of $\mathcal{G}$ are

$$\mathcal{G}_{in} = \sqrt{\frac{2}{J}} \cos(\frac{(2i+1)}{4J}(2n - J + 1)\pi), \;\; n \in \{0, \cdots, 2J-1\}.$$

A window satisfying the power complementarity condition 9.68 is

$$W_n = \sin(\frac{n\pi}{2(J-1)}), \;\; n \in \{0, \cdots, J-1\}.$$

Note that W_n describes the diagonal elements of only the first half of the windowing matrix for $\mathcal{G}$.

The rows of the matrix G form the basis functions of the MLT. Thus, we can now write down a subset of basis functions for a practical (tails-only) MLT:

$$\phi_{i0}(n) = \begin{cases} \sin(\frac{n}{2(J-1)}\pi)\sqrt{\frac{2}{J}}\cos(\frac{(2i+1)(2n-J+1)}{4J}\pi), & n \in \{0,\cdots,J-1\}, \\ \sin(\frac{2J-1-n}{2(J-1)}\pi)\sqrt{\frac{2}{J}}\cos(\frac{(2i+1)(2n-J+1)}{4J}\pi), & n \in \{J,\cdots,2J-1\}, \\ 0, & n \in \mathcal{Z} - \{0,\cdots,2J-1\}, \end{cases}$$

where the frequency index satisfies $i \in \{0,\cdots,J-1\}$. The remaining basis functions $\phi_{il}(n)$ are obtained by time-shifts over multiples of J:

$$\phi_{il}(n) = \phi_{i0}(n+lJ), \quad l \in \mathcal{Z}.$$

The unwindowed matrix rows $\mathcal{G}_{0l}$, $\mathcal{G}_{1l}$, and $\mathcal{G}_{2l}$, as well as the MLT basis functions, ϕ_{0l}, ϕ_{1l}, ϕ_{2l} are illustrated in figure 9.12, for the case $J = 128$.

Performing the MLT is straightforward. In the forward transform, we multiply the signal by the MLT basis functions to obtain the MLT coefficients $\langle f, \phi_{il}\rangle$. The inverse transform consists of summing the same basis functions scaled by the MLT coefficients: $f(n) = \sum_{i,l}\langle f, \phi_{il}\rangle\, \phi_{il}(n)$.

9.3.5 The Gabor Transform

The main attraction of the Gabor transform is that it is based on the discrete Fourier transform and describes a signal in terms of frequencies, amplitudes, and phases. However, in contrast to the MLT, it requires oversampling if smooth windows are to be used.

For convenience, we use two **Gabor frame** definitions. In the first definition, we follow the UFB approach to create the Gabor frame by M-sample translations of a set of J windowed complex exponentials, $w(n)\exp(j\frac{2\pi mn}{J})$, where $m \in \{0,\cdots,J-1\}$ indexes frequency. In the second definition we construct the frame by multiplying a set of J complex exponentials $\exp(j\frac{2\pi mn}{J})$, with $n \in \mathcal{Z}$, with uniformly M-sample spaced translates of the window $w(n)$. We note that the difference is a simple phase factor in the coefficients.

We start with a simple discussion of the Gabor transform for the case that the number of frequency channels, J, is equal or more than the support of the window $w(n)$. Thereafter, we give a more formal treatment of the Gabor transform that does not require this condition.

The Gabor Transform for "Short" Windows

When the support of the window is J samples or less, the Gabor transform has a straightforward interpretation. In a first step, overlapping sequences of J samples are extracted. Subsequently extracted sequences are offset by M samples. In a second step, each data sequence is windowed by multiplying with a diagonal matrix W of dimensionality $J \times J$, which has as diagonal elements the window samples $w(n)$ (with centered support). Note that the support of the window can be less than J. In the third and final step, each windowed data set Wx_m^J is subjected to multiplication by an $J \times J$ DFT matrix, $U_{\mathcal{F}}$. The coefficient vectors

$$y_m^J = U_{\mathcal{F}} W x_m^J \tag{9.74}$$

for block m form the Gabor transform coefficients corresponding to this block. The rows of the matrix $U_{\mathcal{F}}W$ form the conjugates of the frame functions. The Hermitian-transpose transform consists of first multiplying each subvector by the inverse DFT matrix and then by a diagonal windowing matrix W (we assume W is real):

$$\begin{aligned} z_m^J &= WU_{\mathcal{F}}^H y_m^J \\ &= WU_{\mathcal{F}}^H U_{\mathcal{F}} W x_m^J \\ &= WWx_m^J. \end{aligned} \tag{9.75}$$

For final reconstruction, we add all vectors z_m^J, offset by mM. That is, each sequence $z_m^J = WWx_m^J$ contributes to the segment from which the corresponding x_m^J was extracted. The contributions are added together by means of an overlap-add mechanism. This means that the Gabor frame is tight if the window satisfies the power-complementarity condition

$$\sum_{m\in\mathcal{Z}} w(n+mM)^2 = \frac{A}{J}, \quad n \in \{0,\cdots,M-1\}. \tag{9.76}$$

(Note that the factor $1/J$ is generally part of the inverse DFT transform.)

Doubly oversampled Gabor transforms are perhaps most common. The frame functions of a tight Gabor frame with double over-sampling have a simple, appealing form:

$$\psi_{lm}(n) = w(n-mM)\exp(j\frac{2\pi ml}{J}), \tag{9.77}$$

where $w(n)$ satisfies the power complementarity condition 9.76 and is zero outside the interval $0,\cdots,2M-1$.

Example 9.13: A practical Gabor frame with double over-sampling
Using a sinusoidal window of appropriate power complementarity, we can design a doubly over-sampled Gabor frame:

$$w_s(n) = \begin{cases} \sqrt{\frac{1}{J}}\sin(\frac{\pi n}{J}), & n \in \{0,\cdots,J-1\}, \\ 0, & \text{elsewhere.} \end{cases}$$

An over-sampled Gabor frame can thus be constructed using functions of the form

$$\phi_{lm}(n) = w_s(n-m\frac{J}{2})\exp(j\frac{2\pi mn}{J}), \quad m \in \{0,\cdots,J-1\},\ l \in \mathcal{Z},$$

where n represents time, m is the frequency index of the frame function, and l is the time index of the frame.

Implementation of the forward and backward transforms of the doubly over-sampled tight Gabor transform is straightforward. The operations are similar to those in the MLT transform described in example 9.12. However, since the Gabor transform is oversampled, the synthesis frame functions differ from the analysis frame functions by an additional factor 1/2.

Basic Conditions for Tight Gabor Frames

We now derive the general conditions for constructing a tight Gabor frame with J channels (J frequency samples per 2π radians) and a rational over-sampling rate[8]. That

[8] [64] provides an alternative derivation using polyphase matrices.

is, for every M time samples there are J frequency samples. In contrast to the preceding simple derivation, the window support is now unconstrained.

To simplify our notation, we again use the twiddle factor

$$\mathcal{W} \equiv \exp(j\frac{2\pi}{J}). \tag{9.78}$$

To gain insight in the derivation of a tight Gabor frame, we use matrix notation. Let U be an infinite-dimensionality matrix of the form

$$U_{in} = \mathcal{W}^{\rho(i,J)n}\, w(n - M\lfloor\frac{i}{J}\rfloor), \quad i, n \in \mathcal{Z}, \tag{9.79}$$

where $\lfloor\cdot\rfloor$ indicates rounding down to the nearest integer and where

$$\rho(i, J) = -i + J\lfloor\frac{i}{J}\rfloor. \tag{9.80}$$

The Gabor frame functions are defined as the complex conjugates of the rows of the matrix U.

The structure of the matrix U is illustrated in figure 9.13.

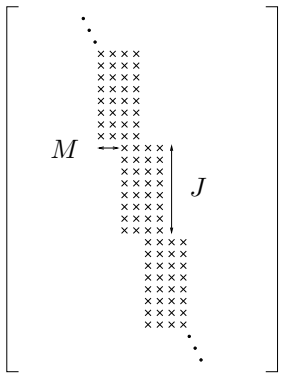

Figure 9.13: The structure of the matrix U. Every J rows, the window shifts by M samples. For critical sampling $J = M$ and for over-sampling $J > M$.

For a tight frame we have $U^H U = AI$, where A is the frame bound and the elements of U satisfy

$$\sum_m U^H_{im} U_{mn} = \sum_m U^*_{mi} U_{mn} = A\,\delta_{in}. \tag{9.81}$$

Substituting equation 9.79 into equation 9.81, we obtain the conditions

$$\sum_m w(n - M\lfloor\frac{m}{J}\rfloor)^2 \quad = \quad A, \quad \forall_{n\in\mathcal{Z}} \tag{9.82}$$

$$\sum_m \mathcal{W}^{\rho(m,J)(n-i)} w(i - M\lfloor\frac{m}{J}\rfloor)\, w(n - M\lfloor\frac{m}{J}\rfloor) \quad = \quad 0, \quad \forall_{n\in\mathcal{Z},\, i\in\mathcal{Z},\, i\neq n}. \tag{9.83}$$

In the summation of equation 9.83, the window functions change only after every J terms. Furthermore, we have that

$$\sum_{m=0}^{J-1} \mathcal{W}^{\rho(m,J)(n-i)} = \begin{cases} J, & n-i \text{ multiple of } J, \\ 0, & \text{otherwise.} \end{cases} \tag{9.84}$$

Condition 9.83 can then be simplified to

$$\sum_m w(l - M\lfloor\frac{m}{J}\rfloor)\, w(l + qJ - M\lfloor\frac{m}{J}\rfloor) \quad = \quad 0, \quad \forall_{l\in\mathcal{Z}, q\in\mathcal{Z}-\{0\}}. \tag{9.85}$$

The formulation of 9.85 is redundant since each term in the summation is repeated J times. Eliminating this redundancy in the summation, we can write

$$\sum_m w(l - mM)\, w(l + qJ - mM) \quad = \quad 0, \quad \forall_{l\in\{0,\cdots,M-1\}, q\in\mathcal{Z}-\{0\}}. \tag{9.86}$$

Equations 9.82 and 9.86 form the necessary and sufficient conditions for a tight Gabor frame.

The conditions for a tight Gabor frame are simpler to understand for the case that J/M is an integer. Equation 9.86 shows that it is then convenient to define M independent "phases" of the window function $w(\cdot)$. Each of these phases, $w_l(m) = w(l+mM)$ must be orthogonal to its translates by multiples of the integer J/M. We can then design M such windows $w_l(\cdot)$, each representing one phase of the window $w(\cdot)$. Rewriting equations 9.82 and 9.86 in terms of the various window phases, we see that the conditions for a tight Gabor frame with J/M an integer become

$$\sum_m w_l(m)^2 \quad = \quad \frac{A}{J}, \quad l \in \{0, \cdots, M-1\}, \tag{9.87}$$

$$\sum_m w_l(m)\, w_l(q\frac{J}{M} + m) \quad = \quad 0, \quad l \in \{0, \cdots, M-1\}, q \in \mathcal{Z} - \{0\}. \tag{9.88}$$

Equation 9.88 is a shift-orthogonality condition.

The Constraints on Orthonormal Gabor Frames

The shift-orthogonality condition 9.88 has important implications for the properties of critically-sampled ($J = M$) Gabor frames. It turns out that critically-sampled Gabor frames must have a rectangular window, which means that critical sampling results in low frequency resolution. This can be seen as follows. An orthonormal basis is a tight frame with critical sampling ($J = M$). Shift-orthogonality condition 9.88 becomes for this case

$$\sum_m w_l(m)\, w_l(q + m) \quad = \quad 0, \quad q \in \mathcal{Z} - \{0\}. \tag{9.89}$$

For a window w_l with a finite support larger than 1, we can always choose q such that the sum in this equation has only one nonzero term, violating the condition. Thus, for an orthonormal frame, the window w_l must have a support of one sample and it

then follows from equation 9.87 that w is a rectangular window of length M. Thus, a critically-sampled Gabor frame with a smooth window does not exist. This is a fundamental result, which was developed during the nineteen-eighties (e.g., [65, 66]). Its continuous equivalent is commonly referred to as the **Balian-Low theorem**[9].

Tight Gabor Frames with Integer Oversampling and Limited Time Support

When J/M is an integer, then each of the M polyphase windows $w_l(n)$ must be orthogonal to itself shifted by J/M. This is trivial for a window w_l with a support of only J/M samples (or less), corresponding to an overall window w with support J (or less). In this case, shift-orthogonality condition 9.88 is always satisfied and the only remaining condition is the power complementarity condition

$$\sum_{m=0}^{J/M} w_l(m)^2 \quad = \quad \frac{A}{J}. \tag{9.90}$$

Realizing that $w(l+nM) = w_l(n)$, it is seen that condition 9.90 corresponds to equation 9.76, which we earlier derived in a more direct and simple manner.

9.4 Transforms with A-Priori Adaptation

In this section, we optimize the transform assuming that the statistics of the data vector are known. We consider two scenarios that have in common the mean squared-error criterion. In the first scenario, we determine the transform that minimizes the overall distortion, assuming independent coding of the resulting vector components and a certain distortion-rate behavior. In the second scenario, we determine the optimal transform for encoding only a subset of the vector components of the transformed data vectors. That is, in the second scenario, we encode a best linear approximation of the data vector. In both scenarios, we find the Karhunen-Loève transform.

9.4.1 Definition of the Karhunen-Loève Transform

The Karhunen-Loève transform for a random vector X^k is a unitary transform that diagonalizes its covariance matrix. For the case that the vector components of X^k are zero mean, the Karhunen-Loève transform, U, diagonalizes the autocorrelation matrix $\mathrm{E}[X^k X^{kH}]$:

$$U\mathrm{E}[X^k X^{kH}]U^H = \begin{bmatrix} \lambda_0 & 0 & \cdot & \cdot & 0 \\ 0 & \lambda_1 & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \lambda_{k-2} & 0 \\ 0 & \cdot & \cdot & 0 & \lambda_{k-1} \end{bmatrix}. \tag{9.91}$$

Thus, the Karhunen-Loève transform is computed with a simple eigenvalue decomposition of the autocorrelation matrix. It transforms a random vector into a vector with uncorrelated components, and, thus, independent components if the vector is Gaussian.

[9] The Balian-Low theorem states [66]: if $g_{mn}(t) = \mathrm{e}^{2\pi jmt} g(t-n)$ forms a frame for the continuous Hilbert space $\mathcal{L}(\mathbb{R})$, then either $\int \omega^2 |\mathcal{F}g(\omega)|^2 d\omega = \infty$ or $\int t^2 |g(t)|^2 dt = \infty$.

9.4.2 Transform for Independent Coding of Components

In section 7.5, we saw that a scalar quantizer or a low-dimensionality vector quantizer performs best when the vector components are independent. For Gaussian data, this is equivalent to the vector components being uncorrelated. In section 9.3, we set out to decorrelate a large class of data vectors with fixed transforms and ended up with transforms such as the DCT and MLT. They were only asymptotically optimal, however. Naturally, we can perform better decorrelation of a finite-dimensionality vector if we know its statistics and optimize the transform for the statistics. As we will see, under certain conditions, this type of decorrelation results in the Karhunen-Loève transform.

In the following, we will prove that the Karhunen-Loève transform minimizes the distortion associated with scalar quantization of vector components under the following conditions:

1. The distortion criterion is the squared error criterion.
2. The sum of the average rates of the vector components is given.
3. The distortion-rate function for the components of a random vector X^k is of the form $\mathrm{E}[d_i] = C\ \sigma_{X_i}^2 \mathrm{e}^{-2R_{X_i}}$, where $\sigma_{X_i}^2$ is the variance of a random component X_i and C is a constant.
4. The transform is optimized a-priori for the vector statistics.

The condition that the distortion-rate function for each of the components of the random vector X^k is of the form

$$\mathrm{E}[d_i] = C\ \sigma_{X_i}^2 \mathrm{e}^{-2R_{X_i}}, \tag{9.92}$$

is correct if X^k has a Gaussian multi-variate density, if the rate is more than zero for each component (this corresponds to the high-rate requirement[10]), and if the squared error criterion applies[11]. Fortunately, the form of the rate-distortion function is not affected by a unitary transformation.

We now start the derivation. The objective is to find the unitary transform U that minimizes the total distortion

$$D = \sum_{i=0}^{k-1} \mathrm{E}[d_i] = \sum_{i=0}^{k-1} C\ \sigma_{Y_i}^2 \mathrm{e}^{-2R_{Y_i}}, \tag{9.93}$$

where $Y^k = UX^k$, under the constraint

$$k\bar{R} = \sum_{i=0}^{k-1} R_{Y_i}, \tag{9.94}$$

where $\bar{R}$ is the (constrained) average of the entropy of the vector components.

[10] Note that the meaning of "high rate" varies: for the Gaussian multi-variate density case it means that no component has zero rate; for the scalar quantizer it means that the density varies slowly compared to the quantizer cell size.

[11] Since the entropy of the indices of a high-rate scalar quantizer for Gaussian data is also of this form, the results apply directly to practical entropy-constrained scalar quantization and lossless coding.

To optimize the rate distribution, we use the familiar Lagrange multiplier method, with λ as Lagrange multiplier. The extended criterion then becomes:

$$\eta = \sum_{i=0}^{k-1} \left(C\ \sigma_{Y_i}^2 \mathrm{e}^{-2R_{Y_i}} + \lambda R_{Y_i} \right). \tag{9.95}$$

Differentiating equation 9.95 with respect to R_{Y_i} and setting the result equal to zero gives

$$R_{Y_i} = \frac{1}{2} \log(\frac{2\sigma_{Y_i}^2 C}{\lambda}). \tag{9.96}$$

Combining this with the constraint 9.94 results in

$$R_{Y_i} = \bar{R} + \log(\sigma_{Y_i}) - \frac{1}{k} \sum_{j=0}^{k-1} \log(\sigma_{Y_j}). \tag{9.97}$$

Inserting this result into equation 9.93 gives an expression for the distortion that can be used to find the optimal unitary transform:

$$D = kC\ \mathrm{e}^{-2\bar{R}} \prod_{i=0}^{k-1} \sigma_{Y_i}^{\frac{2}{k}}. \tag{9.98}$$

The unitary transform that minimizes equation 9.98 is straightforward to obtain. First we note that the determinant inequality $\prod_i A_{ii} \geq \det(A)$ holds for symmetric positive-definite matrices (see Appendix D) and thus also for the covariance matrix. We then see that

$$\begin{aligned} D &= kC\ \mathrm{e}^{-2\bar{R}} \prod_{i=0}^{k-1} \sigma_{Y_i}^{\frac{2}{k}} && (9.99) \\ &\geq kC\ \mathrm{e}^{-2\bar{R}} \left(\det(\mathrm{E}[Y^k Y^{kH}]) \right)^{\frac{1}{k}}. && (9.100) \end{aligned}$$

The determinant $\det(\mathrm{E}[Y^k Y^{kH}])$ is the same for all unitary transforms U. However, inequality 9.100 becomes an equality only for the case that $\mathrm{E}[Y^k Y^{kH}]$ is *diagonal*, i.e., if the transform is the Karhunen-Loève transform. Thus, the distortion is minimized for the Karhunen-Loève transform.

Example 9.14: Karhunen-Loève transform for a uniform density with finite support

In practice, the Karhunen-Loeve transform is often useful for coding even when the conditions required for optimality are not valid. Let us consider an example where we want to encode independent two-dimensional vectors, each with the density shown in figure 9.14 (density uniform within the "bar"), with two bits per vector. Only scalar quantizers are to be used and the squared-error criterion applies.

We first design optimal 1-bit scalar quantizers for X_1 and X_2. It is easily seen that the optimal scalar quantizers must be symmetric. The positive centroid locations are easily computed as

$$c_i = \underset{y}{\operatorname{argmin}} \int_0^{\sqrt{\frac{1}{8}}} (x-y)^2 dx = \frac{\sqrt{2}}{8}.$$

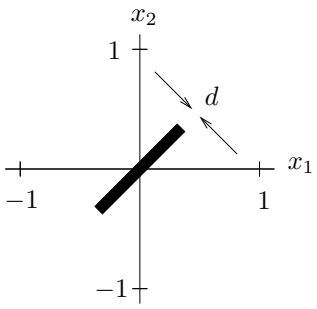

Figure 9.14: Joint density function for example 9.14. The bar is of unity length and $d << 1$.

Thus, the centroids for X_1 and X_2 are $\pm\frac{\sqrt{2}}{8}$. Neglecting bar thickness, the mean squared error is

$$D = \frac{1}{12}\Delta^2 = \frac{2}{12 \cdot 64} = 0.0026$$

per dimension; for the vector it is $2 \cdot 0.026 = 0.0052$. Let $[I_1, I_2]$ denote the index vector. The entropy of the index vector can be computed as follows. The probability of the first and third quadrant is ≈ 0.5, and the probability of the second and fourth quadrants is negligible. Furthermore, the vectors are independent. Thus, the entropy rate of the vectors is $H_\infty([I_1, I_2]) = H_1([I_1, I_2]) \approx 1$. This indicates that our encoding at two bits per vector is very inefficient.

A simple method to improve on this code would be to design a Huffman code for the vectors $[I_1, I_2]$. The four possible vectors label that the data are in first, second, third or fourth quadrant. First and third quandrant probabity are 0.5, second and fourth quadrant probability are very small, ϵ. The design of a Huffman code is trivial. A particular Huffman code is $\{1, 00, 010, 011\}$ for first, third, second, and fourth quadrants. The rate is $1 \cdot 0.5 + 2 \cdot 0.5 + 3 \cdot \epsilon + 3 \cdot \epsilon = 1.5$ bit per vector, which is 0.5 bit above the entropy rate of the vectors, despite the lossless coding.

Next, we use a Karhunen-Loève transform, U, which transforms the two-dimensional vector x^2 (remember that the superscript ^{2} indicates the vector dimensionality) into $y^2 = Ux^2$. The covariance matrix is of the form

$$C = \begin{bmatrix} 1 & 1-\epsilon \\ 1-\epsilon & 1 \end{bmatrix},$$

where ϵ is small. The eigenvalues are ϵ and $2-\epsilon$ and the eigenvectors are $\frac{1}{\sqrt{2}}[1,1]^T$ and $\frac{1}{\sqrt{2}}[1,-1]^T$. The unitary transform U is thus

$$U = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}.$$

We can now design appropriate scalar quantizers for the components of y^2. U results in a y^2 such that y_1 is parallel to the bar and y_2 is orthogonal to the bar. We spend all bits on y_1 and none on y_2. The quantizer for y_1 simply divides the bar in five equal parts. The error is then $\frac{1}{12}\Delta^2 = \frac{1}{12}0.2^2 = 0.0033$. This is one quarter of the error we obtained when quantizing the components of x^2 directly, thus illustrating the power of the Karhunen-Loeve transform. Since all four indices now have equal probability, we have an entropy rate of 2 bits and Huffman coding is unnecessary.

Example 9.15: Constrained-resolution coding using the Karhunen-Loève transform

In this example, we consider constrained-resolution quantization of a two-dimensional distribution exemplified by a data set. We know from equation 7.51 that, at least for high rate, the relation between rate and distortion is of the form required for the Karhunen-Loève transform.

We are given the 1000 training data shown in the left subfigure of figure 9.15. The data are drawn from a Gaussian distribution with covariance matrix

$$C = \begin{bmatrix} 3/2 & 1/2 \\ 1/2 & 3/2 \end{bmatrix}.$$

When we use the discrete generalized Lloyd algorithm of section 8.2.2 to obtain a 6-bit quantizer, we obtain the second subfigure. The average squared error is 0.172 for the training data. If, instead, we use two 3-bit scalar quantizers (optimized separately), then we obtain the quantizer in the third figure. While scalar quantization simplifies the complexity, the performance of this quantizer is significantly worse than the vector quantizer and corresponds to an average squared error of 0.551. If, however, we first apply the Karhunen-Loève transform, and then distribute the rate according to equation 9.96, which corresponds to 4 bits and 2 bits in this case, then we obtain a more reasonable average squared error of 0.241. The corresponding quantizer is shown in the right subfigure. We note that the precise results vary from experiment to experiment with this small number of training data.

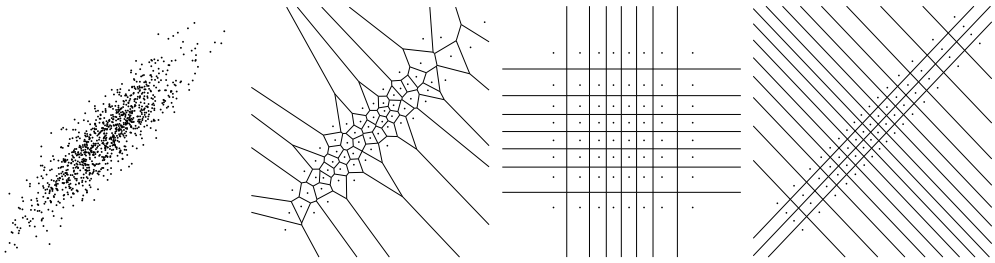

Figure 9.15: Data distribution, vector quantizer, scalar quantizer, and scalar quantizer applied after the Karhunen-Loève transform.

9.4.3 Decorrelation, Energy Concentration, and Coding Gain

The Karhunen-Loève transform leads to a diagonal covariance matrix: assuming that the vector components are zero mean, the Karhunen-Loève transform decorrelates the vector components. We can also interpret the Karhunen-Loève transform as the transform that maximizes a certain energy concentration measure. By taking the logarithm of the multiplication to be optimized in equation 9.98 we obtain the summation

$$\log(\prod_{i=0}^{k-1} \sigma_{Y_i}^2) = \sum_{i=0}^{k-1} \log(\sigma_{Y_i}^2). \tag{9.101}$$

This expression forms a measure of the flatness of the distribution of the variances, since the logarithm is a concave function. Thus, given a sum of variances, equation 9.101 is maximized when all $\sigma^2_{Y_i}$ are equal, i.e., when the distribution is flat. Conversely, minimization of equation 9.101 (or, equivalently, 9.98) can be interpreted as the optimization of a particular energy concentration measure[12].

It is common to evaluate the energy concentration obtained with a unitary transform for a stationary signal with the so-called **coding gain** performance measure. The coding gain is defined as (e.g., [62, 24])

$$G = \frac{\frac{1}{k}\sum_{i=0}^{k-1}\sigma^2_{Y_i}}{\left(\prod_{i=0}^{k-1}\sigma^2_{Y_i}\right)^{\frac{1}{k}}}. \tag{9.102}$$

The arithmetic mean (the numerator), is invariant with a unitary transform U for $Y^k = UX^k$. Thus, it is identical to the sample variance of the original stationary signal. In other words, for a random vector X^k taken from a stationary signal, we have a unity coding gain: $\frac{1}{k}\sum_{i=0}^{k-1}\sigma^2_{X_i} = \left(\prod_{i=0}^{k-1}\sigma^2_{X_i}\right)^{\frac{1}{k}}$ since all σ_{X_i} are equal. If the unitary transform $y = Ux$ results in energy concentration, then the geometric mean $\left(\prod_{i=0}^{k-1}\sigma^2_{Y_i}\right)^{\frac{1}{k}}$ (the denominator) is smaller and the coding gain takes a value greater than unity.

For the case of high-rate quantization, a stationary Gaussian process, and a squared-error distortion measure, the coding gain has a very concrete meaning. From equation 9.99 and the fact that the numerator of the coding gain is unaffected by the unitary transform $Y^k = UX^k$, it follows that, at a given overall rate $k\bar{R}$, the reduction in distortion due to U is inversely proportional to the coding gain. Indeed, if we rewrite equation 9.99 to read

$$\bar{R} = \frac{1}{2}\log\left(\frac{kC\prod_{i=0}^{k-1}\sigma^{\frac{2}{k}}_{Y_i}}{D}\right), \tag{9.103}$$

then we see that, at a given distortion, the reduction in overall rate is proportional to the logarithm of the inverse of the coding gain.

9.4.4 Best Linear Approximation; Coding in a Subspace

In this section, we discuss a coding system where we first transform the k-dimensional vector into a p-dimensional vector with $p < k$, then encode that p-dimensional vector, and then reconstruct an approximation to the original vector. The transform to the p-dimensional subspace is based only on the statistics of the data vectors. We will show that also for this scenario, the Karhunen-Loève is optimal. This means that it is optimal for linear approximation of the vector. To be precise, in this subsection we study the transform coding of a vector where:

1. Only a p-dimensional vector is encoded.
2. The transform to the p-dimensional vector is determined a-priori from the data vector statistics.

[12] Other energy concentration measures are easily defined with different concave functions.

3. The distortion criterion is the squared error criterion.

4. The quantization error is white Gaussian noise.

The latter condition is practical since it allows us to optimize the subspace selection independently of the quantizer properties. The transform to a vector of lower dimensionality prior to encoding results in a reduction of the computational burden of the quantization process. We also note that in section 6.6, which described the rate allocation for Gaussian independent variables, we showed that, at low overall rates, it is often advantageous to allocate a zero rate for some of the individual variables. That result was discussed in the section on reverse water filling, section 6.6.2, which was based on the rate-distortion function for stationary Gaussian processes.

We make only very basic assumptions for our derivation. To simplify our notation, we define $I_{i:j},\ j \geq i$ to be a $k \times k$ matrix with zeros everywhere except for $j - i + 1$ ones along the diagonal in the rows i through j. For example, for k=3 we have

$$I_{1:2} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}. \tag{9.104}$$

We want to find the unitary matrix U that minimizes the mean squared error between the random vector UX^k and p of its rows. Without loss of generality, we assume these to be the first p rows. We define $Z^k = I_{1:p}UX^k$ to be this approximation. The fact that the last $k - p$ rows of z^k are zero implies that $I_{p+1:k}Z^k = 0$. The objective is now to minimize $\mathrm{E}[\|UX^k - Z^k\|^2]$:

$$\begin{aligned} \mathrm{E}[\|UX^k - Z^k\|^2] &= \mathrm{E}[\|I_{p+1:k}UX^k\|^2] \\ &= \mathrm{tr}(I_{p+1:k}U\mathrm{E}[X^kX^{kH}]U^HI_{p+1:k}), \end{aligned} \tag{9.105}$$

where tr is the trace. Let the eigenvalues and eigenvectors of the symmetric autocorrelation matrix $\mathrm{E}[X^kX^{kH}]$ be λ_i and q_i^k, respectively. We note that, since $\mathrm{E}[X^kX^{kH}]$ is symmetric and nonnegative definite, all eigenvalues are nonnegative and that the eigenvectors are orthonormal (assuming normalization). For convenience, we order the eigenvalues: $\lambda_1 \geq \lambda_2 \geq \lambda_3 \cdots$. We then see that

$$\begin{aligned} \mathrm{E}[\|UX^k - Z^k\|^2] &= \mathrm{tr}(I_{p+1:k}U\mathrm{E}[X^kX^{kH}]U^HI_{p+1:k}) \\ &= \mathrm{tr}(I_{p+1:k}U(\sum_{i=1}^{k}\lambda_i q_i^k q_i^{kH})U^HI_{p+1:k}) \\ &= \sum_{i=1}^{k}\lambda_i\|I_{p+1:k}Uq_i^k\|^2. \end{aligned} \tag{9.106}$$

We note that $\|I_{p+1:k}Uq_i^k\|^2$ is the norm of the projection of q_i^k onto a particular $k-p$ dimensional subspace. Since the eigenvectors q_i^k form an orthonormal set, the sum of the norms of their projections onto this subspace must be $k-p$. In equation 9.106 these norms are weighted by the eigenvalues. It is then clear that the expression of equation 9.106 is minimized by selecting U so that the norms $\|I_{p+1:k}Uq_i^k\|^2$ of the largest p

eigenvalues are zero:

$$\begin{aligned} \mathrm{E}[\|UX^k - Z^k\|^2] &= \sum_{i=1}^{k} \lambda_i \|I_{p+1:k} U q_i^k\|^2 \\ &\geq \sum_{i=p+1}^{k} \lambda_i. \end{aligned} \tag{9.107}$$

Equation 9.107 forms a lower bound on the dimensionality-p fit, Z^k, to UX^k over all possible unitary transforms U. By itself, this bound is not useful. However, it is clear that the Karhunen-Loève transform reaches this bound and is, therefore, the optimal transform we are looking for.

Let $U^{(p\times k)}$ denote the transform that provides the coefficients of the components corresponding to the p largest eigenvalues of the Karhunen-Loéve transform. Because the rows of $U^{(p\times k)}$ are orthonormal, the inverse transform is $U^{(p\times k)H}$ and our results imply that the expansion

$$\hat{x}^k = \sum_{i=1}^{p} \langle x^k, U_i^{(p\times k)H} \rangle U_i^{(p\times k)H} \tag{9.108}$$

forms the best p-dimensional linear approximation to x^k in the squared-error sense. $\hat{x}^k$ is the projection of x^k onto the rows of $U^{(p\times k)}$. We have thus shown that, for all k and $0 < p < k$, the optimal p-dimensional subspace for quantization based on the squared error distortion criterion is spanned by the eigenvectors corresponding to the p largest eigenvalues of the covariance matrix. Note that, in this derivation, we did not require the data to be Gaussian.

9.4.5 The Karhunen-Loève Transform in Practical Applications

The Karhunen-Loève transform is the optimal a-priori transform given the statistics and some additional conditions. However, the Karhunen-Loève transform does have a number of practical drawbacks, which differ in importance with the application. Some of the more commonly mentioned drawbacks are:

1. The statistics of the vectors are assumed to be known. In many cases this is not the case.
2. The Karhunen-Loève transform changes with changing statistics. In such cases the transform must be transmitted as side information.
3. The Karhunen-Loève transform is optimal for scalar quantizers only at high rates and for normal-distributed vectors.
4. While the Karhunen-Loève transform adapts to the a-priori statistics, it does not adapt to the input vector, a situation that will be discussed in section 9.5.
5. The Karhunen-Loève transform requires a high computational effort, compared to nonadaptive transforms with a convenient structure such as the fast Fourier transform or wavelet transform.

For the above reasons, the Karhunen-Loève transform is not commonly used in practical applications. However, we can use the Karhunen-Loève transform as another way to motivate the DFT and DCT transforms. (This argument is, of course, not independent from our earlier arguments.) The Szegö theorem (theorem 13) shows that, with increasing dimensionality, the eigenvalue density of the autocorrelation matrix approaches the power spectrum of the signal under certain, reasonable conditions. This means that for stationary signals both the DFT and the DCT transform are equivalent to the Karhunen-Loève transform for infinite vector dimensionality. It is then reasonable to assume that for large vector dimensionality the DFT and the DCT form a good approximation to the Karhunen-Loève transform and that they perform a reasonable decorrelation. It can be argued that for common signals the DCT converges (with increasing dimensionality) quicker to the Karhunen-Loève transform than the DFT [62, 61]. Naturally, the present rate-of-convergence arguments are related to those discussed in section 9.3.2.

9.5 Transforms with A-Posteriori Adaptation

Instead of using a fixed transform, or a transform dependent on the a-priori statistics of the data, a common strategy in recent source coding algorithms is to modify the transform based on the actual input data. A reasonable high-level interpretation is that the adaptation is part of a particular codebook structure. All these methods are inherently nonlinear, since the transform properties depend on the input data. Various flavors of this a-posteriori adaptation exist and we describe a few of them qualitatively below.

9.5.1 A-Posteriori Energy Concentration

Experimental results have shown that methods based on the a-posteriori selection of a subset of the transform coefficients are often efficient at low rates. These methods split the transform coefficients a-posteriori into two classes: those that are insignificant and can be set to zero and those that are to be quantized with a scalar quantizer (e.g., [67, 68]). This two-way classification itself is described with a so-called **significance map**, which must be encoded.

A-posteriori energy concentration is equivalent to an approximation of the data vector by means of a projection onto a subspace. However, in contrast to Karhunen-Loève based linear approximations, the present operation is a **nonlinear approximation**. The nonlinearity is a direct result of the subspace being selected for the specific outcome of the random data vector.

The binary significance value for each coefficient is a Bernoulli process. The rate required to describe this Bernoulli process is very low when the probability of nonzero coefficients is very low (in problem 2 of chapter 6, the rate is given for a Hamming distortion measure). Encoding of the significance map can be performed with a low-complexity vector quantizer or using lossless coding.

Since the energy-concentrating methods involve nonlinear operations, and vector quantization or lossless coding, and since these methods do not rely on Gaussianity, they can perform better in practical applications than various Karhunen-Loève based methods

or than fixed transforms such as the DCT-based approaches. A theoretical relation between rate and distortion can be found [68] for the energy-concentrating procedure under the assumption of optimal encoding of the significance map and the assumption of scalar encoding of the significant coefficients with validity of the high-rate entropy-constrained quantization results. The theory predicts the experimental findings for image data very well. It also shows that the assumptions imply that transforms that lead to more insignificant coefficients provide a better rate versus distortion trade-off.

The transforms used for the energy-concentrating approaches are usually selected by trial and error. It is currently not possible to select these transforms on a theoretical basis. For many signals, experimental evidence suggests that wavelet and wavelet-packet transforms [65, 69] lead to a large fraction of insignificant coefficients and are, thus, good for energy-concentrating approaches.

9.5.2 Matching Pursuit

Matching pursuit can be interpreted as a particular method for a-posteriori concentration of energy in relatively few coefficients. It is based on the expansion of an input vector or signal that we have seen many times within this chapter. In these methods, the expansion coefficients are quantized. In matching pursuit, the expansion vectors or functions are selected from a codebook with nonorthogonal vectors using a greedy approach (that is, one-at-a-time), thus concentrating the energy. In the matching-pursuit context, the codebook is often referred to as a dictionary. The matching pursuit method can also be interpreted as a multi-stage gain-shape quantization procedure.

Let us define the dictionary as $\{s_i\}_{i\in\mathcal{I}}$, where the s_i are unit norm vectors, i.e., $\langle s_i, s_i\rangle = 1$, and $\mathcal{I}$ is the set of indices labeling the dictionary vectors. The (closed) span of the dictionary vectors forms a Hilbert space, which we denote as $\mathcal{H}$. The first step in the matching pursuit algorithm is to project a vector $x \in \mathcal{H}$ onto the dictionary element s_i that best matches the vector and then subtract this projection to form a residual vector. The operation is then repeated for the residual vector, and so on. Naturally, the iterative nature of the method is reasonable only if the dictionary consists of vectors that are nonorthonormal.

The projection onto s_i is equivalent to selecting the optimal gain, λ, for s_i. Let e_{m-1} denote the residual vector after $m-1$ iterations and $e_0 = x$. Recalling that $\mathbb{C}$ is the set of complex numbers, the optimal gain is found as

$$\begin{aligned}
\lambda &= \inf_{\lambda\in\mathbb{C}} \langle e_{m-1} - \lambda s_i, e_{m-1} - \lambda s_i\rangle \\
&= \inf_{\lambda\in\mathbb{C}} \langle e_{m-1}, e_{m-1}\rangle - \lambda^*\langle e_{m-1}, s_i\rangle - \lambda\langle s_i, e_{m-1}\rangle + |\lambda|^2\langle s_i, s_i\rangle \\
&= \inf_{\lambda\in\mathbb{C}} |\lambda|^2 - \lambda^*\langle e_{m-1}, s_i\rangle - \lambda\langle s_i, e_{m-1}\rangle \\
&= \inf_{\lambda\in\mathbb{C}} |\lambda - \langle e_{m-1}, s_i\rangle|^2 \\
&= \langle e_{m-1}, s_i\rangle. \qquad (9.109)
\end{aligned}$$

Thus, at iteration m, the algorithm decomposes e_{m-1} as

$$e_{m-1} = \langle e_{m-1}, s_{i_m}\rangle s_{i_m} + e_m, \qquad (9.110)$$

where s_{i_m} is selected such that

$$|\langle e_{m-1}, s_{i_m}\rangle| = \sup_{i\in\mathcal{I}} |\langle e_{m-1}, s_i\rangle|, \tag{9.111}$$

and where $\langle e_{m-1}, s_{i_m}\rangle$ is the gain. We note that, in the case of real numbers, this gain is identical to that of the gain-shape quantizer of section 8.4.3 for $W = I$ and unit norm shape vectors (cf. equation 8.29). After M iterations, the matching-pursuit approximation to the vector x is

$$\hat{x} = \sum_{m=1}^{M} \langle e_{m-1}, s_{i_m}\rangle s_{i_m}. \tag{9.112}$$

The matching pursuit algorithm generally provides a relatively accurate fit for a low value of M (short expansion). This advantage is lost for high M. For example, for a Hilbert space of dimensionality k, the matching-pursuit expansion generally has finite error after k iterations (except for a fortunate choice of the dictionary).

It is easily seen that the convergence of the matching-pursuit algorithm is exponential with the iteration number. That is, there exists a $\mu > 0$ such that

$$\|e_m\| \leq 2^{-\mu m}\|x\|. \tag{9.113}$$

To prove this, we note that the dictionary $\{s_i\}_{i\in\mathcal{I}}$ is complete in the Hilbert space $\mathcal{H}$, and that, therefore, there exists, at each iteration, an $\alpha > 0$ such that for any $x \in \mathcal{H}$

$$|\langle e_{m-1}, s_{i_m}\rangle| \geq \alpha\|e_{m-1}\|. \tag{9.114}$$

Using equation 9.114 and the fact that e_{m-1} and s_{i_m} are orthogonal, we conclude that

$$\|e_m\| \leq (1-\alpha^2)^{\frac{1}{2}}\|e_{m-1}\|. \tag{9.115}$$

Thus, if we select $2^{-\mu} = (1-\alpha^2)^{\frac{1}{2}} < 1$ we have proved inequality 9.113.

9.5.3 Adaptive Basis Selection

In adaptive basis selection, a basis is selected a-posteriori. In a typical procedure, one has a dictionary of bases and selects one of these bases based on an energy concentration measure [70]. A subset of the coefficients is then coded using scalar quantizers.

A commonly mentioned energy concentration measure is [71]

$$\sum_{i=1}^{k} \mathrm{E}[\|y_i\|^2 \log(|y_i|^2)], \tag{9.116}$$

which is called an **entropy measure**, but other measures are used as well. Note that the entropy measure is only a measure of energy concentration, and is not meant to represent the entropy in an information-theory sense.

Particularly in tree-structured basis dictionaries such as wavelet packets, it is possible to concentrate the energy by using more formal criteria [70]. The algorithms resemble those used for entropy-constrained vector quantizers.

9.6 Problems

1. Prove that by combining N orthonormal bases into a frame, we always get a tight frame.

2. Prove that, if the rows of U form a tight frame, then the corresponding transform maps two vectors that are orthogonal in C^k onto two vectors that are orthogonal in C^m.

3. We have a sequence of two-dimensional Gaussian random vectors, $[X_1, X_2]$. The components of the vectors are unit variance, zero mean, and uncorrelated (both inter- and intra-vector). One kind of scalar quantizer is used for both X_1 and X_2. This scalar quantizer has a step size of 0.01.

   For the second part of the problem, consider the frame of the vectors x_1, x_2, y_1, y_2 displayed in figure 9.16. The vectors are of unit length.

   Show your calculations for the plots.

   (a) Plot the mean distortion (in dB) as a function of the percentage of packets lost (the loss is random, and from 0 to 100 %) when we transmit one packet for each vector $[x_1, x_2]$.

   (b) Plot the mean distortion (in dB) as a function of the percentage of packets lost if we repeat the transmission of each packet two times and three times.

   (c) Prove that the four vectors x_1, x_2, y_1, y_2 form a tight frame.

   (d) For each two-dimensional original vector we transmit two packets: one with $[X_1, X_2]$, and one with $[Y_1, Y_2]$. Plot the mean distortion (in dB) as a function of the percentage of packets lost. (Hint: first compute the distortion if you receive zero, one, and both packets).

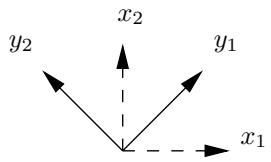

Figure 9.16: The frame of problem 3.

4. Show that, for a tight frame and for frame vectors of unit length, the frame bound A indicates the dimensionality redundancy factor.

5. Let $y^k = Ux^k$, where U is a unitary transform. Show that:

$$\arg\min_U h(f_{Y^k}(.) \| \prod_{i=1}^{k} f_{Y_i}(.)) = \arg\min_U \sum_{i=1}^{k} h_1(Y_i).$$

6. Generally energy concentration increases the performance of scalar quantizers, but this is not always the case. Show a two-dimensional example where energy-concentration by means of a unitary transform reduces the performance of the scalar quantizers.

7. Consider the random vector $X = [X_1, X_2]$ with the probability mass function shown in figure 9.17:

$$f(x_1, x_2) = \frac{1}{2}\delta(x_2 + 1)(u(x_1 + 1) - u(x_1)) + \frac{1}{2}\delta(x_2 - 1)(u(x_1) - u(x_1 - 1)),$$

where $u(t)$ is the unit step function.

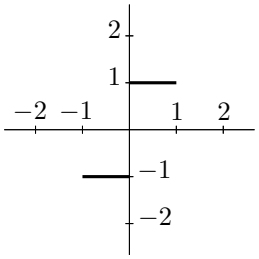

Figure 9.17: Probability-mass function.

The distortion used is the mean squared error. Your goal is to encode the vector with a scalar quantizer using only one bit (i.e., you select one vector component and have two levels for that component). The decoder sets the other component to its mean value.

(a) Compute the covariance matrix associated with the probability-mass function.

(b) You do the coding on X. Select the best component of X for the scalar quantizer. Plot in the figure where the corresponding centroids are. Compute the mean distortion.

(c) Find the Karhunen-Loève transform, U. Plot the probability mass function in the new coordinates.

(d) You do the coding after transformation. Select the best component of $Y = UX$ for the scalar quantizer. Plot in the figure where the corresponding centroids are. Compute the mean distortion.

8. W_1 and W_2 are independent random variables with uniform distributions:

$$f_{w_1} = \begin{cases} \frac{1}{2}, & w_1 \in [-1, 1] \\ 0, & \text{elsewhere} \end{cases}$$

and

$$f_{w_2} = \begin{cases} \frac{1}{4}, & w_2 \in [-2, 2] \\ 0, & \text{elsewhere} \end{cases}.$$

Let $X_1 = W_1 + W_2$, $X_2 = W_1 - W_2$, and $X = [X_1 X_2]$.

(a) Using 1000 random data points, make a scatter plot (X_1 along the horizontal axis, X_2 along the vertical axis) for X.

(b) Find the index entropy for uniform scalar quantizers as a function of the step size Δ for both X_1 and X_2, when Δ is small.

(c) Evaluate the covariance matrix of the random vector X.

(d) Determine the Karhunen-Loève transform, U, for the random vector X.

(e) Using 1000 random data points, make a scatter plot of $Y = UX$.

(f) Find the index entropy for uniform scalar quantizers as a function of the step size Δ for both Y_1 and Y_2, when Δ is small.

(g) Compare the rate-distortion relations for Y and X for scalar quantizers. Explain.

9. Consider a zero-mean Gaussian-distributed random vector with covariance matrix

$$\mathrm{E}[X^k X^{kT}] = \begin{bmatrix} 4 & \sqrt{2} \\ \sqrt{2} & 2 \end{bmatrix}.$$

The distortion is mean squared error, summed over the vector components. Your goal is to encode only one vector component, using scalar quantization. You want to see if a transform is beneficial. (Hint: equation 7.33 is helpful.)

(a) You attempt the coding prior to transformation. Select the component for coding that results in the lowest distortion, and give the high-resolution relation between overall distortion and overall rate in this case.

(b) Find the Karhunen-Loève transform.

(c) You do the coding after transformation. Select the component for coding that results in the lowest distortion, and give the high-resolution relation between overall distortion and overall rate in this case.

(d) Provide qualitative reasoning what strategy is best.

10. Consider a zero-mean Gaussian variable X_1 with variance $\sigma^2_{X_1} = 1$. A second variable X_2 is defined as $X_2 = X_1 + W$ where W is also a zero-mean Gaussian variable with variance $\sigma^2_W = \frac{1}{100}$. X and W are independent. The distortion measure is the mean squared error.

(a) Determine the covariance matrix of the vector $[X_1, X_2]$.

(b) Determine the rate-distortion function for as far as you can do this analytically.

(c) You want to employ high-rate scalar entropy-constrained quantization. Obtain the relation between rate and distortion for high rate when scalar quantization is applied to X_1 and X_2 separately.

(d) Find the transformations that maximize the coding efficiency of the high-rate entropy-constrained quantizers. Then obtain the resulting relation between rate and distortion (again for high rate).

(e) Compare all three rate-distortion relations. Explicitly list all reasons for the differences.

(f) In a packet-network, you sometimes loose the index of either one of the two variables because the corresponding data packets of the network do not arrive in time. Both for the case without the transform and the case with the transform, propose good decoding strategies. Discuss which method of encoding/decoding you would use for which loss-rate ranges.

11. Derive the equivalent of equation 9.50 for the continuous-time Fourier transform.

12. The four bands of a subband coder form a stationary vector process. The vector components are to be quantized with scalar quantizers. We describe a vector sample of the process with the random vector $X = [X_1, X_2, X_3, X_4]$. Its components are independent and have the probability densities given in figure 9.18. We use as distortion criterion the mean squared error and assume that the high-resolution approximation is valid.

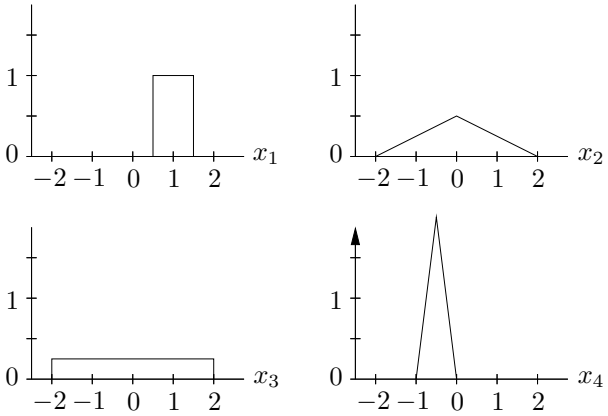

Figure 9.18: Densities of the four vector components.

(a) Compute the differential entropies of the four components.

(b) Set a given value for the sum of the four index entropies, $H_T = \sum_{m=1}^{4} H(I_m)$. Using the method of Lagrange multipliers, derive an algorithm to determine the optimal index entropy distribution for the four components.

(c) Express the optimal entropy distribution for very high H_T as a function of the overall entropy H_T and the differential entropies of the components.

(d) Give an expression for overall distortion at very high H_T.

(e) Explain if you expect to obtain further improvement of coding performance by performing the Karhunen-Loève transform on the vector X.

13. Prove that $\mathrm{tr}(UAU^H) = \mathrm{tr}(A)$ (tr is the trace) for unitary U.

14. Prove that the DCT-II transform is unitary.

15. Consider a first-order Gauss-Markov process (pass white Gaussian noise through a single-pole filter) with $\mathrm{E}[X_i X_{i+1}] = 0.9$ and $[X_i^2] = 1$.

(a) Determine the autocorrelation function.

(b) Make a plot of the correlation matrix for X^6, plotting the 36 squares with color or grey scale indicating the value.

(c) Determine the Karhunen-Loève transform.

(d) Plot the correlation matrix that the FFT diagonalizes.

(e) Plot the correlation matrix that the DCT-II diagonalizes.

(f) Redo all your plots for dimensionality 16.

(g) Discuss whether DCT-II or DFT is a better approximation for this signal.

16. Using the DCT-IV as a starting point, find different MLT bases that have increasingly large bodies. That is for a given l find a series of MLT basis functions with increasing k.

17. (a) Prove directly that the MLT given in example 9.12 satisfies the unitarity criterion 9.72.

    (b) Plot the frequency responses of an 8-channel filter bank using a blockwise DCT-II, with $k = 16$.

    (c) Plot the frequency responses of an 8-channel DCT-IV based MLT filter bank with $k = 16$ and $l = 8$ and a sine window.

18. Consider a signal-processing system, where we first multiply the signal by window functions of length M and overlap $M/2$. The window functions are complementary in that they sum to unity, as shown in an example in figure 9.19. Each windowed signal segment is then transformed with a DFT of dimensionality M. The sequence of DFT coefficient sets is often referred to as the short-time Fourier transform (STFT). We consider the effect of modification of the STFT coefficients on the reconstructed signal. (The modification can be a quantization, or another modification, such as the modification used in speech enhancement.) In the following, we confirm with a practical example that the Gabor transform is a better choice.

    (a) For an arbitrary window shape, find an inverse transform that gives perfect reconstruction.

    (b) Assume that the coding step adds Gaussian white noise to the transform coefficients. For each transformed segment the signal-to-noise ratio of this noise is 10 dB, what is the signal-to-noise ratio of the reconstructed signal? Show that this result does not depend on the window shape.

    (c) Now assume that we use the Gabor transform instead (recall the windows are then *power* complementary). If the signal-to-noise ratio of this noise is 10 dB, what is the signal-to-noise ratio of the reconstructed signal? Show that this result does not depend on the window shape.

    (d) Argue that, for quantization in the transform domain of an audio signal, the reconstruction of a steady-state sinusoid (a tone) would be more accurate with the Gabor transform than with the STFT. Explain why this is consistent with your above results (hint: this is not trivial).

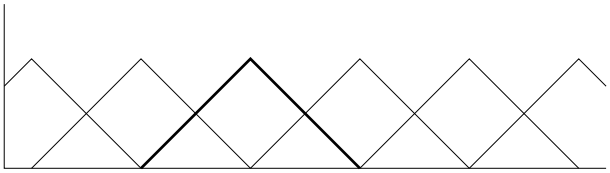

Figure 9.19: The windows of problem 18.

19. We consider a signal segment of 64 samples, labeled 0 through 63. The signal is of the form

$$x(i) = \begin{cases} i - 10, & i = 10, \cdots 40, \\ 0, & \text{elsewhere.} \end{cases}$$

The DFT basis vectors (dimensionality 64) form a dictionary consisting of 64 complex exponentials. We denote this dictionary as Γ_a. We create an additional dictionary, Γ_b, with 400 complex exponentials, also evenly spaced in frequency. Note that the vectors in Γ_a are orthonormal, whereas those of Γ_b are not.

(a) Exploiting the fast Fourier transform, approximate the signal with the 10 basis vectors of Γ_a with the largest coefficient norm. Plot the original signal segment and your approximation.

(b) Exploiting the fast Fourier transform, plot the logarithmic distortion as a function of the number of basis vectors from Γ_a, assuming you use the subset of basis vectors that have the largest coefficient norm.

(c) Using matching pursuit, approximate the signal with an expansion into 10 functions from the dictionary Γ_b. Plot the original signal segment and your approximation.

(d) Using matching pursuit, plot the logarithmic distortion as a function of the number of iterations. Compare to the DFT results.

20. Decorrelating transforms generally make squared-error scalar quantizers more efficient. In this problem we study an, admittedly somewhat artificial, example that shows that this is not necessarily so. Consider the density of figure 9.20, which is uniform over the black area. The black area has probability 0.5, and so has the single point in the third quadrant. This means that the random vector with this density is zero mean. Consider *constrained-resolution quantization* based on the squared error criterion.

(a) If possible without making any calculations at all, write down a unitary transform that decorrelates the density. Provide both a plot of the density after your transform and a reasoning for how you found the transform. Is your transform unique?

(b) Consider the case *without* a decorrelating transform. You encode the signal with two scalar, resolution-constrained 4-bit quantizers. Select the optimal constrained resolution quantizers (you may guess) and provide a solid reasoning that these quantizers are indeed (locally) optimal (hint: think of training).

(c) Provide the mean distortion for the resulting 4-bit quantization scheme without the transform.

(d) Consider the case *with* a decorrelating transform. Again, you encode the signal with two scalar, resolution-constrained 4-bit quantizers. Describe how to design *reasonably* optimal quantizer (*no need to calculate explicit quantizers and you can ignore the singular point in the design of the cells; don't waste time on refinement*).

(e) Provide a reasonably accurate *estimate* of the mean distortion for the resulting $2\times$ 4-bit quantization scheme *with* the transform. *To simplify your computations, ignore the effect of the single point (except for its effect on the overall probability).*

21. Consider an N-sample stationary and ergodic sequence with alphabet $\{-1, 1\}$. N is very large. The dependencies between samples in the sequence decrease rapidly with increasing separation. (However, except where explicitly stated, the sequence is not assumed to be iid.)

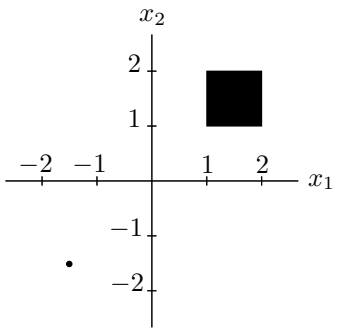

Figure 9.20: The density function of problem 20. The square has uniform density with total probability 0.5. At [-3/2,-3/2] is a singular point that also has probability 0.5. The random vector corresponding to the density has mean [0,0].

(a) Find a transform (motivate or give a derivation!) that approximates the Karhunen-Loeve transform for *any* stationary sequence for N approaching infinity. (Note: the alphabet is of no significance for this part of the problem.)

(b) Assuming that the sequence is iid, provide the distribution of the sequence after this generic Karhunen-Loeve transform is applied. Motivate or give a derivation and provide all parameters for the distribution.

(c) Assuming that the sequence is iid, you are to design a minimum squared error constrained-entropy scalar quantizer for the generic Karhunen-Loeve transformed sequence. Assuming that high-rate theory holds, what rate do you require for a mean error of 0.1? How does this compare to the maximum entropy rate of the signal?

(d) Assuming that all possible realizations of a sequence of N samples with the alphabet $\{-1, 1\}$ are equally likely, derive how many -1 and how many 1 are most likely to occur. Relate the result to the entropy.

22. Consider a zero-mean Gaussian random vector $X^2 = [X_1, X_2]^T$ and the squared-

error criterion. The vector has a covariance matrix

$$R_X = \begin{bmatrix} 1 & 0 \\ 0 & 100 \end{bmatrix}.$$

You want to transmit the data X^2 at low distortion over a network that occasionally loses the encoded quantization indices. Instead of losing an index describing either X_1 or X_2 you decide that is better to combine them and reduce the chance to lose all information about X_1 or X_2. Thus, you compute $Y^2 = BX^2$ and encode Y^2, where

$$B = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix}.$$

(a) Find the mutual information between X_1 and X_2.

(b) Find the mutual information between Y_1 and Y_2.

(c) Determine the rate-distortion relation for encoding the vector X^2 with constrained-entropy scalar quantizers operating on its two components. Use the assumption of low distortion for the rate distribution between the components.

(d) Making the same low distortion assumptions, determine the rate-distortion relation for encoding the vector Y^2 with constrained-entropy scalar quantizers operating on its components. (The scalar quantizers do not exploit in any way the correlations between the components.)

(e) Explain why you expect the rate results above.

(f) Consider the case that the mean distortion is 0.1 if both indices are received. What is the mean distortion if only the index for X_1 or the index for X_2 arrives, but not both?

(g) Consider the case that the mean distortion is 0.1 if both indices are received. Design a good decoder that provides an estimate of X^2 and give the mean distortion if you know that only the index Y_1 is received.

(h) In light of your results conclude when dependency between components is beneficial.

23. Consider a random process X_i and the squared error criterion. The samples of X_i are iid. The marginal distribution of each sample is boxcar shaped (rectangular):

$$f_X(x) = \begin{cases} 1, & x \in [0,1) \\ 0, & \text{elsewhere} \end{cases}$$

(a) Find the Shannon lower bound for X_i.

(b) Is the Shannon lower bound tight? Explain!

Consider a k-point DFT of a block of k sequential data of the process. We write this as $Y^k = U_{\mathcal{F}} X^k$, where $U_{\mathcal{F}}$ is the DFT matrix. We consider a DFT transform that conserves the variance of vectors.

(c) Provide a formula or definition of the density that approximates the marginal densities of the real and imaginary parts of the components of Y^k for sufficiently large k. Hint: note that the energy was spread over only k real components.

(d) Using the fore-mentioned density, find the Shannon lower bound for the real and imaginary parts of Y_i. (This represents the case of separate quantizers for the real and imaginary components.)

(e) Is the Shannon lower bound on the real and imaginary parts of Y_i tight?

(f) You want to compare the performance of scalar quantizers operating on the components X^k (a real vector) versus scalar quantizers operating on the components of Y^k (a complex vector). You assume the mean distortion over all components of the vectors is uniform. Compute the sum of the Shannon lower bound rates (SLBs) over the components X^k and Y^k (this is trivial). First find the sum of the SLBs for the k real components of X^k and then for the $2k$ real and imaginary components of Y^k. Then you take into account the symmetry of the components of Y^k and sum the SLBs for only k appropriate quantizers of components of Y^k. You now get a paradox.

i. What assumption is made when you assume the distortion is constant over the components?

ii. Do the sums over k and $2k$ components form the SLB for vectors X^k and Y^k? How about the sum over k components of Y^k?

iii. Explain what the paradox is and resolve it (this is not simple!).

10

Linear Prediction

10.1 Introduction

Prediction is commonly used in source coding. Particularly in speech coding, linear prediction is ubiquitous: the large majority of speech coding standards defined between 1980 and 2000 are based on linear prediction. The performance obtained by these coders clearly shows the practical advantages of the method. In this chapter, we describe linear prediction methods and analyze their performance.

In chapter 7, we saw that scalar quantizers and low-dimensional vector quantizers are (given certain conditions) most efficient for processes with zero redundancy (redundancy is defined in 3.10). This suggests that it is advantageous to reduce the redundancy of the process by some operation prior to quantization with such quantizers and this was confirmed in chapter 9 where we used unitary transforms for this purpose. In this chapter, we see that linear prediction (or prediction in general) can also be used to remove redundancy from a signal prior to quantization.

In signal processing the term "prediction" refers to the operation of predicting each sample of a process from previous samples. The difference between the predicted value and the actual value of a sample forms the **prediction error** or **residual**. In practice, the computation of the prediction residual signal is a simple filtering operation with the **prediction-error** or **analysis filter**. The predictor (and, thus, the prediction error filter) is usually optimized by minimizing the variance of the residual signal. For a Gaussian signal, this minimization corresponds to minimizing the first-order differential entropy of the residual signal (see equation 3.22) and this, in turn, ensures a minimization of the quantization-index entropy of the entropy-constrained scalar quantizer (see equation 7.34). Thus, assuming entropy-constrained scalar quantization (with an entropy coder of sufficient quality) and a Gaussian signal, the linear prediction operation results in a reduction of bit rate. In the more general case, a direct association between bit rate reduction and linear prediction is more difficult to make. However, in practice we usually get significant redundancy removal and, thus, a reduction in rate.

Before the motivation for the usage of linear prediction in source coding is complete, we have to pass another obstacle: the fact that the distortion measure is not invariant

with the filter operation needed to obtain the prediction residual. In motivating the use of unitary transforms for coding in chapter 9, we only considered the squared-error criterion and then exploited the fact that this distortion criterion is invariant with such transforms. Unfortunately, the squared-error criterion is not invariant with filtering with the analysis filter. Let us consider the method most obvious at first sight (illustrated in figure 10.1): filtering with the analysis filter, scalar quantization with the straight squared-error criterion, and then filtering with the **synthesis** filter, which is a simple inverse of the analysis filter[1]. This method, generally referred to as **open-loop prediction** does, in general, not increase coding efficiency significantly. This is easily seen for the Gaussian case and high-rate entropy-constrained scalar quantization. In this case the rate depends on the signal-to-noise ratio (SNR) only (as shown by equation 7.34). For white quantization noise, the synthesis filter does not change the SNR, and so quantization in the original domain would have been just as effective.

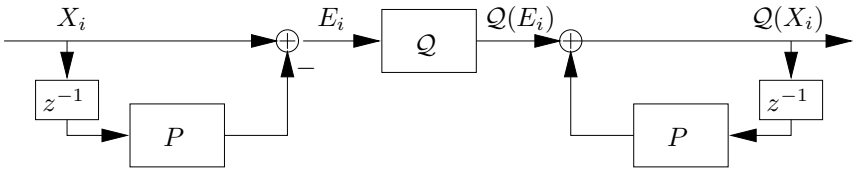

Figure 10.1: Open-loop prediction. P is the prediction operator and $\mathcal{Q}$ the quantizer operator. The structure left of the quantizer forms the analysis filter and the structure to the right the synthesis filter.

The simplest method to make prediction an effective method for source coding is to ensure that the distortion criterion used by the quantizer operating on the residual signal is identical to the distortion criterion used by a quantizer operating on the original signal. At first sight this might seem to result in a complicated distortion criterion involving weighting, but it can in fact be attained with only a small change in the architecture of the coding structure. Instead of predicting from the original signal samples, we predict the next original signal sample from the previous *reconstructed* signal samples. In this case, the distortion of the reconstructed residual signal is identical to the distortion of the reconstructed original signal. This method is generally referred to as **closed-loop prediction** and is illustrated in figure 10.2. Closed-loop prediction is a very effective tool for source coding and is commonly used.

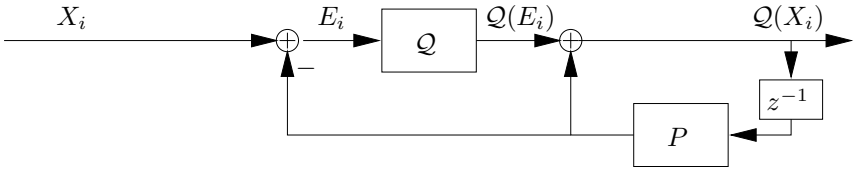

Figure 10.2: A closed-loop predictor. The prediction structures in encoder and decoder are identical and plotted only once.

A more thorough motivation for closed-loop prediction is given in section 10.4. It is

[1]We thus ignore the results of section 6.6.2 which show that, at least in the Gaussian case, this is not correct. However, the fact that the filters are each others inverse is of no significant consequence in the present discussion.

also be shown that a more complicated coding structure, where the distortion criterion includes weighting, is often beneficial. In this manner, it is possible to generalize the closed-loop prediction procedure and obtain advantages in coding efficiency from both vector quantization and the prediction operation. The resulting procedure is called linear-prediction based analysis-by-synthesis. In the terminology of section 7.5, a prediction-based analysis-by-synthesis source coders exploits the correlation-related (linear) part of the memory advantage through the predictor and the remainder (non-linear) part of the memory advantage, the space filling, and shape advantages through vector quantization.

In the remainder of this chapter, we first discuss the fundamentals of linear prediction assuming known data statistics. We then discuss linear prediction for given data (this could be called a-posteriori prediction) and thereafter how to exploit prediction in practical coder architectures. We end the chapter with sections on the quantization and computation of the predictor coefficients.

10.2 Fundamentals of Linear Prediction

10.2.1 Mean Squared-Error Optimal Linear Prediction

Consider a wide-sense stationary process X_i. We make a linear estimate of X_i based on the p previous samples of the process, $X_{i-p}, \cdots, X_{i-1}$. The estimate $\hat{X}_i$ can thus be expressed as

$$\hat{X}_i^{(p)} = -\sum_{j=1}^{p} a_j^{(p)} X_{i-j}, \tag{10.1}$$

where p is the order of linear prediction analysis and the $a_j^{(p)}$ are the linear prediction coefficients. (For speech sampled at 8 kHz, the value of p is typically 10). We let $E_i^{(p)}$ denote the mean squared error of the p'th order predictor:

$$\begin{aligned} E_i^{(p)} &= X_i - \hat{X}_i^{(p)} \\ &= X_i + \sum_{j=1}^{p} a_j^{(p)} X_{i-j} \\ &= a^{(p)T} X_{i:i-p} \end{aligned} \tag{10.2}$$

where $a^{(p)T} = [1, a_1^{(p)}, \cdots, a_p^{(p)}]$ and $X_{i:i-p}^T = [X_i, \cdots, X_{i-p}]$.

The linear prediction operation is equivalent to an all-zero filter, which is illustrated in figure 10.3. The inverse of this filter, where we recreate the signal from the prediction-error signal is called an **autoregressive** or AR process. Thus, estimation of the linear prediction coefficients can be interpreted as **identification** of the AR system.

Equation 10.2 is commonly written in the z-transform domain notation[2]:

$$E(z) = A^{(p)}(z) X(z), \tag{10.3}$$

[2]The z-transform of a discrete process X_i is denoted as $X(z)$. Our discrete-time Fourier transform notation for a process exploits this notation by replacing z with $e^{j\omega}$, resulting in $X(e^{j\omega})$.

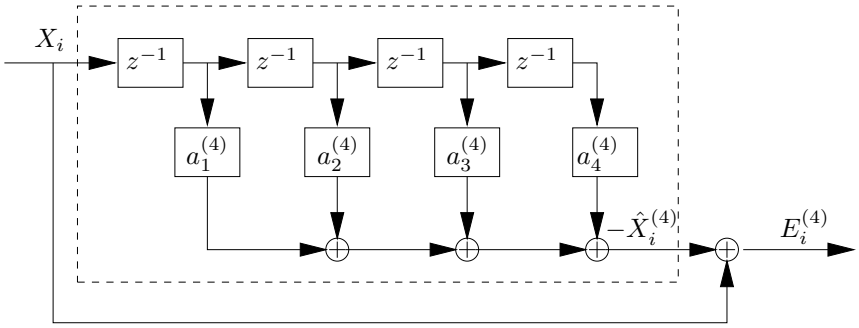

Figure 10.3: A fourth-order linear predictor (in box) and the associated relation between the input sequence X_i, the prediction estimate $\hat{X}_i$ and the prediction error $E_i^{(4)}$.

where $X(z)$ and $E(z)$ are the z transforms of the original process, X_i, and prediction-error process, E_i, respectively, and where

$$A^{(p)}(z) = 1 + \sum_{j=1}^{p} a_j^{(p)} z^{-j}. \tag{10.4}$$

defines the prediction-error filter.

We define the optimal linear predictor as the linear predictor which minimizes the variance of the prediction error. We can find a set of equations determining this predictor using the Lagrange multiplier method. The variance of the prediction error is

$$\begin{aligned} \sigma_p^2 \equiv \mathrm{E}[E_i^{(p)2}] &= \mathrm{E}[(a^{(p)T} X_{i:i-p})^2] \\ &= a^{(p)T} \mathrm{E}[X_{i:i-p} X_{i:i-p}^T] a^{(p)} \\ &= a^{(p)T} R^{(p)} a^{(p)}, \end{aligned} \tag{10.5}$$

where $R^{(p)}$ is the p'th order autocorrelation (auto-covariance for zero-mean processes) matrix of the process X_i. The $(p+1) \times (p+1)$ matrix $R^{(p)}$ is **Toeplitz**, that is, they are constant along the diagonals.

Equation 10.5 must be optimized under the constraint that $a_0 = 1$, i.e., under the constraint that $A(z)$ is a **monic polynomial**. Let $b = [1, 0, \cdots, 0]^T$, then the constraint can be written as

$$\mathbf{b}^T a^{(p)} = 1 \tag{10.6}$$

With λ as the Lagrange multiplier, we write the augmented criterion as

$$\eta = a^{(p)T} R^{(p)} a^{(p)} - \lambda \mathbf{b}^T a^{(p)}. \tag{10.7}$$

Differentiating and setting the result to zero leads to

$$0 = R^{(p)} a^{(p)} - \lambda \mathbf{b}. \tag{10.8}$$

or

$$R^{(p)} a^{(p)} = \begin{bmatrix} \lambda \\ 0^p \end{bmatrix}. \tag{10.9}$$

Much can be learned from equation 10.9. First, we analyze the bottom p rows of equation 10.9 and then the top row. The bottom p rows imply an important **orthogonality property of optimal linear predictors**:

$$\begin{aligned} 0 &= \mathrm{E}[X_m X_{i:i-p}^T] a^{(p)} \\ &= \mathrm{E}[X_m E_i^{(p)}], \qquad i-p \le m < i, \end{aligned} \tag{10.10}$$

which states that the original-process samples $X_{i-p}, \cdots, X_{i-1}$ (on which the prediction is based) each are uncorrelated with the prediction-residual sample E_i.

Furthermore, by multiplying both sides of equation 10.9 on the left with $a^{(p)T}$ and comparing to equation 10.5, we see that $\lambda = \sigma_p^2$. Thus, we can write $p+1$ equations specifying the optimal linear predictor, $a^{(p)}$ and the prediction error variance σ_p^2:

$$R^{(p)} a^{(p)} = \begin{bmatrix} \sigma_p^2 \\ 0^p \end{bmatrix}. \tag{10.11}$$

The lower p rows of equation 10.11 are known as **Yule-Walker equations**. These equations also fall into classes called **normal equations** and **Wiener-Hopf equations**. The Wiener-Hopf equations are obtained when estimating variables in a more general situation, and cover the cases of both system identification and prediction. The normal equations are also used for other sets of equations specifying predictors; an example of this situation occurs in section 10.3. All $p+1$ equations 10.11 together are sometimes called the **extended Wiener-Hopf equations**.

The $p+1$ extended Wiener-Hopf equations can be solved for the predictor coefficients $a_1^{(p)}, \cdots, a_p^{(p)}$ and the prediction error variance σ_p^2. By means of inequalities it can be seen that the extremum defined by the Wiener-Hopf equation minimizes the prediction error variance σ_p^2 (see problem 2).

10.2.2 Maximum Likelihood Spectral Estimation

The Yule-Walker equations were derived by minimizing the mean squared prediction error. These equations can also be obtained using the **maximum-likelihood method** [72, 73]. This approach starts with the assumption that a process X_i is the output of an all-pole filter $1/A^{(p)}(z)$ that has as input an iid Gaussian process, E_i. Let us consider a finite random sub-sequence with k samples, X^k. The Gaussian multi-variate density of X^k is

$$f_{X^k}(x^k) = \frac{1}{\sqrt{(2\pi)^k \det(R)}} \mathrm{e}^{-\frac{1}{2} x^{kT} R^{-1} x^k}, \tag{10.12}$$

where the covariance matrix R is determined by the autoregressive signal model (i.e., by the predictor coefficients). (In example 4.2 the case without the model assumption is treated.) Our objective is to find the model (the predictor coefficients) that maximizes the likelihood of a given data vector x^k. It is convenient to maximize the logarithm of the likelihood,

$$\log(f_{X^k}(x^k)) = -\frac{k}{2}\log(2\pi) - \frac{1}{2}\log(\det(R)) - \frac{1}{2} x^{kT} R^{-1} x^k. \tag{10.13}$$

If we assume that k is large so that it is reasonable to ignore the zero-input response of the filter, we note that the random linear-prediction residual vector E^k is related to a

random data vector by

$$X^k = BE^k, \tag{10.14}$$

where B is a $k \times k$ lower triangular Toeplitz matrix with as first column the impulse response of the autoregressive model $1/A^{(p)}(z)$. We note that the diagonal of the matrix B consists of ones. Exploiting that E_i is an iid process with an as-yet unknown variance σ_p^2, we have that

$$\begin{aligned} \det(R) &= \det(\mathrm{E}[X^k X^{kT}]) \\ &= \det(\mathrm{E}[BE^k E^{kT} B^T]) \\ &= \det(B)\det(\mathrm{E}[E^k E^{kT}])\det(B) \\ &= \sigma_p^{2k}\det(B)^2 \\ &= \sigma_p^{2k}. \end{aligned} \tag{10.15}$$

Equation 10.14 approximates a filtering operation with the infinite-impulse response filter $1/A^{(p)}(z)$. The inverse of the filter is $A^{(p)}(z)$ with impulse response $[1, a^{(p)}{}_1, \cdots a^{(p)}{}_p]^T$. Thus, it is seen that the matrix B can be inverted, at least approximately, by a matrix H which has as first column the impulse response of the all-zero filter $A(z)$:

$$\begin{aligned} R^{-1} &= \mathrm{E}[BE^k E^{kT} B^T]^{-1} \\ &= \sigma_p^{-2} B^{-T} B^{-1} \\ &= \sigma_p^{-2} H^T H. \end{aligned} \tag{10.16}$$

Let $\mathcal{X}$ denote a $k \times k$ lower-triangular Toeplitz data matrix with as first column the data vector x^k. Using the results 10.15 and 10.16, we can then rewrite equation 10.13 as follows:

$$\begin{aligned} \log(f_{X^k}(x)) &\approx -\frac{k}{2}\log(2\pi) - \frac{k}{2}\log(\sigma_p^2) - \frac{1}{2\sigma_p^2} x^{kT} H^T H x^k \\ &= -\frac{k}{2}\log(2\pi) - \frac{k}{2}\log(\sigma_p^2) - \frac{1}{2\sigma_p^2} a^{(p)T} \mathcal{X}^T \mathcal{X} a^{(p)T} \\ &= -\frac{k}{2}\log(2\pi) - \frac{k}{2}\log(\sigma_p^2) - \frac{k}{2\sigma_p^2} a^{(p)T} C_x a^{(p)T}, \end{aligned} \tag{10.17}$$

where $C_x = \frac{1}{k}\mathcal{X}^T\mathcal{X}$ is a data-covariance matrix. The criterion of equation 10.17 can be made to look nice by multiplying by $\frac{2}{k}$, removing the first term, by changing the sign, and by subtracting $\log(\sigma_\infty^2)$ and unity. We then obtain the **Itakura-Saito distortion criterion**:

$$D_{IS} = \log(\frac{\sigma_p^2}{\sigma_\infty^2}) + \frac{a^{(p)T} C_x a^{(p)}}{\sigma_p^2} - 1. \tag{10.18}$$

It is easy to see that the Itakura-Saito criterion is zero when the model description is entirely accurate.

The autoregressive model is identified by minimizing the Itakura-Saito criterion for the predictor coefficients $a_1^{(p)}, \cdots, a_p^{(p)}$ and the variance σ_p^2. It is clear from equation 10.18 that for the predictor coefficients, the criterion is similar to that of equation 10.5. We thus conclude that the maximum-likelihood method for optimizing the predictor

parameters under the assumption of Gaussianity leads to the Yule-Walker equations, just like minimizing the prediction-error signal variance.

If the data-covariance matrix is forced to be Toeplitz and written as $\hat{R}_X$ then, using the discrete-time Fourier transform, the Itakura-Saito criterion can also be written as

$$D_{IS} = \log(\frac{\sigma_p^2}{\sigma_\infty^2}) + \frac{1}{2\pi}\int_{-\pi}^{\pi} \hat{R}_X(\mathrm{e}^{j\omega})\frac{|\mathcal{A}^{(p)}(\mathrm{e}^{j\omega})|^2}{\sigma_p^2}d\omega - 1. \tag{10.19}$$

We note that if the power-spectral estimate $\hat{R}_X(\mathrm{e}^{j\omega})$ and the model spectrum $\frac{|\mathcal{A}^{(p)}(\mathrm{e}^{j\omega})|^2}{\sigma_p^2}$ are identical, then the Itakura-Saito distortion vanishes. This, together with the non-negativeness of the criterion (problem 4) illustrates the spectral estimation properties of the criterion.

10.2.3 Spectral Domain Interpretation

Let us consider the autocorrelation $\mathrm{E}[E_i^{(p)}E_{i+n}^{(p)}]$ of the prediction-error process, for a p'th order predictor. This correlation sequence is the inverse discrete-time Fourier transform of the power spectrum $R_{E^{(p)}}(\mathrm{e}^{j\omega})$ of the linear prediction error $E_i^{(p)}$:

$$\mathrm{E}[E_i^{(p)}E_{i+n}^{(p)}] = \frac{1}{2\pi}\int_{-\pi}^{\pi} R_{E^{(p)}}(\mathrm{e}^{j\omega})\mathrm{e}^{j\omega}d\omega. \tag{10.20}$$

Furthermore, we denote the power spectrum of the process X_i by $R_X(\mathrm{e}^{j\omega})$. Then we have

$$\mathrm{E}[E_i^{(p)}E_{i+n}^{(p)}] = \frac{1}{2\pi}\int_{-\pi}^{\pi} R_X(\mathrm{e}^{j\omega})|A^{(p)}(\mathrm{e}^{j\omega})|^2\mathrm{e}^{j\omega}d\omega, \tag{10.21}$$

where $A^{(p)}(\mathrm{e}^{j\omega})$ is the transfer function of the prediction filter. For $n = 0$, we can write

$$\sigma_p^2 = \mathrm{E}[E_i^{(p)2}] = \frac{1}{2\pi}\int_{-\pi}^{\pi} \frac{R_X(\mathrm{e}^{j\omega})}{1/|A^{(p)}(\mathrm{e}^{j\omega})|^2}d\omega. \tag{10.22}$$

Equation 10.22 indicates that minimization of the prediction error variance is equivalent to minimizing the integral of the ratio of the original signal power spectrum and the power spectrum $1/|A^{(p)}(\mathrm{e}^{j\omega})|^2$ corresponding to the synthesis transfer function. In both cases, the minimization happens under the constraint that $a_0^{(p)} = 1$.

It is useful to consider the meaning of equation 10.22 in some more detail. The criterion turns out to share two important properties with the human auditory system that make it very suitable for spectral estimation of speech. First, it is largely the envelope of the speech power spectrum $R_X(\mathrm{e}^{j\omega})$ that determines the linguistic meaning of speech. Thus, it is important that linear-prediction can separate the spectral envelope from other spectral structure. Second, the spectral envelope of the speech signal has a large dynamic range. Often, the high-frequency regions (> 3 KHz) have peak magnitudes 20 or even 30 dB below those of the lower frequency regions (< 1.5 KHz). Despite this large dynamic range, the human auditory system is capable of distinguishing features across all frequency regions.

We start with explaining the suitability of linear prediction for spectral envelope estimation. Let us consider the case where the order p is relatively small (e.g., $p = 10$).

The transfer function $|1/A^{(p)}(\mathrm{e}^{j\omega})|^2$ is then mostly relatively smooth, i.e., it can describe some "macro-structure" of $R_X(\mathrm{e}^{j\omega})$, but not its "micro-structure" (often called **fine-structure**). Thus, the transfer function $|1/A(\mathrm{e}^{j\omega})|^2$ can be approximated as locally constant when evaluating the relative importance of local peaks and valleys in the spectrum. The criterion of equation 10.22 emphasizes the local peaks, and, as a result, $|1/A(\mathrm{e}^{j\omega})|^2$ locally approximates the shape of the envelope of the power spectrum $R_X(\mathrm{e}^{j\omega})$. This is sometimes referred to as the **local property** of the linear-prediction based spectral estimate. The local valleys between spectral peaks (such as those between harmonics in voiced speech) are often governed by extraneous noise, making this a very desirable property in spectral estimation.

Example 10.1: The local property

Figure 10.4 provides a simple demonstration of the local property. Let $G_i^{(c)}$ be a random signal with power spectrum $R_{G^{(c)}}(\mathrm{e}^{j\omega})$. This power spectrum consists of segments of two all-pole spectra with 6 poles each (the poles come in conjugate pairs). Let $R_{G^{(u)}}(\mathrm{e}^{j\omega})$, $R_{G^{(l)}}(\mathrm{e}^{j\omega})$, $R_{G^{(c)}}(\mathrm{e}^{j\omega})$ be the upper, lower and composite power spectra respectively. Which of the two spectra dominates when we optimize the linear predictor for this composite signal? This is easily seen. The distortion criterion is

$$\begin{aligned}
\eta &= \int_{-\pi}^{\pi} \frac{R_{G^{(c)}}(\mathrm{e}^{j\omega})}{1/|A^{(p)}(\mathrm{e}^{j\omega})|^2} d\omega \\
&\approx \frac{1}{2}\int_{-\pi}^{\pi} \frac{R_{G^{(u)}}(\mathrm{e}^{j\omega})}{1/|A^{(p)}(\mathrm{e}^{j\omega})|^2} d\omega + \frac{1}{2}\int_{-\pi}^{\pi} \frac{R_{G^{(l)}}(\mathrm{e}^{j\omega})}{1/|A^{(p)}(\mathrm{e}^{j\omega})|^2} d\omega \\
&\approx \frac{1}{2}\int_{-\pi}^{\pi} \frac{R_{G^{(u)}}(\mathrm{e}^{j\omega})}{1/|A^{(p)}(\mathrm{e}^{j\omega})|^2} d\omega.
\end{aligned}$$

Let $G_i^{(u)}$ and $G_i^{(l)}$ be independent signals with power spectrum $R_{G^{(u)}}(\mathrm{e}^{j\omega})$ and $R_{G^{(l)}}(\mathrm{e}^{j\omega})$ respectively. Furthermore, Finding the predictor is then equivalent to finding the best predictor for the signal $\frac{1}{2}G_i^{(u)} + \frac{1}{2}G_i^{(l)}$. Since $G_i^{(u)}$ has much higher variance than $G_i^{(l)}$, the linear predictor essentially ignores the signal $G_i^{(l)}$. In the spectral domain, this implies that we fit the shape of the spectral envelope, i.e. $R_{G^{(u)}}(\mathrm{e}^{j\omega})$. Note, that the linear predictor matches a signal with half the power in this case. Thus, the spectral match is about 3 dB below the envelope.

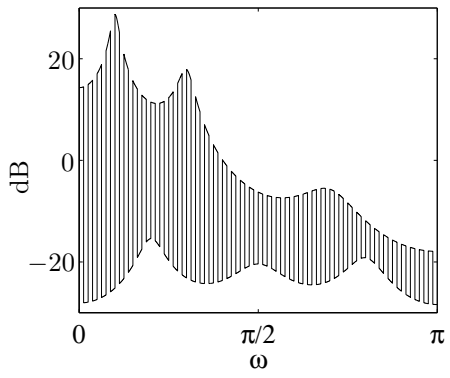

Figure 10.4: The artificial power spectrum of example 10.1.

Next, we consider the suitability of linear-prediction for the estimation of spectra with a large dynamic range. This follows immediately from the criterion equation 10.22, which is not sensitive to the magnitude of the envelope of $R_X(e^{j\omega})$. Thus, even if this envelope has a large dynamic range, it is described with essentially identical relative accuracy everywhere. This is sometimes referred to as the **global property** of linear-prediction based spectral estimation. This is a first property that is essential for matching the performance of the human auditory system for acoustic spectra that have a large dynamic range.

It is interesting to compare these results with some simple alternative criteria. A straightforward least-squares error criterion on the difference of $|1/A(e^{j\omega})|^2$ and $R_X(e^{j\omega})$ would perform significantly worse in two respects: *i*) it would not accurately represent a large dynamic range and *ii*) it would not match the spectral envelope, but be heavily influenced by the noise in local spectral valleys. A least-squares error criterion on $\log|1/A(e^{j\omega})|$ and $\log|R_X(e^{j\omega})$ would perform somewhat better in that it would fit accurately over a large dynamic range. However, it would be sensitive to noise between the local spectral peaks.

10.2.4 Optimal Linear Predictors are Minimum Phase

The filters $A^{(p)}(z)$ and $1/A^{(p)}(z)$ are commonly used in source coding algorithms, particularly in speech coders, and it is important to know if they are stable (if all bounded inputs produce a bounded output). Of course, the filter $A^{(p)}(z)$ is stable because it is an FIR filter. However, $1/A^{(p)}(z)$ is stable only if all roots of the polynomial $A^{(p)}(z)$ are within the unit circle. When both poles and zeros are within the unit circle, the filter is called minimum phase, and since we deal both with $A^{(p)}(z)$ and $1/A^{(p)}(z)$, it is convenient to adopt this terminology. In this chapter, we prove, with two different methods, that the optimal linear predictor is stable. In this section, we show that the prediction error is not minimal for nonminimum phase predictors. The second procedure is based on the reflection coefficient representation, which is obtained from the Levinson algorithm and is given in section 10.6.3.

The argument of this section consists of the following steps:

1. Consider an arbitrary filter $A^{(p)}(z)$ with one or more zeros outside the unit circle.
2. Multiply the filter by an all-pass filter that effectively moves a zero from outside to inside the unit circle.
3. Evaluate the variance of the prediction error and note that it was reduced by this operation. Thus, moving all zeros that were outside the unit circle inside the unit circle results in lower prediction error.

For simplicity, we only consider an even-order predictor polynomial $A^{(p)}(z)$ with no roots on the real axis. Since the polynomial coefficients are real, this means that complex roots come in pairs which are each others complex conjugates. If $\alpha_i^{(p)}$ are the roots in the top half plane, the polynomial can be written as follows:

$$A^{(p)}(z) = \prod_{i=1}^{\frac{p}{2}} (1 - \alpha_i^{(p)} z^{-1})(1 - \alpha_i^{(p)*} z^{-1}). \tag{10.23}$$

Let us assume that $|\alpha_m^{(p)}| > 1$. Furthermore, let us define an all-pass filter $H(z) = H(\mathrm{e}^{j\omega})$ of the following form:

$$H(z) = \frac{(1-\alpha_m^{(p)-1}z^{-1})(1-(\alpha_m^{(p)-1})^*z^{-1})}{(1-\alpha_m^{(p)}z^{-1})(1-(\alpha_m^{(p)})^*z^{-1})}, \tag{10.24}$$

where $1 \leq m \leq p/2$. On the unit circle ($|z| = 1$), we can write:

$$\begin{aligned} & |H(z)|^2 \\ = & \frac{(1-\alpha_m^{(p)-1}z^{-1})(1-(\alpha_m^{(p)-1})^*z^{-1})}{(1-\alpha_m^{(p)}z^{-1})(1-(\alpha_m^{(p)})^*z^{-1})} \frac{(1-(\alpha_m^{(p)-1})^*z)(1-(\alpha_m^{(p)-1})z)}{(1-(\alpha_m^{(p)})^*z)(1-(\alpha_m^{(p)})z)} \\ = & \frac{(1-\alpha_m^{(p)-1}z^{-1})(1-(\alpha_m^{(p)-1})^*z^{-1})}{(1-\alpha_m^{(p)}z^{-1})(1-(\alpha_m^{(p)})^*z^{-1})} \frac{((\alpha_m^{(p)})^*z^{-1}-1)((\alpha_m^{(p)})z^{-1}-1)}{((\alpha_m^{(p)-1})^*z^{-1}-1)((\alpha_m^{(p)-1})z^{-1}-1)}|\alpha_m^{(p)}|^{-4} \\ = & |\alpha_m^{(p)}|^{-4}. \end{aligned} \tag{10.25}$$

Thus, the all-pass filter has a gain of $1/|\alpha_m^{(p)}|^{-4}$. This all-pass filter can be used to move a pair of complex conjugate roots of $A^{(p)}(z)$ from outside the unit circle to inside the unit circle. If we replace $A^{(p)}(\mathrm{e}^{j\omega})$ by $H(\mathrm{e}^{j\omega})A^{(p)}(\mathrm{e}^{j\omega})$ in equation 10.22, we get:

$$\begin{aligned} & \frac{1}{2\pi}\int_{-\pi}^{\pi} R_X(\mathrm{e}^{j\omega})|H(\mathrm{e}^{j\omega})|^2|A^{(p)}(\mathrm{e}^{j\omega})|^2 d\omega \\ & \quad = |\alpha_m^{(p)}|^{-4}\frac{1}{2\pi}\int_{-\pi}^{\pi} R_X(\mathrm{e}^{j\omega})|A^{(p)}(\mathrm{e}^{j\omega})|^2 d\omega \\ & \quad = |\alpha_m^{(p)}|^{-4}\sigma_p^2. \end{aligned} \tag{10.26}$$

Since we started from the assumption that $|\alpha_m^{(p)}| > 1$, the prediction error for the minimum phase solution is smaller than σ_p^2, the error obtained for the nonminimum phase solution. The same logic can also be used for roots on the real axis. We conclude that minimization of the criterion in equation 10.22 always results in an $A^{(p)}(z)$ with its roots inside the unit circle. We summarize our results in the following theorem:

Theorem 23 *The linear predictor that minimizes the prediction error variance for given autocorrelation matrix is minimum phase.*

10.2.5 Infinite-Memory Prediction

For the finite-order linear predictor, the orthogonality relation $\mathrm{E}[E_i^{(p)}X_n] = 0$ holds for $i - p \leq n < i$. Then, for an infinite-order predictor $\mathrm{E}[E_i^{(\infty)}X_n] = 0$ holds for $n < i$. Now note that all $E_n^{(\infty)}$ with $n < i$ can be written as linear combinations of X_n with $n < i$. We have that

$$\mathrm{E}[E_n^{(\infty)}E_m^{(\infty)}] = \delta_{nm}\sigma_\infty^2, \tag{10.27}$$

where, as usual, δ_{nm} is the Kronecker delta, which is unity only if $n = m$ and zero otherwise. Thus, for a Gaussian process, infinite-memory prediction should result in a process for which an entropy-constrained scalar quantizer performs at its best.

Since the samples of the prediction error signal of an optimal infinite-order linear predictor are uncorrelated, this signal must have a constant power spectrum. In other words, the optimal infinite-order linear predictor whitens the signal power spectrum. Taking the discrete-time Fourier transform of equation 10.27 results in

$$R_{E^{(\infty)}}(\mathrm{e}^{j\omega}) = \sigma_\infty^2. \tag{10.28}$$

We conclude that the power spectrum of the infinite-order-predictor prediction error is constant, and that it equals the prediction error variance.

Equation 10.28 implies that, except for a scaling, the transfer function of the synthesis filter, $|1/A^{(\infty)}(\mathrm{e}^{j\omega})|^2$, equals the power spectrum of the original signal:

$$|1/A^{(\infty)}(\mathrm{e}^{j\omega})|^2 = \frac{1}{\sigma_\infty^2}\, R_X(\mathrm{e}^{j\omega}). \tag{10.29}$$

10.2.6 Prediction Filters and Differential Entropy

The main goal of this subsection is to show that filtering with a prediction filter does not affect the differential entropy rate of a process. To explain this, we start with a finite random vector $X^k = [X_0, \cdots, X_{k-1}]$ that we assume to be a segment of a wide-sense stationary process. As earlier, let B be a $k \times k$ lower-triangular, Toeplitz matrix, with as first column the impulse response of the filter $1/A^{(p)}(z)$. We consider the transformation

$$Y^k = BX^k, \tag{10.30}$$

that is, Y^k is the zero-state response of the filter $1/A^{(p)}(z)$ to X^k. Note that the elements of X^k do not have to be iid.

The linear relationship of equation 10.30 can be used to relate the differential entropy of X^k with the differential entropy of Y^k. First, we recall some elementary relations (e.g., [9]):

$$f_{Y^k}(y^k) = \frac{f_{X^k}(x^k)}{|\det(B)|} \tag{10.31}$$

$$dy^k = |\det(B)| dx^k \tag{10.32}$$

Thus, we can write

$$\begin{aligned} h(Y^k) &= -\int f_{Y^k}(y^k) \log(f_{Y^k}(y^k)) dy^k \\ &= -\int f_{X^k}(x^k) \log(\frac{f_{X^k}(x^k)}{|\det(B)|}) dx^k \\ &= h(X^k) + \log(|\det(B)|) \\ &= h(X^k). \end{aligned} \tag{10.33}$$

where we used that $\det(B) = 1$. This follows from the facts that B is lower-triangular and that the first sample of the impulse response is unity, implying that the diagonal of B consists of ones. In other words, we used that $A^{(p)}(z)$ is a monic polynomial. Thus, we have shown that filtering with the inverse prediction-error filter and, therefore, the forward prediction-error filter, do not affect the entropy rate of the process.

We can also obtain this result directly for infinite sequences, by using the Szegö theorem (theorem 13, equation 3.31). We first note that $\det(B)^2 = \det(B^T B)$ and that $B^T B$ falls in the class of Hermitian matrices. We start from the same point as equation 10.33, divide by k and take the limit $k \to \infty$. We can then write, under certain conditions

$$\begin{aligned} h_\infty(Y_i) &= h_\infty(X_i) + \frac{1}{2}\lim_{k\to\infty}\log(|\det(B^T B)|) \\ &= h_\infty(X_i) + \frac{1}{4\pi}\int_{-\pi}^{\pi}\log(|B(\mathrm{e}^{j\omega})|^2)d\omega, \end{aligned} \tag{10.34}$$

where $|B(\mathrm{e}^{j\omega})|^2$ is the power spectrum of a wide-sense stationary process with covariance matrix $B^T B$.

We now proceed with evaluating the second term on the right-hand side of equation 10.34. We note that the conditions we have used so-far imply that $B(z)$ is a minimum phase, monic (first coefficient is unity) polynomial, conditions that are both satisfied by the optimal prediction filter obtained from the Yule-Walker equations. We have

$$\begin{aligned} \frac{1}{2\pi}\int_{-\pi}^{\pi}\log(|B(\mathrm{e}^{j\omega})|^2)d\omega &= \oint \log(|B(z)|^2)\frac{1}{2\pi jz}dz \\ &= \oint \log(B(z))\frac{1}{2\pi jz}dz + \oint \log(B(z^{-1}))\frac{1}{2\pi jz}dz \\ &= \oint 2\mathrm{Re}(\log(B(z^{-1})))\frac{1}{2\pi jz}dz \\ &= 2\mathrm{Re}(\oint \log(B(z^{-1}))\frac{1}{2\pi jz}dz) \\ &= 2\mathrm{Re}\oint \log(\sum_{i=0}^{p} a_i^{(p)} z^i)\frac{1}{2\pi jz}dz) \\ &= 2\mathrm{Re}(\log(a_0^{(p)})) = 2\log(1) = 0. \end{aligned} \tag{10.35}$$

In equation 10.35 we used that, since $B(z)$ is minimum phase (and, therefore, has its roots inside the unit circle), $B(z^{-1})$ has no roots inside the unit circle, which in turn means that $\log(B(z^{-1}))$ is analytic inside the unit circle. Equation 10.35 shows that the mean value of the log of the power of the transfer function is zero for a minimum-phase prediction filter.

Our results lead to the following theorem:

Theorem 24 *For an operation $Y(z) = B(z)X(z)$ with a minimum-phase linear filter $B(z)$, where $B(z)$ is a real, monic polynomial in z^{-1}, the differential entropy rate of the output process, Y_i equals that of the input process X_i:*

$$h_\infty(Y_i) = h_\infty(X_i). \tag{10.36}$$

This theorem has an interesting corollary. When optimal prediction is performed, the variance of the prediction error is a nonincreasing function of the prediction order. However, the differential entropy rate is the same for the prediction error and the original process, independent of the prediction order. We know from section 3.4 that an iid Gaussian process has the maximum entropy rate for a given variance. This suggests

that the optimal prediction error process converges to an iid Gaussian process with increasing prediction order [74]. In problem 3, we show that an appropriate similarity measure to illustrate this effect is the **relative entropy rate**.

10.2.7 A Bound on Redundancy

Since a Gaussian process has the maximum differential entropy, given the variance, over all random processes (see section 3.4) the following inequality holds for redundancy

$$\begin{aligned} \rho(X_i) &\equiv h_1(X_i) - h_\infty(X_i) \\ &= h_1(X_i) - h_\infty(E_i^{(p)}) \\ &\geq h_1(X_i) - h_1(E_i^{(p)}) \\ &\geq h_1(X_i) - \frac{1}{2}\log(2\pi e\sigma_p^2), \end{aligned} \tag{10.37}$$

The bound is tightest for $p \to \infty$. The value of σ_∞^2 can be evaluated using Kolmogorov's formula, which is given in the next subsection, 10.2.8. The inequalities in equation 10.37 become equalities if the prediction residual is Gaussian. Similar inequalities for nonlinear predictors and their relation to prediction gain have been presented in [75].

10.2.8 Kolmogorov's Formula

Equation 10.35 implies that the mean of the log of the power spectrum of any signal does not change upon filtering with any minimum-phase prediction filter (also a nonoptimal filter):

$$\frac{1}{2\pi}\int_{-\pi}^{\pi}\log(R_{E^{(p)}}(\mathrm{e}^{j\omega}))d\omega = \frac{1}{2\pi}\int_{-\pi}^{\pi}\log(R_X(\mathrm{e}^{j\omega}))d\omega. \tag{10.38}$$

We now extend this result to the limit where $p \to \infty$. From section 10.2.5, we know that $R_E^{(\infty)}(e^{j\omega}) = \sigma_\infty^2$ is constant, allowing us to remove the integral. Thus, we see that we have obtained a relation between the ultimate prediction error and the power spectrum of the original signal:

$$\sigma_\infty^2 = \exp(\frac{1}{2\pi}\int_{-\pi}^{\pi}\log(R_X(\mathrm{e}^{j\omega}))d\omega). \tag{10.39}$$

This equation is known as Kolmogorov's formula.

10.2.9 The Spectral-Flatness Measure

The prediction gain is the variance of the original signal divided by the variance in the prediction error. From Parseval's theorem, the variance of the speech signal is $\sigma_0^2 = \frac{1}{2\pi}\int_{-\pi}^{\pi} R_X(\mathrm{e}^{j\omega})d\omega$. The maximum prediction gain (the gain in the limit of infinite-order prediction) is then:

$$\frac{\sigma_0^2}{\sigma_\infty^2} = \frac{\frac{1}{2\pi}\int_{-\pi}^{\pi} R_X(\mathrm{e}^{j\omega})d\omega}{\exp(\frac{1}{2\pi}\int_{-\pi}^{\pi}\log(R_X(\mathrm{e}^{j\omega}))d\omega)}. \tag{10.40}$$

For obvious reasons, the inverse of equation 10.40 is called the **spectral-flatness measure**:

$$\Xi_0 = \frac{\sigma_\infty^2}{\sigma_0^2} = \frac{\exp(\frac{1}{2\pi}\int_{-\pi}^{\pi}\log(R_X(\mathrm{e}^{j\omega}))d\omega)}{\frac{1}{2\pi}\int_{-\pi}^{\pi}R_X(\mathrm{e}^{j\omega})d\omega}. \tag{10.41}$$

Finally, we note that the spectral flatness for the prediction residual of an p'th order linear predictor is

$$\begin{aligned}\Xi_p &= \frac{\sigma_\infty^2}{\sigma_p^2} \\ &= \frac{\exp(\frac{1}{2\pi}\int_{-\pi}^{\pi}\log(R_{E^{(p)}}(\mathrm{e}^{j\omega}))d\omega)}{\frac{1}{2\pi}\int_{-\pi}^{\pi}R_{E^{(p)}}(\mathrm{e}^{j\omega})d\omega} \\ &= \frac{\exp(\frac{1}{2\pi}\int_{-\pi}^{\pi}\log(R_X(\mathrm{e}^{j\omega}))d\omega)}{\frac{1}{2\pi}\int_{-\pi}^{\pi}R_{E^{(p)}}(\mathrm{e}^{j\omega})d\omega},\end{aligned} \tag{10.42}$$

where we used equation 10.35. Since for optimal linear prediction we must have $\sigma_p^2 \leq \sigma_{p-1}^2$ (for all $p > 0$), it is seen that the spectral flatness of the prediction error cannot decrease with increasing predictor order and that minimization of the prediction error variance is equivalent to maximization of the spectral flatness measure. Furthermore, it is observed from the definition of the flatness measure (equation 10.42) that the spectral flatness measure is in the range $(0, 1]$ with $\Xi_p = 1$ corresponding to optimal prediction.

10.2.10 Linear Prediction and Gaussian Processes

As we have seen in earlier chapters, Gaussian processes are special: they are often good approximations of reality and they often make solutions analytically tractable. This is also the case for linear predictors. Generally, there is strong synergy between Gaussian densities and linear operators (in the case of Gaussian additive noise, optimal estimators are linear for example [76]), and this is no different for linear prediction and Gaussian processes.

Filtering with the optimal infinite-order predictor results in a prediction error which has a white power spectrum. Thus, if X_i is a stationary Gaussian process, the prediction error process is iid. That is, in this special case the first-order differential entropy of the prediction error equals the differential entropy rate of both the process X_i and the prediction error $E_i^{(\infty)}$:

$$\begin{aligned}h_\infty(X_i) &= h_\infty(E_i^{(\infty)}) \\ &= h_1(E_i^{(\infty)}) \\ &= \frac{1}{2}\log(2\pi\mathrm{e}\sigma_\infty^2),\end{aligned} \tag{10.43}$$

where we used equation 3.22, which specifies the differential entropy of a Gaussian variable.

The redundancy of a Gaussian process can then be related to the gain of the infinite-

order predictor

$$\begin{aligned}\rho(X_i) &\equiv h_1(X_i) - h_\infty(X_i) \\ &= \frac{1}{2}\log(2\pi e\sigma_0^2) - \frac{1}{2}\log(2\pi e\sigma_\infty^2) \\ &= \frac{1}{2}\log(\frac{\sigma_0^2}{\sigma_\infty^2}) \end{aligned} \tag{10.44}$$

i.e., for a Gaussian process, half the log prediction gain of the optimal infinite-order linear predictor equals the redundancy. Equation 10.44 specifies the redundancy relation of equation 3.33 in terms of prediction gain.

Since for optimal predictors we have that $\sigma_p^2 \leq \sigma_{p-1}^2$ for $p \in \{1, 2, \cdots\}$, the following bound holds for stationary Gaussian processes:

$$\frac{\sigma_0^2}{\sigma_p^2} \leq e^{2\rho(X_i)}, \tag{10.45}$$

where the redundancy is stated in nats.

Before we continue our discussion, a few words of caution are in order. Since the redundancy of the original Gaussian process is higher than that of the prediction error process, it may seem natural to conclude that, for the case of scalar quantization, it is advantageous simply to code the prediction error process $E_i^{(p)}$ rather than the original process X_i. However, as was seen in the introduction of this chapter, and as is discussed in more detail in section 10.4, this conclusion is not straightforward since the distortion criterion is not invariant with open-loop prediction. By using closed-loop prediction or, equivalently, noise shaping it is possible to exploit the decorrelating properties of linear predictors.

Since the infinite-order linear predictor error process has no redundancy for stationary Gaussian processes, it follows from equation 7.34 that, at a given high rate, the spectral flatness measure,

$$\Xi_0 = \frac{\sigma_\infty^2}{\sigma_0^2} = \frac{\exp(\frac{1}{2\pi}\int_{-\pi}^{\pi}\log(R_X(e^{j\omega}))d\omega)}{\frac{1}{2\pi}\int_{-\pi}^{\pi}R_X(e^{j\omega})d\omega}, \tag{10.46}$$

is the reduction in distortion obtained by applying ideal entropy-constrained scalar quantization of the infinite-memory prediction error process instead of applying such scalar quantization to the original process. Alternatively, at fixed distortion, the logarithm of the spectral-flatness measure is equal to the log rate reduction (assuming the same conditions). Naturally, this rate reduction is precisely the redundancy rate of equation 3.33 (in fact, both equation 3.33 and equation 10.44, show that $\rho(X_i) = -\frac{1}{2}\log(\Xi_0)$). For Gaussian processes and for k approaching infinity, the Karhunen-Loeve transform removes all redundancy, and so does the linear predictor.

We can combine the spectral flatness measure with the rate-distortion function for Gaussian processes at low distortion (equation 6.84), to obtain a simple expression for the distortion rate function of Gaussian processes at low distortion:

$$D_{\text{ldg}} = \sigma_0^2\,\Xi_0\,2^{-2R_{\text{ldg}}}, \tag{10.47}$$

where the subscripts "ldg" are a reminder that the equation only holds for low distortion, and Gaussian statistics; the rate is given in bits.

10.3 Linear Prediction for Given Data

So-far in this chapter, all discussion has been based on the assumption that we have knowledge of the statistics of the process. In practice, one usually receives data without prior knowledge of the statistics of the signal. The descriptions and derivations for this case mirror that of the case of given statistics provided in section 10.2.1. The main difference is that the variables are not random variables (and thus written in lowercase), and that the expectation operators are replaced by a summation. Thus, we only provide some important starting points and results. The prediction is described by

$$e_i^{(p)} = x_i - \hat{x}_i = a^{(p)} x_{i:i-p}. \tag{10.48}$$

The prediction error to be minimized is

$$\varepsilon = \sum_{i=i_1}^{i_2} e_i^2, \tag{10.49}$$

where $[n_1, n_2]$ is the summation range used.

Using the Lagrange multiplier method, we minimize ε under the constraint that $a_0^{(p)} = 1$, i.e., that $\mathbf{b}^T a^{(p)} = 1$, as in equation 10.6. The resulting **normal equations** are

$$C^{(p)} a^{(p)} = \begin{bmatrix} \varepsilon \\ 0^p \end{bmatrix}. \tag{10.50}$$

where

$$C^{(p)} = \frac{1}{i_2 - i_1 + 1} \sum_{i=i_1}^{i_2} x_{i:i-p} x_{i:i-p}^T \tag{10.51}$$

Two methods, the autocorrelation method and the covariance method, are commonly used for the determination of linear predictors for the given data case. Both of these methods are based on equation 10.50, the only difference being the summation ranges selected. This seemingly small difference results in a significant difference in the properties of the estimated predictors. In the following two subsections, we describe the two methods.

10.3.1 Autocorrelation method

In the autocorrelation method, the summation range in equation 10.49 is set to $[-\infty, \infty]$. For a practical situation, this can be done by first windowing the original signal and assuming the samples outside the window to be zero. Tapered cosine window functions (such as the Hamming and Hann window functions) generally result in better performance of the predictor than a rectangular window function, because of reduced edge effects. In speech coders, the tapered windows are often selected to be asymmetric so as to minimize coder delay [77].

A major advantage of the autocorrelation method is that the matrix $C^{(p)}$ becomes Toeplitz. For a window, w_i, with a support of length N (ranging from 1 to N), we obtain

$$\tilde{R}^{(p)} a^{(p)} = \begin{bmatrix} \varepsilon/N \\ 0^p \end{bmatrix}. \tag{10.52}$$

where

$$\tilde{R}_{i,j}^{(p)} = \frac{1}{N} \sum_{n=-\infty}^{\infty} w_{n-i} x_{n-i} w_{n-j} x_{n-j}, \qquad w_i = 0, \forall_{i \in \{\cdots -2,-1,N,N+1,\cdots\}} \tag{10.53}$$

The normal equations specifying the linear predictor with the autocorrelation method are of the same form as those obtained for the case where the statistics are known in section 10.2.1. The difference is that $R^{(p)}$ is replaced with $\tilde{R}^{(p)}$, which can be seen as an estimate of the autocorrelation matrix. Implicit is the assumption that the process is stationary and ergodic.

The autocorrelation method has two major advantages over the covariance method described below. First, the prediction filter is guaranteed to be minimum phase, since the stability properties discussed in section 10.2.4 carry over to the autocorrelation method. Second, fast computational procedures can be derived because of the Toeplitz structure of the matrix R. These procedures, which lower the complexity from $O(p^3)$ to $O(p^2)$ are the subject of section 10.6.

10.3.2 Covariance method

In the covariance method, the summation range is simply curtailed to $[p+1, N]$ and no windowing is used. The covariance method consists of equation 10.50, in combination with the coefficients

$$C_{i,j}^{(p)} = \frac{1}{N-p} \sum_{n=p+1}^{N} x_{n-i} x_{n-j}, \qquad i,j \in \{0,1,\cdots,p\}. \tag{10.54}$$

The main advantage of the covariance method over the autocorrelation method is that it does not require a windowing procedure. Thus, the prediction gain is generally somewhat higher than that obtained with the autocorrelation method.

The covariance method has some significant disadvantages. The matrix C, while symmetric, does not have a Toeplitz in structure. The symmetric structure of C can be exploited with the Choleski algorithm (e.g., [78]), but the computational effort remains $O(p^3)$. Furthermore, the linear predictor $a^{(p)}$ estimated by the covariance method is not guaranteed to be minimum phase, implying that the filter $1/A^{(p)}(z)$ is sometimes unstable. Procedures to "stabilize" the covariance method have been reported in the literature [79] but such methods generally result in a decrease of the prediction gain.

10.3.3 Robust Linear Prediction

Like in almost any application of theory, practical problems occur when linear prediction is applied to the processing of real-world signals. These problems relate to nonrobust estimation of the prediction coefficients and the mismatch between the prediction model and certain input signals. We discuss a few of these problems and their heuristic solutions in this subsection.

The Noise Correction Method

Many signals on which linear prediction is used have a large dynamic range. For example the (voiced) speech spectrum typically has a tilt of -6 dB per octave (per doubling of the frequency). This large spectral dynamic range is often further increased due to the low-pass filtering used prior to the analog-to-digital conversion process. As a result, the estimation of the linear prediction coefficients requires high computational precision to capture the description of features at the high end of the spectrum. In certain cases, the autocorrelation or covariance matrices become effectively singular given the numerical precision used. Rather than to increase the numerical precision, or find numerically more stable algorithms, it is common practice to modify the input data to prevent these numerical problems.

By adding to the original signal a low-level high-frequency noise, the dynamic range of the power spectrum is reduced [80, 79]. It is convenient to add such a contribution directly to the autocorrelation matrix for the autocorrelation method (or the corresponding covariance matrix for the covariance method) and to the signal variance. This so-called **high-frequency correction** substantially reduces numerical problems in computational devices of limited precision. The procedure is often simplified to be a **white-noise correction**. This entails simply adding a small value to the diagonal samples of the autocorrelation matrix and to the signal variance. If ϵ is the white-noise correction and $I^{(p+1)}$ is the identity matrix, then equation 10.52, modified to include the white-noise correction, becomes

$$(\tilde{R}^{(p)} + \epsilon I^{(p+1)})a^{(p)} = \begin{bmatrix} \epsilon + \varepsilon/N \\ 0^p \end{bmatrix}. \tag{10.55}$$

Lag Windows

Another method to improve numerical stability, which is somewhat less common, is the lag window method [81]. Application of the lag window minimizes the dynamic range of the spectrum. Since this is done prior to the determination of the linear prediction coefficients, it results in better numerical properties.

In the lag window procedure the autocorrelations are multiplied by a so-called *lag window*. Usually, this lag window is chosen to have a Gaussian shape. This corresponds to convolving the power spectrum with a Gaussian shape, widening the peaks of the spectrum.

In voiced speech, the spectral envelope contains fine structure generated by the vocal cords. If one is interested in the effect of the vocal tract only (as is commonly the case), a spectral smoothing prior to the computation of the predictor coefficients is reasonable and this forms a second motivation for the usage of lag windows.

Bandwidth Widening

In the application of linear prediction, it is often found that the method has the tendency to underestimate the resonance bandwidth. This effect is particularly noticeable in the estimation of the vocal-tract transfer function in speech, where it is emphasized by the

fine structure generated by the vocal cords. A commonly used remedy is bandwidth widening [82]. In this procedure, each linear prediction coefficient $a_n^{(p)}$ is multiplied by a factor γ^n (i.e., all $a_n^{(p)}$ are replaced by $\gamma^n a_n^{(p)}$). Such a multiplication moves all the poles of $H(z)$ inward by a factor γ and causes bandwidth expansion for all the poles.

The bandwidth expansion is often reported in Hz and it is thus relevant to understand the relation between pole radius and bandwidth in Hz. For simplicity, let us consider a single pole. Let a pole be represented by the factor $1 - \alpha_i z^{-1}$ in the denominator of the transfer function. Let $\alpha_i = |\alpha_i| e^{j\omega_i}$. On the unit circle, we can write

$$\begin{aligned} |1 - |\alpha_i| e^{-j(\omega-\omega_i)}| &= |1 - \mathrm{e}^{-j(\omega-\omega_i)+\log(|\alpha_i|)}| \\ &\approx |1 - (1 - j(\omega - \omega_i) + \log(|\alpha_i|))| \\ &= |j(\omega - \omega_i) - \log(|\alpha_i|))|. \end{aligned} \tag{10.56}$$

The resonance peak of the magnitude transfer function (the inverse of equation 10.56) is $\log(|\alpha|))|$ at $\omega = \omega_i$. We are down to half the resonance peak when

$$|\omega - \omega_i| = |\log(|\alpha_i|))|. \tag{10.57}$$

Defining the bandwidth as twice this width and noting that $|\alpha_i| < 1$, we obtain a bandwidth for pole i, in Hz,

$$B_i = -\frac{1}{\pi T} \log(|\alpha_i|)) \text{ Hz}, \tag{10.58}$$

where T is the sampling interval in seconds.

From equation 10.58, we see immediately that multiplication of the radius by γ expands the bandwidth to $B_i + \Delta B$, where

$$\Delta B = -\frac{1}{\pi T} \log(\gamma) \text{ Hz}. \tag{10.59}$$

Multiplication of z^{-1} in the all-pole transfer function by γ expands all poles by the same bandwidth. Bandwidth widening is commonly used in speech coders. For 8 kHz sampled speech, typical values for γ are between 0.988 and 0.996 [81], corresponding to between 10 and 30 Hz widening.

10.4 Source Coding Based on Linear Prediction

Thus-far we have discussed the basic principles of linear prediction, some of its properties, and the estimation of linear predictors from data. To allow the creation of practical coders, we must decide on the coder architecture and, for nonstationary signals, how to provide both encoder and decoder with the same predictor coefficient information.

In the first subsection of this section, we discuss in more detail how to exploit linear prediction in a source coder. We start with a discussion from the viewpoint of noise shaping and show that this leads to closed-loop prediction. We then introduce analysis-by-synthesis coding. The adaptation of the prediction filters in nonstationary signals is described in subsection 10.4.3.

10.4.1 Optimal Noise Shaping

Before embarking on a discussion of noise shaping of the prediction error process, some words on the characterization of quantization noise are in order. Rather than assuming, as we have done in the past (for example, in the introduction of this chapter), particular source signal characteristics and ideal high-rate quantization, we now make assumptions on the quantization noise. Quantization noise is often modeled as having a white power spectrum. This is guaranteed to be correct when the high-rate conditions hold and when unbounded cells can be neglected. The high-rate conditions imply that the probability density of the data is smooth compared to the quantizer cell size. In the following, we assume that quantization noise is white. We also assume that the appropriate distortion criterion is the minimum mean squared error between X_i and $\mathcal{Q}_A(X_i)$, where $\mathcal{Q}(\cdot)$ denotes the quantization operator.

The white quantization noise assumption means that for direct quantization of a process X_i we have that $\mathcal{Q}_A(X_i) - X_i$ has a white power spectrum. If we quantize the (open-loop) prediction error process, E_i, then $\mathcal{Q}(E_i) - E_i$ has a white power spectrum and upon filtering with the synthesis filter[3] $1/A(z)$, the quantization noise $(\mathcal{Q}(E_i) - E_i)/A(z) = \mathcal{Q}_A(X_i) - X_i$ has a colored spectrum with the same signal-to-noise ratio (SNR) as $\mathcal{Q}(E_i) - E_i$. Thus, we obtained no performance benefit in terms of SNR from the prediction process. However, for certain applications, the resulting coloring of the quantization noise is advantageous.

To allow us to benefit from prediction, we introduce **noise shaping**, to color the power spectrum of the quantization noise on $\mathcal{Q}(E_i) - E_i$. If we can make sure that the SNR of $\mathcal{Q}(E_i) - E_i$ is high in the spectral regions which are amplified the most by the filter $1/A(z)$ and low in the other regions, the average SNR of the signal $\mathcal{Q}_A(X_i)$ is increased by the synthesis filtering operation, the more so the sharper the resonances of the filter $1/A(z)$.

We optimize the noise shaping so as to minimize the distortion $\mathcal{Q}_A(X_i) - X_i$. However, we note that noise shaping can also be used to minimize other, possibly perceptually based, criteria.

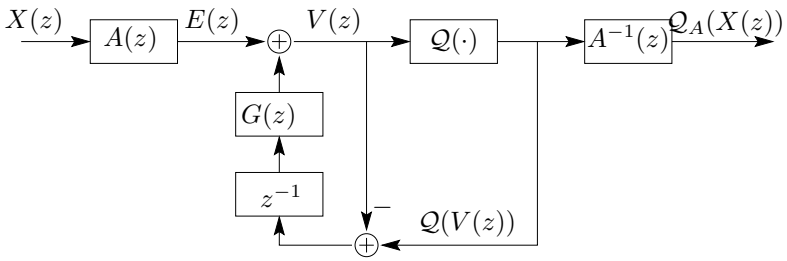

Figure 10.5: Noise shaping for the quantizer $\mathcal{Q}(\cdot)$ in a predictive coder structure. P is the predictor operator

In noise shaping, the quantization error is fed back to the input through a causal noise shaping filter, $G(z)$. To ensure proper functioning of the setup, a delay is required prior to the noise shaping filter (otherwise the quantizer would operate on its own output). Let

[3] In this analysis, we assume analysis and synthesis filters are identical, thus ignoring the results of section 6.6.2

V_i denote the signal that is quantized (it differs from E_i by the feedback), as illustrated in figure 10.5. Thus, the noise shaping system is described by

$$V(z) = E(z) + z^{-1}G(z)(\mathcal{Q}(V(z)) - V(z)), \tag{10.60}$$

The quantized output signal is now

$$\begin{aligned} \mathcal{Q}_A(X(z)) &= A^{-1}(z)\mathcal{Q}(V(z)) \\ &= A^{-1}(z)(V(z) + \mathcal{Q}(V(z)) - V(z)) \\ &= A^{-1}(z)(E(z) + (z^{-1}G(z) + 1)(\mathcal{Q}(V(z)) - V(z))) \\ &= X(z) + A^{-1}(z)(z^{-1}G(z) + 1)(\mathcal{Q}(V(z)) - V(z)). \end{aligned} \tag{10.61}$$

The quantization error is thus

$$Q(X(z)) - X(z) = A^{-1}(z)(z^{-1}G(z) + 1)(\mathcal{Q}(V(z)) - V(z))). \tag{10.62}$$

Using the white quantization noise assumption, the signal distortion can be written as

$$\mathrm{E}[|Q(X_i) - X_i|^2] = \frac{1}{2\pi}\int_{-\pi}^{\pi} \frac{|\mathrm{e}^{-j\omega}G(\mathrm{e}^{j\omega}) + 1|^2}{|A(\mathrm{e}^{j\omega})|^2}\sigma_{\mathcal{Q}}^2\, d\omega, \tag{10.63}$$

where $\sigma_{\mathcal{Q}}^2 = |\mathcal{Q}(V(\mathrm{e}^{j\omega})) - V(\mathrm{e}^{j\omega})|^2$ is the variance of the white quantization noise of the quantizer.

We note from the structure of the noise shaping that $G(z)$ must be a causal filter. Then, it is convenient to define $G'(z) \equiv z^{-1}G(z) + 1$, and the distortion can be written as

$$\mathrm{E}[|Q(X_i) - X_i|^2] = \sigma_{\mathcal{Q}}^2\frac{1}{2\pi}\int_{-\pi}^{\pi} \frac{|G'(\mathrm{e}^{j\omega})|^2}{|A(\mathrm{e}^{j\omega})|^2}\, d\omega. \tag{10.64}$$

We now want to find the monic polynomial $G'(z)$ that minimizes the expression in equation 10.64. We see that the optimal noise shaping problem is equivalent to finding the optimal predictor for a signal with power spectrum $1/|A(\mathrm{e}^{j\omega})|^2$ (cf. section 10.2.3). We know that the optimal infinite-order linear predictor has a spectral flatness measure of unity, and that spectral flatness cannot decrease with increasing predictor order. We then conclude that

$$G'(\mathrm{e}^{j\omega}) = A(\mathrm{e}^{j\omega}) \tag{10.65}$$

gives optimal noise shaping, since it ensures a spectral flatness measure of unity.

Our optimal noise-shaping filter result is equivalent to closed-loop prediction. This can be shown by a simple rearrangement. For $z^{-1}G(z) = A(z) - 1$ we have

$$V(z) = E(z) - (A(z) - 1)V(z) + (A(z) - 1)A(z)Q(X(z)) \tag{10.66}$$

or

$$A(z)V(z) = A(z)X(z) + (A(z) - 1)A(z)Q(X(z)) \tag{10.67}$$

and thus

$$V(z) = X(z) + (A(z) - 1)Q(X(z)), \tag{10.68}$$

which describes a closed-loop predictor.

10.4.2 Analysis-by-Synthesis

In the previous subsection, we found that, under the assumption of white quantization noise, closed-loop prediction allows us to exploit the decorrelating properties of prediction. It is natural to combine the closed-loop predictor with a scalar quantizer, resulting in the so-called DPCM (differential pulse-code modulation) coder. If the predictor and/or the quantizer is adaptive, then the coder is known as ADPCM (adaptive DPCM). The DPCM principle was earlier illustrated in figure 10.2. When we make the quantization explicit (as a codebook search), we obtain figure 10.6. We note that the feedback within the quantizer structure ("encode" within the figure) is performed within each time step of the remainder of the figure.

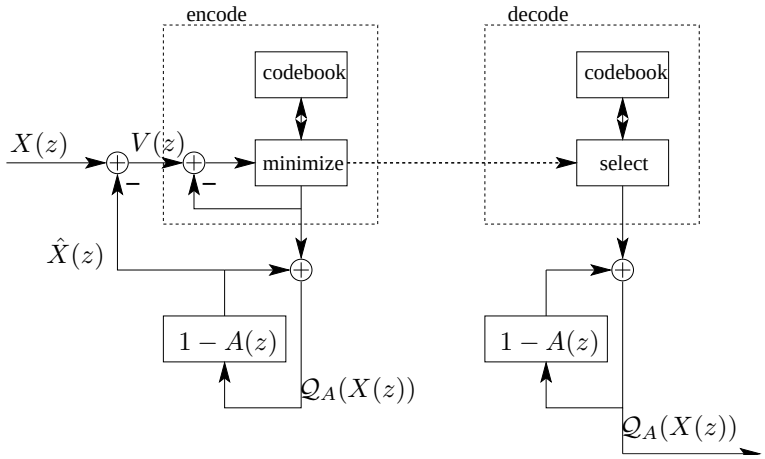

Figure 10.6: The principle of closed-loop prediction. The loop within the encode box indicates the quantizer selection process for each sample.

Since the DPCM structure removes correlations from the structure prior to quantization, there would, at first sight, seem to be relatively little motivation to use vector rather than scalar quantization. Some advantage is to be expected from the space filling and shape advantages discussed in section 7.5, but this effect may be affected by any filtering performed on codebook vectors. A more important motivation for the the vector quantizer is that it can account for dependencies that cannot be removed by the prediction-error filter (and that are, thus, not modeled by the synthesis filter). In other words, the vector quantizer tries to account for the nonGaussian nature of the signal. An important class of such dependencies results in a time-varying envelope of the signal variance. In the case of speech, the linear-prediction residual signal generally contains so-called pitch pulses, where signal energy is localized.

Thus, there is significant motivation for the combination of vector quantization and prediction. However, the DPCM structure of figure figure 10.6 does not lead to an effective structure exploiting vector quantization. The most straightforward generalization of the scalar quantizer in figure 10.6 is to look at the signal as a sequence of vectors and predict vectors. This procedure is ineffective, since the prediction now has to predict over longer time intervals.

A more effective incorporation of vector quantization in the DPCM structure is obtained

if we modify the coding structure prior to introducing vector quantization. Let ϵ_i be the quantizer error. Then it follows from figure 10.6 that

$$\begin{aligned} \epsilon_i &= V_i - \mathcal{Q}(V_i) = X_i - \hat{X}_i - \mathcal{Q}(V_i) && (10.69) \\ \mathcal{Q}_A(X_i) &= \hat{X}_i + \mathcal{Q}(V_i) && (10.70) \end{aligned}$$

and that, thus

$$\epsilon_i = V_i - \mathcal{Q}(V_i) = X_i - \mathcal{Q}_A(X_i). \quad (10.71)$$

Equation 10.71 is important for several reasons:

1. It shows explicitly that, for single-letter distortion criteria, the distortion is invariant with the closed-loop prediction operation.
2. It explains why the predictor is advantageous for scalar quantization. Consider a quantizer that can obtain a certain signal-to-noise ratio (SNR) for a given rate (e.g., an ideal entropy-constrained high-rate scalar quantizer operating on a Gaussian signal). At a given rate, the quantizer operates at the same SNR for residual and original signal. Equation 10.71 says that the quantization error does not change with filtering by the synthesis filter, and this means that the SNR for the reconstructed signal increased by the prediction gain of the filter. For speech this is typically 10 dB or so.
3. We can modify the structure of the DPCM coder to that shown in figure 10.7. This structure facilitates vector quantization. The coding structure of figure 10.7 is called **analysis-by-synthesis coding**.

The analysis-by-synthesis principle is not unique to linear prediction and goes back at least to 1961 [83]. Linear prediction based analysis-by-synthesis coding forms the basis for almost all speech coding standards developed in the last two decades of the 20'th century. The well-known CELP coder [84, 85] falls into this class of coders. Some pioneering works are [80, 86, 87].

10.4.3 Forward and Backward Adaptation of the Predictor

For proper operation of a linear-prediction based coder, the receiver must have access to the same prediction coefficients as the transmitter. This can be done in three ways: *i*) by keeping the predictor fixed, *ii*) by estimating the predictor coefficients from earlier reconstructed speech, or *iii*) by transmitting the predictor coefficients as side-information.

The case where the predictor is estimated from the original signal and transmitted as side information is known as **forward adaptation** of the predictor whereas the case where the predictor is estimated from the reconstructed speech is known as **backward adaptation** of the predictor. In speech coding, backward adaptation is mainly used for higher bit rates (i.e., rates over 10 kb/s, such as the 32 kb/s ITU G.726 ADPCM coder and the 16 kb/s ITU G.728 coder). Because backward adaptation of the predictor is based on the earlier reconstructed speech signal, it tends to become less effective at low bit rates. As a result, forward adaptation of the predictor, which requires quantization of the prediction coefficients, is generally used for coders with lower bit rates (e.g., the 8 kb/s ITU G.729 coder, and the 12 kb/s GSM EFR coder). How to quantize the prediction parameters is discussed in section 10.5.

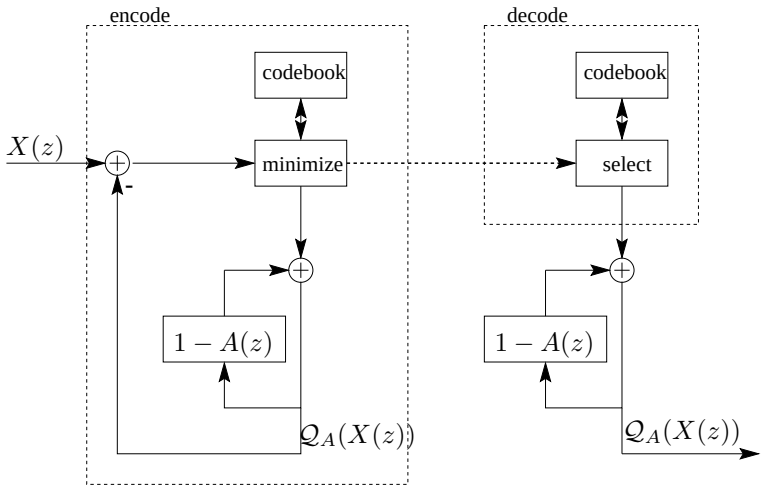

Figure 10.7: The principle of analysis-by-synthesis. The loop within the encode box indicates the quantizer selection process for each sample or vector.

10.5 Quantizing Linear Prediction Coefficients

In source coders using forward adaptation, the predictor coefficients must be quantized to allow transmission as side information. Furthermore, it has empirically been found to be advantageous to transmit the predictor coefficients at regular time intervals and interpolate the coefficients between the updates. Both tasks can be performed more efficiently if the predictor coefficients are subjected to a nonlinear transform prior to quantization and interpolation. The inverse transform is applied prior to usage of the coefficients.

The desirability of a particular representation depends on several factors. It is desirable that the coefficients of the representation are independent, so that scalar quantization is efficient. However, in practice, vector quantization has been found to be a viable option for predictor coefficient quantization. More importantly, it is desirable that a weighted squared-error is a proper criterion, so that computationally efficient quantization can be performed. Finally, it is desirable that quantization and interpolation cannot lead to unstable filter representations. In this section, we first discuss how to evaluate performance of quantizers and then discuss several representations and their quantization and interpolation performance.

10.5.1 Evaluating Quantizer Performance

Since the estimation of the predictor coefficients (the identification of the AR model) uses a particular criterion, perhaps best understood in the form of the Itakura-Saito criterion (equations 10.18 and 10.19), it is surprising to note that that current practical quantizer-evaluation methods in prediction generally employ criteria different from that used for the estimation of the predictor coefficients. This situation can be motivated from the fact that, in any case, it is natural to use perceptually based criteria to evaluate performance. For the estimation of the predictor coefficients, the criterion also must

lead to practical computations. For the evaluation of the quantizer performance, this is not the case, and this has led to a discrepancy between the criteria used for estimation of the predictor coefficients and the criteria used for evaluating quantizer performance.

The best criterion for evaluating quantizer performance of speech coders is a subjective listening test where people evaluate the quality of the resulting coded speech. However, such **subjective tests** are time-consuming and expensive. Therefore, the evaluation of an predictor-coefficient quantization procedure is usually performed using an objective measure (a measure that can be evaluated by means of a computer) operating directly on the predictor coefficients, and not involving the entire coding structure. Usually, the criteria are rather simple. More sophisticated objective measures for spectral fidelity based on understanding of the human auditory systems [88, 89, 90, 91] are not used in the evaluation of the precision of the predictor coefficient quantizers. This can be motivated by the complicated interplay between quantizer performance and the performance of the entire coder, particularly since the quantization of the excitation signal is generally based on a weighted squared-error criterion.

A simple spectral distortion measure that is most commonly used in the literature to evaluate the goodness of a quantizer (e.g., [92, 93, 94, 95, 96, 97, 98, 99]) is defined as

$$D = \sqrt{\frac{1}{\pi}\int_0^{\pi} [10\log_{10}(|A(\mathrm{e}^{j\omega})|^2) - 10\log_{10}(|\mathcal{Q}(A)(\mathrm{e}^{j\omega})|^2)]^2 d\omega}, \tag{10.72}$$

where D is the distortion in dB and $\mathcal{Q}(A)(\mathrm{e}^{j\omega})$ is the power spectrum associated with the filter transfer function corresponding to the quantized predictor coefficients. The criterion accounts for the large dynamic range of hearing by means of the logarithm.

To evaluate the performance of a quantizer, the spectral distortion of equation 10.72 is averaged over all blocks (typically 100.000 to 1.000.000) in a set of test data. The resulting average value represents the distortion associated with a particular quantizer. Most studies [100, 93, 101, 102] have used an average spectral distortion of 1 dB as difference limen for spectral transparency. However, it has been observed [94] that too many outlier frames in the speech utterance having large spectral distortion can cause audible distortion, even when the average spectral distortion is 1 dB. Therefore, more recent studies [94, 103, 95] have tried to reduce the number of outlier frames, in addition to the average spectral distortion.

10.5.2 Alternative Prediction Coefficient Representations

In source coders with forward adapting predictors, the information contained in the predictor coefficients is usually quantized at regularly spaced time intervals and then interpolated between the updates. To obtain the best performance, it is important to perform this quantization and interpolation using an appropriate representation.

The predictor parameters are commonly interpolated over intervals of 20 to 30 ms. For good interpolation performance it is essential that *i)* the transfer function spectrum evolves smoothly when the parameters of the particular representation are interpolated linearly and *ii)* the filters associated with interpolated coefficients remain minimum phase.

To facilitate quantization, the representation should at least satisfy the following crite-

ria: *i)* a computationally simple error criterion can be employed and *ii)* the filter remains minimum phase after the quantization operation. The first condition results from implementation constraints. It is estimated (e.g., [104]) that about 20 bits are required for quantization of the predictor coefficients in speech. Such large codebooks lead both to the usage of structured codebooks and to the usage of the weighted squared-error criterion.

Since we generally intend to keep quantization errors small, expansions can be used to relate more sophisticated criteria, such as the (admittedly flawed) criterion of equation 10.72 to the squared error criterion for a particular representation. Let s^k denote a k-dimensional vector representing the log power spectrum of the signal in a practical implementation. Quantization introduces an error Δs. Assuming certain conditions on the smoothness of s^k, we have that, for a sufficiently small change in a parameter representation b^m, say Δb^m,

$$\Delta s^k = J \Delta b^m, \tag{10.73}$$

where J is a $k \times m$ Jacobian matrix. This then implies that the spectral distortion is

$$\Delta s^{kT} \Delta s^k = \Delta b^{mT} J^T J \Delta b^m. \tag{10.74}$$

We call the $m \times m$ matrix $J^T J$ the spectral sensitivity matrix.

If $J^T J$ is the identity matrix (possibly multiplied by a scalar), then a simple minimum squared-error distortion criterion for $\vec{b}$ would be accurate for small b^m. For predictor coefficient representations, this situation does not occur. If $J^T J$ is diagonal, then a simple weighting would suffice. As we see, this occurs for the line-spectral frequency representation (the weighting varies with the input). If the sensitivity matrix is not diagonal, then independent quantization (based on the squared-error criterion) of the components of b^m leads to degraded performance compared to joint optimization even when ignoring the space-filling and shape advantages discussed in section 7.5.

The Predictor-Coefficient Representation

The predictor coefficients themselves, $a_i^{(p)}$, are not well behaved with respect to either quantization or interpolation. In both cases, stability of the synthesis filter is not guaranteed. Moreover, the sensitivity matrix is nondiagonal and changes rapidly as a function of the coefficients. In general, small changes in any one predictor coefficient can lead to large changes in the transfer function spectrum and the spectrum. It is for these reasons that we discuss alternative representations for the predictor coefficients. We note that, for speech, the predictor coefficients are statistically highly dependent.

The Reflection-Coefficient Representations

As is shown in section 10.6, the reflection coefficients $\{k_i\}$ form a representation of the linear predictor associated with a lattice filter realization of the predictor filter. Since this representation is implicit in common fast computational procedures (see section 10.6)), the reflection coefficients are often available without additional computation.

A major advantage of the reflection coefficients over the predictor coefficients is that the minimum-phase property is easily conserved by keeping the the magnitude of the

reflection coefficients at less than unity. This is proven in section 10.6.3. Furthermore, generally no additional computational effort is required to obtain these coefficients.

The main disadvantage of the reflection-coefficient representation is that the spectral sensitivity matrix $J^T J$ is not diagonal. Moreover, the matrix changes significantly over the parameter range. The diagonals of the sensitivity matrix are U-shaped as a function of value of the reflection coefficients. This drawback is important only at lower rates, and can easily be overcome by the use of an appropriate nonlinear transformation which expands the region near $|k_i| = 1$. Two such transformations are the log-area ratio transformation [82] and the inverse-sine transformation [105]. The arcsine reflection coefficients are defined as

$$ASRC_i = \sin^{-1}(k_i). \tag{10.75}$$

The log-area ratios are defined as

$$LAR_i = \log(\frac{1+k_i}{1-k_i}), \tag{10.76}$$

The log-area ratios obtained their name because the reflection coefficients are associated with computing the transfer function obtained for pressure waves moving through a concatenation of tube sections of various width (a commonly used model of the human vocal tract). The log-area ratios are the logarithm of the ratio of the areas of two successive tube sections.

The reflection coefficient are generally highly dependent for speech signals.

The Line Spectral Frequency Representation

Experiments have shown that the line-spectral frequency (LSF) representation (introduced by Itakura [106]) provide good performance with respect to both quantization and interpolation (e.g, [107, 108, 104]). The spectral sensitivity matrix $J^T J$ of the LSF is diagonal with respect to the criterion of equation 10.72 [109, 110]. As a result, the LSF representation has become the most common representation for quantization and interpolation of the predictor coefficients. The main disadvantage of the LSF representation is the complexity required for its computation.

To define the LSFs, the filter polynomial is used to construct two polynomials,

$$P_p(z) = A^{(p)}(z) + z^{-1}A^{(p)\#}(z), \tag{10.77}$$

and

$$Q_p(z) = A^{(p)}(z) - z^{-1}A^{(p)\#}(z), \tag{10.78}$$

where $A^{(p)\#}(z) = z^{-p}A^{(p)}(z^{-1})$. The polynomials $P_p(z)$ and $Q_p(z)$ have the following two properties: *i)* all roots of $P_p(z)$ and $Q_p(z)$ lie on the unit circle, *ii)* the roots of $P_p(z)$ and $Q_p(z)$ are interlaced with each other; i.e., the LSFs are ordered. The former property allows the roots to be specified as frequencies and the latter property facilitates interpolation. The frequencies (angles) corresponding to the roots are the LSFs. An example of the LSF representation is shown in figure 10.8.

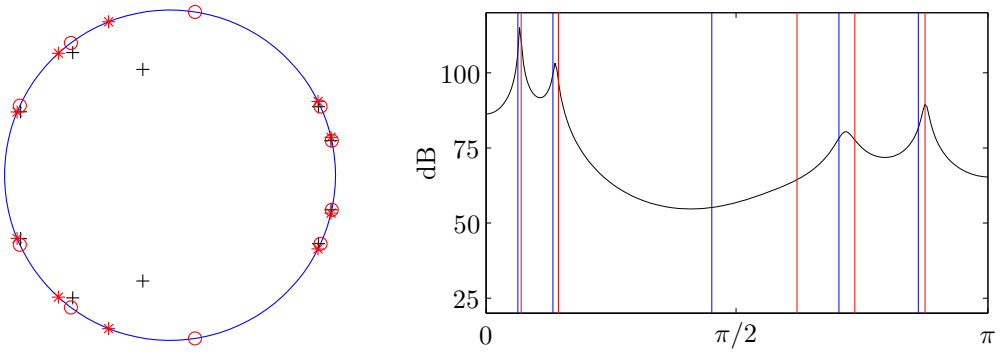

Figure 10.8: The line-spectral frequencies for a typical speech power spectrum and a typical model order used in speech coding. The left figure shows the ten poles (+) and the odd (o) and even (*) LSF in the z-plane; the right figure shows the corresponding power spectrum and the line-spectral frequencies. (The low-frequency poles and LSF are too close to resolve in the left figure.)

We now prove that the roots of $P_p(z)$ are located on the unit circle; the same procedure is valid for $Q(z)$. For this purpose, we use the properties of all-pass functions given in Appendix E to show that the poles of $P(z)$ and $Q(z)$ are on the unit circle. We write

$$P(z) = A^{(p)}(z)(1 + z^{-1}\frac{A^{(p)\#}(z)}{A^{(p)}(z)}). \tag{10.79}$$

The term $\frac{A^{(p)\#}(z)}{A^{(p)}(z)}$ is an all-pass filter with unity gain on the unit circle. Thus, it is seen that

$$|z^{-1}\frac{A^{(p)\#}(z)}{A^{(p)}(z)}| \begin{array}{rll} > & 1, & |z| < 1 \\ = & 1, & |z| = 1 \\ < & 1, & |z| > 1 \end{array} \tag{10.80}$$

For the roots of $P(z)$ we must have that $z^{-1}\frac{A^{(p)\#}(z)}{A^{(p)}(z)} = -1$. From equation 10.80 it is seen that this is possible only if $|z| = 1$. Thus, all roots of $P(z)$ (and $Q(z)$) must fall on the unit circle.

It can also be shown [100] that $A(z)$ is minimum-phase only if its LSFs satisfy the roots-on-the-unit-circle and interlacing properties. Thus, the stability of the synthesis filter (which is an important pre-requirement for speech coding applications) can be ensured by quantizing the predictor-coefficient parameters in LSF domain.

Because of the ordering property, all LSF are easily displayed as a function of time in a single graph. Typically, major spectral peaks are surrounded by an LSF on each side, as is shown in figure 10.8. In general, each LSF determines most strongly the power spectrum near to its frequency.

LSF with an index differing by more than two have a relatively independent effect on the power spectrum even for large-scale changes in their values. This leads to an approximate statistical independence and suggests splitting the LSF in groups prior to quantization. Such systems have indeed been shown to perform well for speech coding [52]. Although the line spectral frequencies are not statistically independent,

it has been shown that, for a given AR model, the covariance matrix for the line-spectral frequencies computed from the empirical predictor coefficients is diagonal [109, 110]. This is associated with class-conditional independence, which is useful for speech recognition[4].

10.6 Lattice Filters and Fast Predictor Computations

The cost of source coders is often related to the computational effort they require. In this section, we describe procedures to lower the computational effort of the estimation of the linear-prediction coefficients. However, the described methods are also useful since they an alternative representation of the prediction coefficients and provide insight into filter stability.

In practice, the linear predictor coefficients are often computed from equations 10.11 and 10.53. Here, we write these equations as

$$R^{(p)}a^{(p)} = \begin{bmatrix} \sigma_p^2 \\ 0^p \end{bmatrix}, \tag{10.81}$$

where $R^{(p)}$ is Toeplitz and symmetric. Because of the structure of $R^{(p)}$, the equations can be solved with a computational complexity of $O(p^2)$. Two procedures are used for this purpose: the Levinson algorithm (the particular flavor of this algorithm used here is sometimes called the Levinson-Durbin algorithm) and the Schur algorithm. Both these procedures lead to an alternative representation of the predictor coefficients, the so-called reflection coefficients, and an alternative filter structure: the lattice filter.

This section is organized as follows: we start our discussion with the Levinson algorithm, then discuss some properties of the reflection coefficients, introduce the lattice filter, and conclude with the Schur algorithm. Appendices provide matlab code for the various algorithms.

10.6.1 The Levinson Algorithm

The problem of solving $R^{(p)}a^{(p)} = [\sigma_p^2 0^{pT}]^T$ for the predictor $a^{(p)}$ appears to be very similar to solving $R^{(p_1)}a^{(p-1)} = [\sigma_{p-1}^2 0^{p-1,T}]^T$ for the order $p-1$. The matrix $R^{(p)}$ is almost identical to the matrix $R^{(p-1)}$ except for an additional row and column. Indeed, if we substitute $a^{(p-1)}$, augmented with a 0, for $a^{(p)}$ in the left-hand side of equation 10.81, we have only one undesired, scalar entry (denoted as δ_{p-1}) on the right-hand-side:

$$R^{(p)} \begin{bmatrix} a^{(p-1)} \\ 0 \end{bmatrix} = \begin{bmatrix} \sigma_{p-1}^2 \\ 0^{p-1} \\ \delta_{p-1} \end{bmatrix}. \tag{10.82}$$

The Levinson algorithm modifies the trial vector $[a^{(p-1)T}0]^T$ so as to force the scalar δ_{p-1} to zero. Let the superscript $^{\#}$ denote reversal of the vector indices (i.e., $a^{(p)\#} = [a_p^{(p)}, \cdots, a_1^{(p)}, 1]^T$). Then it is easily seen that, because of the symmetric Toeplitz nature

[4]This leads one to consider the notion of classification followed by scalar quantization for LSF.

of $R^{(p)}$, we have

$$R^{(p)} \begin{bmatrix} 0 \\ a^{(p-1)\#} \end{bmatrix} = \begin{bmatrix} \delta_{p-1} \\ 0^{p-1} \\ \sigma^2_{p-1} \end{bmatrix}. \tag{10.83}$$

To simplify our solution we combine equations 10.82 and 10.83 as follows:

$$R^{(p)} \begin{bmatrix} a^{(p-1)} & 0 \\ 0 & a^{(p-1)\#} \end{bmatrix} = \begin{bmatrix} \sigma^2_{p-1} & \delta_{p-1} \\ 0^{p-1} & 0^{p-1} \\ \delta_{p-1} & \sigma^2_{p-1} \end{bmatrix}. \tag{10.84}$$

To remove the δ_{p-1} we multiply on the right by a matrix of the form $\begin{bmatrix} 1 & k_p \\ k_p & 1 \end{bmatrix}$:

$$R^{(p)} \begin{bmatrix} a^{(p-1)} & 0 \\ 0 & a^{(p-1)\#} \end{bmatrix} \begin{bmatrix} 1 & k_p \\ k_p & 1 \end{bmatrix} = \begin{bmatrix} \sigma^2_{p-1} & \delta_{p-1} \\ 0^{p-1} & 0^{p-1} \\ \delta_{p-1} & \sigma^2_{p-1} \end{bmatrix} \begin{bmatrix} 1 & k_p \\ k_p & 1 \end{bmatrix}. \tag{10.85}$$

The sign chosen for the **reflection coefficients**, k_p, is obviously arbitrary and it is not consistent in the literature. With the choice $k_p = -\delta_{p-1}/\sigma^2_{p-1}$ we can write

$$R^{(p)} \begin{bmatrix} a^{(p)} & a^{(p)\#} \end{bmatrix} = \begin{bmatrix} \sigma^2_p & 0 \\ 0^{p-1} & 0^{p-1} \\ 0 & \sigma^2_p \end{bmatrix}, \tag{10.86}$$

where we used

$$\begin{bmatrix} a^{(p)} & a^{(p)\#} \end{bmatrix} = \begin{bmatrix} a^{(p-1)} & 0 \\ 0 & a^{(p-1)\#} \end{bmatrix} \begin{bmatrix} 1 & k_p \\ k_p & 1 \end{bmatrix} \tag{10.87}$$

and where $\sigma^2_p = \sigma^2_{p-1} + \delta_{p-1} k_p$. Note that equation 10.86 is what we originally set out to achieve. The manipulations to get there form the Levinson algorithm.

Let us denote the first row (and column) of the Toeplitz matrix $R^{(p)}$ as $[R_0, R_1, \cdots]$. Then, to obtain a q'th order predictor, the Levinson algorithm requires the following operations:

1. Set p=1. Set $a^{(0)} = 1$ and $\sigma^2_0 = R_0$.
2. Compute $\delta_{p-1} = \sum_{i=0}^{i=p-1} a_{p-1,i}\ R_{p-i}$.
3. Compute $k_p = -\frac{\delta_{p-1}}{\sigma^2_{p-1}}$.
4. Compute $a^{(p)}$ with equation 10.87.
5. Compute $\sigma^2_p = \sigma^2_{p-1} + \delta_{p-1} k_p$.
6. If $p < q$, set $p \to p + 1$ and go back to 2.

Appendix F provides the algorithm in Matlab.

From equation 10.87 we can also see that it is easy to compute the predictor coefficients $a^{(n)}_i$, $1 \le i \le n$, given the reflection coefficients $k_1, ..., k_p$, with $1 \le n \le p$. This procedure is called the **step-up** procedure. A Matlab step-up routine is provided in Appendix F.

10.6.2 Basic Reflection Coefficient Properties

Step 5 of the Levinson algorithm can also be written as $\sigma_p^2 = (1 - k_p^2)\sigma_{p-1}^2$. Thus, since $\sigma_p^2 \geq 0$ we have that

$$k_p^2 \leq 1. \tag{10.88}$$

In other words, for a stationary process the magnitude of the reflection coefficients corresponding to the prediction filter must be less than or equal to unity.

From $\sigma_p^2 = (1 - k_p^2)\sigma_{p-1}^2$ it follows that the prediction gain of the p'th order predictor is related to the predictor coefficients by

$$\frac{\sigma_0^2}{\sigma_p^2} = \prod_{i=1}^{i=p}(1 - k_i^2). \tag{10.89}$$

Note that this prediction gain is defined as the ratio of the variance of the original process and the prediction-error process, for the optimal p'th order predictor operating on a stationary signal. In practical situations, where the predictor is estimated from a finite data sequence rather than from accurate signal statistics, the prediction gain as estimated from the average signal power over the data sequence can be quite different. In subsection 10.6.3, we show that the filter $A^{(p)}(z) = a_0^{(p)} + a_1^{(p)}z^{-1} + \cdots a_p^{(p)}z^{-p}$ is stable if all reflection coefficients are less than unity in magnitude.

From equation 10.87 it can also be seen that $k_p = a_p^{(p)}$. Using this property, one readily devise the so-called **step-down** algorithm, which provides the reflection coefficients given the vector $a^{(p)}$.

10.6.3 The Reflection Coefficients and Stability

The minimum-phase property of the optimal linear prediction obtained from the Yule-Walker equations was discussed earlier in section 10.2.4. In this subsection, we show another proof, based on the properties of the reflection coefficients. In section 10.6.2, we showed that the reflection coefficients are of magnitude less than unity for a stationary signal, i.e., if $A^{(p)}(z)$ is stable. In this subsection, we show that, conversely, $A^{(p)}(z)$ is minimum phase if $|k_i| < 1, i = 1, \cdots, p$. We know from equation 10.87 that $A^{(p)}(z) = A^{(p-1)}(z) + z^{-1}k_pA^{(p-1)\#}(z)$. Now let α be a root of $A^{(p)}(z)$. Then

$$k_p = -\alpha\frac{A^{(p-1)}(\alpha)}{A^{(p-1)\#}(\alpha)}. \tag{10.90}$$

It is noted that the right-hand side of 10.90 corresponds to the description of an all-pass filter. From the properties of the all-pass filter shown in Appendix E, it is seen that

$$\begin{array}{rcl} > & 1 \Leftrightarrow & |\alpha| > 1, \\ |k_p| = & 1 \Leftrightarrow & |\alpha| = 1, \\ < & 1 \Leftrightarrow & |\alpha| < 1. \end{array} \tag{10.91}$$

Applying this result recursively to all optimal prediction filters of order less than p, we see that $A^{(p)}(z)$ is minimum phase if all reflection coefficients $k_1, \cdots, k_p$ have magnitude less than unity.

Note also that if $k_1, \cdots, k_{p-1}$ have magnitude less than unity, and $|k_p| = 1$, then all roots of $A^{(p)}(z)$ fall on the unit circle. As was shown in subsection 10.5.2, this is exploited in the definition of the LSF representation. Furthermore, if $|k_p| > 1$, then all roots of $A^{(p)}(z)$ are outside the unit circle.

10.6.4 The Lattice Filter

The Levinson algorithm provides us with both the reflection coefficients, k_p, and the predictor coefficients $a_k^{(p)}$. It is useful to write the recursion of the predictors in polynomial form. Each row of $a^{(p)}$ corresponds to the coefficient of a power of z^{-1}. Equation 10.87 can then be written as

$$\left[\begin{array}{cc} A^{(p)}(z) & A^{(p)\#}(z) \end{array}\right] = \left[\begin{array}{cc} A^{(p-1)}(z) & z^{-1}A^{(p-1)\#}(z) \end{array}\right] \left[\begin{array}{cc} 1 & k_p \\ k_p & 1 \end{array}\right]. \tag{10.92}$$

We can make this a little more elegant:

$$\left[\begin{array}{cc} A^{(p)}(z) & A^{(p)\#}(z) \end{array}\right] = \left[\begin{array}{cc} A^{(p-1)}(z) & A^{(p-1)\#}(z) \end{array}\right] \left[\begin{array}{cc} 1 & 0 \\ 0 & z^{-1} \end{array}\right] \left[\begin{array}{cc} 1 & k_p \\ k_p & 1 \end{array}\right]. \tag{10.93}$$

From this it is seen that

$$\left[\begin{array}{cc} A^{(p)}(z) & A^{(p)\#}(z) \end{array}\right] = \left[\begin{array}{cc} 1 & 1 \end{array}\right] \prod_{i=1}^{i=p} \left[\begin{array}{cc} 1 & 0 \\ 0 & z^{-1} \end{array}\right] \left[\begin{array}{cc} 1 & k_i \\ k_i & 1 \end{array}\right], \tag{10.94}$$

which corresponds to a lattice filter description. Let $s(z)$ be an input signal. Then multiplying by $s(z)$ gives as output the desired $A^{(p)}(z)s(z)$ in the left column on the left-hand side. This can be implemented using the right-hand side. The input signal is first split into two channels and these form the input to p sections which perform the operations

$$\left[\begin{array}{cc} 1 & k_i \\ z^{-1}k_i & z^{-1} \end{array}\right]. \tag{10.95}$$

As an aside, we note that the lattice filter inherent in the Levinson (and Schur) algorithm(s) is similar in form to the transfer function obtained for pressure waves moving through a concatenation of tube sections of various width. The latter representation is a common model for the human vocal tract [111]. However, only for specific terminations of the tube sections is the tube model equivalent to the lattice filter obtained from the Levinson algorithm. It is not possible to obtain a tube model corresponding to the physical vocal tract by performing an analysis of the signal based on the Levinson (or Schur) algorithm(s).

10.6.5 The Schur Algorithm

We introduce the Schur algorithm as an alternative method to compute the reflection coefficients k_p. For consistency, we provide a derivation which is along the same lines as our derivation of the Levinson algorithm. Describing the Schur algorithm after the

Levinson algorithm may not do proper justice to the significance of the Schur procedure, whose historical roots [112, 113] go back further than the Levinson algorithm. However, it is consistent with the later adoption of the Schur algorithm in many applications, where it is also sometimes referred to as the LeRoux or LeRoux-Gueguen algorithm, after researchers who reinvented it [114].

Let our ultimate aim be to solve for a q'th order predictor and let us say that we are at the p'th step of the Schur recursion. We write

$$R^{(q)} \begin{bmatrix} a^{(p)} & a^{(p)\#} \\ 0^{q-p} & 0^{q-p} \end{bmatrix} = \begin{bmatrix} g_{0,0}^{(p)} & g_{0,1}^{(p)} \\ g_{1,0}^{(p)} & g_{1,1}^{(p)} \\ \cdot & \cdot \\ g_{q,0}^{(p)} & g_{q,1}^{(p)} \end{bmatrix}. \tag{10.96}$$

We call the matrix on the right-hand side the **generator matrix** $G^{(p)}$:

$$G^{(p)} \equiv \begin{bmatrix} g_{0,0}^{(p)} & g_{0,1}^{(p)} \\ g_{1,0}^{(p)} & g_{1,1}^{(p)} \\ \cdot & \cdot \\ g_{q,0}^{(p)} & g_{q,1}^{(p)} \end{bmatrix}. \tag{10.97}$$

We already know that $R^{(p)}[a^{(p)}]^T = [\sigma_p 0^{pT}]^T$, and we therefore also know that there are p zeros in both columns of the G matrix. If we show this explicitly we get

$$R^{(q)} \begin{bmatrix} a^{(p)} & a^{(p)\#} \\ 0^{q-p} & 0^{q-p} \end{bmatrix} = \begin{bmatrix} g_{0,0}^{(p)} & 0 \\ 0^{p-1} & 0^{p-1} \\ 0 & g_{p,1}^{(p)} \\ g_{p+1,0}^{(p)} & g_{p+1,1}^{(p)} \\ \cdot & \cdot \\ g_{q,0}^{(p)} & g_{q,1}^{(p)} \end{bmatrix}, \tag{10.98}$$

where $g_{0,0}^{(p)} = g_{p,1}^{(p)} = \sigma_p^2$. We now shift $a^{(p)\#}$ down by one row. That is, we make the following substitution

$$\begin{bmatrix} a^{(p)} & a^{(p)\#} \\ 0^{q-p} & 0^{q-p} \end{bmatrix} \rightarrow \begin{bmatrix} a^{(p)} & 0 \\ 0 & a^{(p)\#} \\ 0^{q-p-1} & 0^{q-p-1} \end{bmatrix}. \tag{10.99}$$

It is easily seen that we then obtain

$$R^{(q)} \begin{bmatrix} a^{(p)} & 0 \\ 0 & a^{(p)\#} \\ 0^{q-p-1} & 0^{q-p-1} \end{bmatrix} = \begin{bmatrix} g_{0,0}^{(p)} & g_{p+1,0}^{(p)} \\ 0^{p} & 0^{p} \\ g_{p+1,0}^{(p)} & g_{p,1}^{(p)} \\ \cdot & \cdot \\ g_{q,0}^{(p)} & g_{q,1}^{(p)} \end{bmatrix}, \tag{10.100}$$

where we noted that it follows from the symmetry of the $R^{(q)}$ matrix that the entry $g_{p+1,0}^{(p)}$ was appropriate for the first entry in the second column.

We now follow the procedure introduced for the Levinson algorithm and multiply on the right by $\begin{bmatrix} 1 & k_{p+1} \\ k_{p+1} & 1 \end{bmatrix}$ with $k_{p+1} = -\frac{g^{(p)}_{p+1,0}}{g^{(p)}_{p,1}}$. Again showing the zeros explicitly, we then have an equation of the form

$$R^{(q)} \begin{bmatrix} a^{(p+1)} & a^{(p+1)\#} \\ 0^{q-p-1} & 0^{q-p-1}b \end{bmatrix} = \begin{bmatrix} g^{(p+1)}_{0,0} & 0 \\ 0^p & 0^p \\ 0 & g^{(p+1)}_{p+1,1} \\ g^{(p+1)}_{p+2,0} & g^{(p+1)}_{p+2,1} \\ \cdot & \cdot \\ g^{(p+1)}_{q,0} & g^{(p+1)}_{q,1} \end{bmatrix}. \tag{10.101}$$

In other words, we have have obtained the equivalent of equation 10.96 for predictor order $p+1$ and completed the Schur recursion step. We can initialize the Schur algorithm with

$$G^{(0)} = \begin{bmatrix} g^{(0)}_{0,0} & g^{(0)}_{0,1} \\ g^{(0)}_{1,0} & g^{(0)}_{1,1} \\ \cdot & \cdot \\ g^{(0)}_{q,0} & g^{(0)}_{q,1} \end{bmatrix} = R^{(q)} \begin{bmatrix} a^{(0)} & a^{(0)\#} \\ 0^{q-1} & 0^{q-1} \end{bmatrix} = \begin{bmatrix} R_0 & R_0 \\ R_1 & R_1 \\ \cdot & \cdot \\ R_q & R_q \end{bmatrix}. \tag{10.102}$$

The main difference between the Schur and Levinson algorithms is that for the Schur algorithm both the numerator and denominator of the reflection coefficients, $k_p = -\frac{g^{(p-1)}_{p,0}}{g^{(p-1)}_{p-1,1}}$, are obtained through a recursion. That is, both numerator and denominator are elements of the generator matrix, which itself is updated by shifting a column and multiplying on the right by $\begin{bmatrix} 1 & k_p \\ k_p & 1 \end{bmatrix}$. In contrast, in the Levinson algorithm the numerator, δ_{p-1}, is obtained by means of an inner product. Such an inner product limits the parallelism allowed in hardware implementations.

We only used $R^{(q)}$ in the derivation of the Schur algorithm to show its validity. However, $R^{(q)}$ is not part of the computational method. The Schur recursion of the reflection coefficients k_p depends solely on the evolving $q \times 2$ generator matrix. Furthermore, we note that the first $p-1$ rows of the generator matrix $G^{(p)}$ do not need to be computed. We know that the first row is just the p'th row with reversed indices and that the next $p-2$ rows are zeros. Thus, to minimize the computational effort, we modify the generator matrix by chopping off the top row in each recursion step. The Schur algorithm can then be implemented with the following steps:

1. Set p=1. Form a $q \times 2$ generator matrix $G^{(0)} = \begin{bmatrix} R_0 & R_0 \\ R_1 & R_1 \\ \cdot & \cdot \\ R_q & R_q \end{bmatrix}$

2. Shift the second column of $G^{(p)}$ down by one row (discarding its last element) and eliminate the first row of the resulting matrix to obtain $\tilde{G}^{(p+1)}$.

3. Compute $k_{p+1} = -\frac{g^{(p)}_{1,0}}{g^{(p)}_{0,1}}$ (i.e., minus the ratio of the first row entries of $\tilde{G}^{(p+1)}$)

4. Obtain $G^{(p+1)} = \tilde{G}^{(p+1)} \begin{bmatrix} 1 & k_{p+1} \\ k_{p+1} & 1 \end{bmatrix}$

5. if $p < q$, set $p \to p+1$ and go back to 2.

This version of the Schur algorithm calculates only the reflection coefficients. We could compute the predictor coefficients simultaneously. However, we have already seen how to compute the predictor coefficients from the reflection coefficients using the step-up algorithm, and so the computation of the predictor coefficients, if needed can also be done afterward. Appendix F provides the Schur algorithm both with and without removal of the first row for each iteration. It also provides the step-up routine to obtain the predictor coefficients.

10.7 Problems

1. Let X_i be an iid Gaussian process with unit variance. In the following, consider the signal $V_i = X_i + 0.7X_{i-1}$.

   (a) Compute optimal second-order and fourth-order linear predictors.

   (b) Compute the prediction gain of the two predictors.

   (c) Compute the prediction gain of the infinite-order predictor.

   (d) Find the first-order entropy, the entropy rate, and the redundancy of the process.

   (e) Find a lower bound on the rate-distortion function for the process.

2. Show by means of inequalities that the solution of the Wiener-Hopf equations minimizes the prediction error variance σ_p^2.

3. (a) Define the relative entropy rate, a measure similar to the relative entropy measure to compare random processes rather than random variables.

   (b) Show that the relative entropy rate of a linear-prediction error process converges to that of an iid Gaussian process as the prediction order increases.

   (c) Under what conditions on the original process does the relative entropy rate vanish for a linear-prediction error process?

4. Prove that the Itakura-Saito criterion cannot be negative.

5. Consider a zero-mean signal with autocorrelation function $R(i)$ with optimal order-p predictor $A^{(p)}(z)$. Define $R_{A^{(p)}}(n) = \int \frac{1}{|A^{(p)}(\mathrm{e}^{j\omega}|^2} \mathrm{e}^{j\omega n} d\omega$, i.e., the autocorrelation of the AR process obtained by filtering a white sequence with the filter $1/A^{(p)}(z)$. The speech sound "a" can be modeled as a real autoregressive process with eight poles. The four poles located in the top half plane are $0.9631\, e^{j0.4571}$ $0.9643\, \mathrm{e}^{j0.7383}$, $0.9580\, \mathrm{e}^{j1.9478}$, and $0.9219\mathrm{e}^{j2.5840}$.

   (a) Prove that, upon normalization so that $R_{A^{(p)}}(0) = R(0)$, the first p autocorrelations of the AR model and signal are equal: $R_{A^{(p)}}(n) = R(n),\ 0 \le n \le p$.

(b) Consider a periodic sequence. Each period consists of the same random, white sequence of length P. Let $s_P(n)$ be the outcome of filtering this signal with $1/A(z)$. Relate the autocorrelation of $s_P(n)$ to the autocorrelation of the process obtained by filtering a white noise sequence by $1/A^{(p)}(z)$, i.e., the autocorrelation of $s_\infty(n)$.

(c) For the sound "a", compute the predictor coefficients, and the associated pole locations for a pitch period of 20, 80, and infinity (samples). Plot the associated spectra on a dB scale.

6. In this problem we consider the ideal scalar encoding of an original signal and two prediction residual signals, computed with optimal linear predictors of both infinite and finite dimension. The original signal, X_i, is a Gaussian process and we use the mean squared-error distortion measure, and we assume high resolution quantization. The power spectrum of the process is shown in figure 10.9.

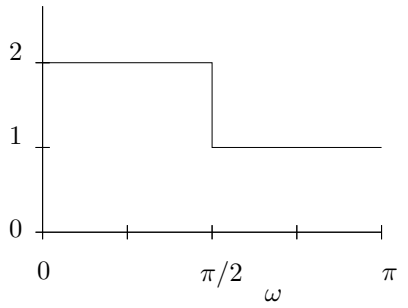

Figure 10.9: Power spectrum of the real process X_i.

(a) Explain if the following cases are reasonable, and if they are more favorable for scalar quantization: *i*) the differential entropy rate is larger than, *ii*) equal to, or *iii*) less than the first-order differential entropy?

(b) Assuming an entropy-constrained *scalar* quantizer operating directly on X_i, find and plot the lowest possible bit rate as a function of distortion.

(c) Assuming an ideal entropy-constrained scalar quantizer and an ideal (infinite-memory) predictor, find the lowest possible bit rate as a function of the distortion for the prediction residual. Add this relation to your plot.

(d) Find the optimal first-order predictor for the spectrum shown in figure 10.9.

(e) We filter X_i with the optimal first-order predictor you just derived. Find the lowest possible bit rate as a function of the distortion for the prediction residual. Add the result to your plot, making sure that the relative positions of the curves are correct. Provide a brief explanation for the relative positions.

(f) When we want to exploit linear prediction in the encoding of the original signal, we cannot simply quantize the linear-prediction residual and run this through the inverse of the prediction filter. Draw an effective architecture for predictive coding with a scalar quantizer and explain why it works. Comment on the performance of this system.

7. Show that equation 10.46 also holds for the Karhunen-Loève transform when the dimension approaches infinity.

8. The first reflection coefficient, k_1, is often used for classification of speech segments. Provide a qualitative argument based on the relation of this coefficient to spectral properties to justify this usage.

9. Consider a first order autoregressive process with coefficient ρ.

   (a) Determine the autocorrelation function for this process.

   (b) Determine the optimal predictors for orders 1, 2, and 3.

   (c) Determine the prediction gain for these predictors.

10. Consider a sequence of samples $-1, -1, 0, 0, 1, 1, 2, 2$.

    (a) Determine the first-order linear predictor with the covariance method.

    (b) Determine the first-order linear predictor with the autocorrelation method.

    (c) Evaluate the predictor errors and the stability of the predictors.

11. Consider a stationary Gaussian process that can be modeled as a zero-mean iid Gaussian process of unit variance driving a filter with poles at $0.95\mathrm{e}^{j\pi/4}$, $0.95\mathrm{e}^{-j\pi/4}$, $0.9\mathrm{e}^{j\pi/3}$, $0.9\mathrm{e}^{-j\pi/3}$.

    (a) Compute the variance of the process.

    (b) Compute the spectral flatness measure.

    (c) Given that you use scalar quantizers, up to how many bits per sample can be saved by using linear prediction?

    (d) Find a lower bound on the rate distortion function for the process.

    (e) Assuming that all you knew a-priori about the signal was the filter order and the fact that the filter is stable, propose a coder design using a predictor.

12. Consider a Gaussian signal with a power spectrum $\frac{2}{|1-0.8\mathrm{e}^{-j\omega}|^2}$. The squared-error criterion holds.

    (a) What is the first-order differential entropy of the signal?

    (b) Assuming you can use any model order, what is the optimal predictor for this signal?

    (c) What is the differential entropy rate of the signal?

    (d) Based on high-rate theory, design (provide the centroid locations and cell boundaries) a scalar, entropy-constrained quantizer that operates directly on the samples of the original signal, and has distortion 0.1. Estimate the rate of this quantizer.

    (e) To obtain better performance, you decide to employ prediction and scalar quantization for your coder. Provide a diagram showing the structure of your encoder and decoder.

    (f) What is the advantage (state your answer in bits) of using the predictor compared to not using the predictor? Motivate your answer.

13. Consider a zero-mean Markov process with two system states, A and B. In state A the observed samples are uncorrelated and have variance 100. The transitions

between the states have probability 0.001. In state B the observed samples are correlated and have covariance matrix

$$R = \begin{bmatrix} 100 & 99 \\ 99 & 100 \end{bmatrix};$$

and you have no knowledge about longer-term correlations.

(a) Is the Markov process stationary? Explain!

(b) Find the optimal predictor for the case that the system is in state A. Provide the prediction-error filter.

(c) Find the optimal predictor for the case that the system is in state B. Provide the prediction-error filter.

(d) (3) Find the optimal predictor for the overall Markov process (no knowledge about which state the system is in). You can neglect transition regions.

(e) Evaluate the prediction gains (ratio prediction-error variance and signal variance) for the three cases discussed before.

(f) You perform scalar quantization. Derive how many bits you can save for the three situations if the signal is Gaussian.

(g) Consider the case where you do not know the system state. Provide a straightforward coding strategy where you perform significantly better than 13d.

11

An Application: Speech Coding

11.1 Introduction

Speech is central in human communication. Thus, it is natural that the representation of the speech signal is a major application of source coding. A large literature on the topic exists; for tutorial overviews we refer to [115, 116, 117].

Speech coding provides good examples for the practical implementation of some of the procedures described in the previous chapters. We will use two guiding principles for our exploration of speech coding:

1. Minimize the redundancy in the transmitted representation of the signal.
2. Minimize the transmission of irrelevant information, i.e., minimize the the entropy rate of the transmitted signal given a certain threshold on the allowed distortion.

The removal of irrelevant information is most intuitive for coding of speech and audio at high rates. In this case, the irrelevant information is information about signal components that are below the hearing threshold or that are masked by other components of the signal. At lower rates, the audible distortion must be set to a certain level and information about signal components below this level is deemed irrelevant. It should be noted that the proper quantitative definition of perceptually accurate distortion criteria is the subject of ongoing research.

Before discussing how to remove redundancy and irrelevancy, we recall several concepts and define the conditions under which we operate. We consider segments of a digitized speech process to be a stationary discrete-time and discrete-amplitude process (vector or scalar) X_i. We recall that the entropy rate of a stationary process X_i is defined as

$$H_\infty(X_i) = \lim_{k\to\infty} \frac{1}{k} H_k(X_i) \tag{11.1}$$

and that this is a lower bound on the average bit rate per sample at which we can

represent the discrete process while ensuring exact reconstruction. Let L be the average bit allocation per sample of the digital signal. We define the **total redundancy rate** as

$$\rho_T(X_i) = L - H_\infty(X_i). \tag{11.2}$$

Thus, it is, at least in principle, possible to reduce the average rate of the signal, L, by the redundancy rate $\rho(X_i)$, and still be able to reconstruct the signal X_i exactly.

In our present analysis, which is a preamble for constructing speech coders, we assume that the speech signal will be coded in blocks. This assumption is valid for most practical speech coders. It is then convenient to decompose the total redundancy rate into two components, one related to the size of the coding blocks, and one related to the actual coding of these blocks. We define (inter-symbol) **redundancy rate** as

$$\rho(X_i) = H_m(X_i) - H_\infty(X_i). \tag{11.3}$$

This is the entity that is conventionally referred to as simply redundancy, as we have in earlier chapters. Inter-block redundancy results from not exploiting the statistical dependencies between the coding blocks. For simple sample-by-sample encodings, $m = 1$. In addition to the inter-symbol redundancy rate, we define the **intra-symbol redundancy rate** as

$$\rho_c(X_i) = L - H_m(X_i). \tag{11.4}$$

This represents the average redundancy within each coded block. Clearly, we have that $\rho_T(X_i) = \rho(X_i) + \rho_c(X_i)$.

Next, we define the irrelevancy rate. We consider a distortion threshold D. Furthermore, let $R(D)$ be the rate-distortion function, i.e., the lower bound on the rates that are achievable at a distortion D. We refer to $R(D)$ as the rate-distortion bound. The **irrelevancy rate** can then be defined as

$$\zeta(X_i) = H_\infty(X_i) - R(D). \tag{11.5}$$

Note that irrelevancy and total redundancy rates sum together to form the difference between the actual average bit rate, L, of a digital signal and the rate-distortion bound $R(D)$. Thus, the bit rate used by a blockwise digital representation of the signal can be divided into three components: the total redundancy rate, the irrelevancy rate, and the rate-distortion bound,

$$L = \rho(X_i) + \rho_c(X_i) + \zeta(X_i) + R(D). \tag{11.6}$$

To obtain the lowest possible rate (the rate-distortion bound), both the redundancy rate and irrelevancy rate must be removed from the overall rate.

In this book, we have discussed a variety of tools that are available to remove redundancy and irrelevancy. In the next section, we briefly discuss these tools from a speech-coding perspective. There-after, we motivate the most common architecture of speech coders, before discussing some particular techniques in some more detail.

11.2 A Source-Coding Toolbox

Table 11.1 provides an overview of a set of commonly used source-coding tools. In the development of speech coders, we will restrict our source-coding horizon to these tools.

In this section, we will briefly review the properties of these tools from the perspective of sampled speech signals. We will consider stationary speech segments only and will assume that a mean squared-error criterion applies. The discussion will sometimes implicitly assume that entropy-constrained quantization is used.

In our description of the tools we begin with discussing the removal of irrelevancy. Irrelevancy is removed by means of quantization and it is thus logical to introduce first simple **scalar quantization** of the random speech process. Scalar quantization has as main advantage its low complexity. However, the results of section 7.3.3 show that (at least for the high-rate Gaussian case) we cannot quite remove all the irrelevancy from the signal with a scalar quantizer. Consider a sampled signal with no inter-sample dependencies, i.e., with no inter-symbol redundancy. For such a signal we have that $H_1(X_i) = H_\infty(X_i)$, and thus that $\zeta(X_i) = H_1(X_i) - R(D)$. The entropy of the quantization indices of the scalar quantizer satisfies inequality 7.38, which becomes an equality for Gaussian processes. This shows that, in general, it is not not possible to remove all irrelevancy from the signal using a scalar quantizer.

Table 11.1: Tools for source coding

method	irrelevancy removal	redundancy removal
scalar quantization	X	
vector quantization	X	X
significance map	X	X
lossless coding		X
prediction		X
orthonormal transforms		X
modeling		X

In the case of speech, the signal samples are dependent, making it useful to consider the effect of scalar quantization on redundancy. In section 11.3, it will be shown (for the Gaussian case only) that the redundancy of the signal samples is decreased by scalar quantization. Thus, if I_i are the indices resulting from the scalar quantizer, then $\zeta(I_i) \leq \zeta(X_i)$. Importantly, in contrast to desirable redundancy removal, *the decrease in redundancy rate due to scalar quantization corresponds to a decrease in reconstruction accuracy.*

The fore-mentioned disadvantages of scalar quantization decrease with increasing dimension in **vector quantization**. Vector quantization is asymptotically optimal; with increasing vector dimension, the performance of an optimal vector quantizer approaches the rate-distortion bound. This implies that a vector quantizer can remove both redundancy and irrelevancy. Redundancy is removed since the dependencies between the vector components are reflected in the codebook vectors. Irrelevancy is removed because the precision of the description is reduced. Unfortunately, the computational complexity of unstructured vector quantization grows exponentially with the dimension of the vectors, for a given rate. This means that unstructured vector quantizers can be used in practice only if they have relatively low dimensionality .

While it is, at least in principle, possible to remove all (inter-sample) redundancy with methods other than vector quantization, this is not possible with the removal of ir-

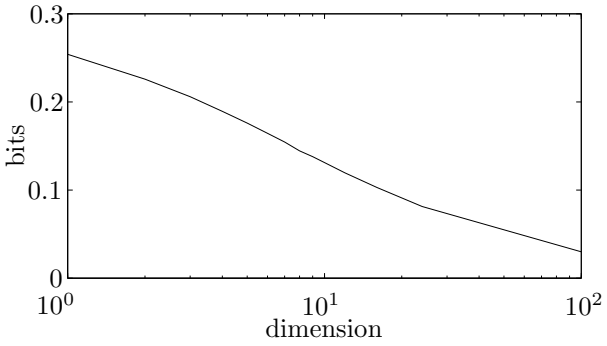

Figure 11.1: Rate penalty as a function of vector quantizer dimension for the case of a high-rate Gaussian source and the squared error criterion. As the dimension approaches infinity, this type of irrelevancy approaches zero.

relevancy. To improve the removal of irrelevancy over that of scalar quantization, no alternatives to increasing the dimension in vector quantization exist. Assuming no sample dependencies, figure 11.1 (see problem 3 of chapter 7) shows the difference between the entropy rate of the indices of the vector quantizer and the rate-distortion bound for the high-rate, Gaussian case. This cost is associated with the nonoptimal shape of the quantizer cells for low-dimensional quantizers. Compared to a scalar quantizer, a vector quantizer of dimension 10 halves the rate penalty for a given distortion.

Many signal representations can be thought of as expansions. That is, the signal is represented by coefficients that multiply functions (whether sinc functions, wavelets or other functions). Often, these representations are *sparse* in nature, i.e., only a small fraction of the expansion terms form a significant contribution to the signal energy. In such cases, it can be beneficial to use a **significance map** [69]. A significance map indicates which of the coefficients of the representation must be nonzero to obtain a good approximation of a particular signal. These nonzero coefficients are then coded with scalar or vector quantizers. The selection of the significance map is *a-posteriori*, i.e., based on a particular realization of the signal. The significance map removes both irrelevancy and redundancy and must be encoded.

Although better than scalar quantizers, which are useless in this respect, low-dimensional vector quantizers are generally not very effective for removing redundancy. This is definitely the case for voiced speech, which is nearly periodic and where dependencies can range over hundreds of samples. Thus, it is useful to explore other techniques than vector quantization to remove redundancy in speech coding.

As was discussed in more detail in chapter 5, **lossless coding** removes redundant information from a finite-alphabet signal. It is convenient to divide the lossless coders into coders that remove coding redundancy and coders that remove both coding and inter-symbol redundancy. The Huffman code removes only intra-symbol coding redundancy (note that by selecting larger blocks, this can be made to be a larger share of the overall redundancy). Universal codes such as the Ziv-Lempel algorithm, which make no assumptions about the statistics, remove both coding redundancy and inter-symbol redundancy. Lossless coding of the first class is commonly applied to the stream of

quantization indices in audio coding. However, lossless coding is not common in speech coding. The main reason for this is probably that it results in either a variable-rate bit stream, or a longer coding delay. In principle, lossless coding can also be applied directly to a digital speech signal (for example, a 16-bit per-sample representation). However, this leaves the speech signal in a representation from which irrelevancy cannot be removed in a convenient manner.

Prediction is commonly used in speech coding for the removal of inter-symbol redundancy (e.g., [72, 118]). For a stationary signal, sample-by-sample infinite-memory prediction can remove all redundancy from the signal. In practice, linear finite-memory prediction is used, with the linearity and memory constraints limiting the efficiency of redundancy removal. Prediction corresponds to a transform that modifies the distortion criterion, but the usage of closed-loop predictive coding and analysis-by-synthesis predictive coding results in the distortion criterion becoming invariant with the prediction operation.

Orthonormal transforms are particularly convenient when the mean squared-error criterion is used, since this criterion is not affected by the transform. Orthonormal transforms include block transforms, lapped transforms, and wavelet transforms. There are two different motivations for using orthonormal transforms in source coding: *i*) removal of inter-symbol redundancy by means of linear decorrelation and *ii*) minimization of the number of significant coefficients either a-priori (i.e., given the signal statistics) or a-posteriori (given a signal realization). The first motivation is most prevalent in speech coding.

For stationary Gaussian processes, linear predictors and filter-bank based coders (often referred to as **subband coders**) can asymptotically remove all inter-symbol redundancy (which is a function of the linear correlations between the signal samples) under idealized conditions. For the linear predictor, all inter-symbol redundancy is removed when the prediction memory approaches infinity, and for the subband coder this is the case when the subbands are ideal and their number approaches infinity. There are some differences in the methods, even under ideal conditions. The subband structure is independent of the signal statistics, whereas the predictor depends on the signal statistics. The (differential) entropy rate distribution over the subbands varies with the signal statistics, whereas the (differential) entropy of all prediction-error samples is identical.

11.3 Designing a Coder Architecture

In the previous section, we saw that some source-coding tools are most useful for redundancy removal, whereas others are most useful for irrelevancy removal. Large-dimensional vector quantization, which can perform both is computationally prohibitively expensive when used as the sole technique for speech coding. In this section, we assume that we use a scalar quantizer to remove the irrelevancy. Under this constraint, forced upon us by computational constraints, we then search for the best coder architecture, i.e., for the best configuration of the scalar quantizer and redundancy removal methods. While our arguments are based on the usage of scalar quantizers, the same arguments can be made for low-dimensional vector quantizers.

To remove redundancy from the signal, we can distinguish between two classes: on

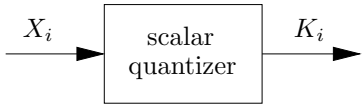

Figure 11.2: Basic quantizer structure.

the one hand signal processing of orthonormal transforms and (linear) prediction, and on the other hand lossless coding techniques exemplified by Huffman and Ziv-Lempel codes. It is interesting to evaluate which techniques are more useful to code a speech signal.

To start the discussion on coder architecture, we note that, in general, an index stream that is the output of a quantizer, in general, contains intra-symbol redundancy. This means that, in general, it is beneficial to have a lossless coder at the output of the quantizer. Furthermore, as was already mentioned in section 11.2, it is clear that the output from a lossless coder is a representation that is not amenable to quantization, so it is not useful to use any kind of lossless coder prior to quantization. Similarly, it is generally not sensible to apply signal processing techniques to the stream of indices. Thus, only one question remains: is it beneficial to use signal processing techniques to remove inter-symbol redundancy from the quantizer input or is it equally good, or better to remove inter-symbol redundancy from the quantizer output?

We answer this question for the simple case of a high-rate coding system operating on a Gaussian process X_i and a mean squared-error criterion. Application of the quantizer directly to the process results in a stream of indices, K_i, with first-order entropy $H_1(K_i)$, as shown in figure 11.2. The inter-symbol redundancy $H_1(K_i) - H_\infty(K_i)$ for the process K_i is, in general, not zero. A lossless coder can be used to remove this redundancy, as well as the coding redundancy and the lower bound on the resulting average bit rate is thus $H_\infty(K_i)$. For our case (high-rate scalar quantization, Gaussian process, mean squared-error criterion), the relation between the distortion D and the rate $H_\infty(K_i)$ can be written as

$$\begin{aligned} H_\infty(K_i) &= H_1(K_i) - (H_1(K_i) - H_\infty(K_i)) \\ &= H_1(K_i) - (H_1(K_i) - H(K_i|[K_{i-1}, K_{i-2}, \cdots])) \\ &= H_1(K_i) - \lim_{m\to\infty} I(K_i; [K_{i-1}, \cdots, K_{i-m}]) \\ &= h_1(X_i) - \frac{1}{2}\log(12D) - \lim_{m\to\infty} I(K_i; [K_{i-1}, \cdots, K_{i-m}]), \end{aligned} \quad (11.7)$$

where we used equation 7.33 and theorem 5.

Next, we look at the case where the inter-symbol redundancy is removed prior to the quantizer, as shown in figure 11.3. This removal would be performed by a signal processing procedure such as an orthonormal filter bank or (closed-loop) prediction. To make the argument, we assume that the inter-symbol redundancy removal is perfect and that the associated transform does not affect the distortion criterion. The relation

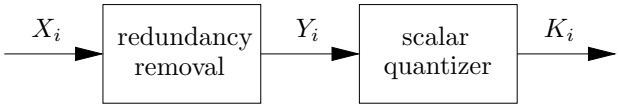

Figure 11.3: Quantizer with front-end removal of redundancy.

between distortion and the entropy rate of the indices is now:

$$\begin{aligned} H_\infty(K_i) &= h_1(Y_i) - \frac{1}{2}\log(12D) \\ &= h_\infty(X_i) - \frac{1}{2}\log(12D) \\ &= h_1(X_i) - \frac{1}{2}\log(12D) - \lim_{m\to\infty} I(X_i;[X_{i-1},\cdots,X_{i-m}]), \end{aligned} \quad (11.8)$$

where we used equation 7.33 and the continuous-alphabet equivalent of theorem 5.

The only difference between equations 11.8 and 11.7 is in the mutual information term. In the case of signal processing prior to quantization, equation 11.8, this term describes the mutual information of a continuous random variable X_i and its past, whereas in equation 11.7, it refers to the mutual information between the quantized equivalents of the variables X_i. To show that it is beneficial to remove the inter-symbol redundancy first, it suffices to show that $I(X_i;[X_{i-1},\cdots,X_{i-m}]) \geq I(K_i;[K_{i-1},\cdots,K_{i-m}])$. We use Jensen's inequality, which states that if $g(x)$ is convex (i.e., its second derivative is positive), then

$$\mathrm{E}[g(x)] \geq g(\mathrm{E}[x]), \quad (11.9)$$

to prove this inequality. (Jensen's inequality itself is the subject of problem 3.) In our proof, we consider a sequence of length $m+1$. We label with V_{k_i} the Voronoi region for x_i, given a particular past $[x_{i-1},\cdots,x_{i-m}]$. Using Jensen's inequality and the fact that a function $t\log t$ of t is convex, it is simple to show that scalar quantization of the samples of the process X_i decreases the mutual information between X_i and the past signal. For notational simplicity we write $\tilde{X} = [X_{i-1},\cdots,X_{i-m}]$ and $\tilde{x} = [x_{i-1},\cdots,x_{i-m}]$. We furthermore omit the usual subscripts from the densities and probability mass functions ($f_{X_i}(x_i)$ becomes $f(x_i)$ and $p_{K_i}(k_i)$ becomes $p(k_i)$). We quantize X_i first with the scalar quantizer. We can write

$$\begin{aligned} I(X_i;\tilde{X}) &= \int_{\mathbb{R}^m}\int_{\mathbb{R}} f(x_i,\tilde{x})\log\left(\frac{f(\tilde{x},x_i)}{f(\tilde{x})f(x_i)}\right)dx_i d\tilde{x} \\ &= \int_{\mathbb{R}^m}\sum_{k_i\in\mathcal{A}} p(k_i)\int_{V_{k_i}} f(\tilde{x}|x_i)\frac{f(x_i)}{p(k_i)}\log\left(\frac{f(\tilde{x}|x_i)}{f(\tilde{x})}\right)dx_i d\tilde{x}. \end{aligned} \quad (11.10)$$

From Jensen's inequality we have that

$$\begin{aligned} &\int_{V_{k_i}} f(\tilde{x}|x_i)\frac{f(x_i)}{p(k_i)}\log\left(f(\tilde{x}|x_i)\right)dx_i \\ &\quad= \mathrm{E}[f(\tilde{x}|x_i)\log\left(f(\tilde{x}|x_i)\right)|x_i\in V_{k_i}] \\ &\quad\geq \mathrm{E}[f(\tilde{x}|x_i\in V_{k_i}]\log\left(\mathrm{E}[f(\tilde{x}|x_i)|x_i\in V_{k_i}]\right) \\ &\quad= \int_{V_{k_i}}\frac{f(\tilde{x},x_i)}{p(k_i)}dx_i\log\left(\int_{V_{k_i}}\frac{f(\tilde{x},x_i)}{p(k_i)}dx_i\right). \end{aligned} \quad (11.11)$$

Using inequality 11.11 in equation 11.10, we obtain

$$\begin{aligned} I(X_i;\tilde{X}) &\geq \int_{\mathbb{R}^m} \sum_{k_i \in \mathcal{A}} \left(\int_{V_{k_i}} f(\tilde{x}, x_i) dx_i \right) \log \left(\frac{\int_{V_{k_i}} f(\tilde{x}, x_i) dx_i}{f(\tilde{x}) p(k_i)} \right) d\tilde{x} \\ &= \int_{\mathbb{R}^m} \sum_{k_i \in \mathcal{A}} f(\tilde{x}|k_i) p(k_i) \log \left(\frac{f(\tilde{x}|k_i) p(k_i)}{f(\tilde{x}) p(k_i)} \right) d\tilde{x} \\ &= I(K_i;\tilde{X}). \end{aligned} \tag{11.12}$$

Next, we can use the same reasoning for the quantization of the components of the vector $\tilde{X}$ to obtain

$$\begin{aligned} I(X_i;[X_{i-1},\cdots,X_{i-m}]) &\geq I(K_i;[X_{i-1},\cdots,X_{i-m}]) \\ &\geq I(K_i;[K_{i-1}\cdots K_{i-m}]), \end{aligned} \tag{11.13}$$

which completes our derivation.

Thus, we have seen that, at least under particular conditions, the removal of redundancy by means of signal processing prior to quantization can indeed reduce the coded bit rate. This result is quite intuitive: if we destroy part of the signal structure by quantization, then this destroyed structure cannot be used to obtain a coding gain through redundancy removal. It should furthermore be mentioned that the signal processing procedures for removing inter-symbol redundancy tend to be of relatively low complexity. This is particularly so since the algorithms are usually optimized for the signal at hand.

To summarize, a good architecture for scalar quantization consists of the following operator steps: *i*) inter-symbol-redundancy removal by means of signal processing, *ii*) irrelevancy removal by means of scalar quantization, and finally *iii*) coding-redundancy removal by means of lossless coding. In practice, the first step is approximated by orthonormal transforms or by prediction, and this is described in the next section.

11.4 Practical Speech Coding Approaches

In this section, we discuss practical speech coding procedures for real-world speech signals. In the application of these methods, the speech signal is usually interpreted as a signal that is stationary over short time intervals (these intervals are usually considered to be 20 ms or so.)

Speech coders are generally classified by the method used for the signal-processing-based inter-symbol-redundancy removal: prediction or orthonormal filter banks. It was mentioned in section 11.2 that these methods provide similar asymptotic performance for stationary signals. However, in practice, where we have to deal with finite filter lengths, filter banks with finite filter bandwidths, and nonstationary signals, the relative performance of the inter-symbol-redundancy removal methods depends on the signal properties. We will discuss the advantages and disadvantages of linear prediction and filter banks in the context of speech coding in the next two subsections.

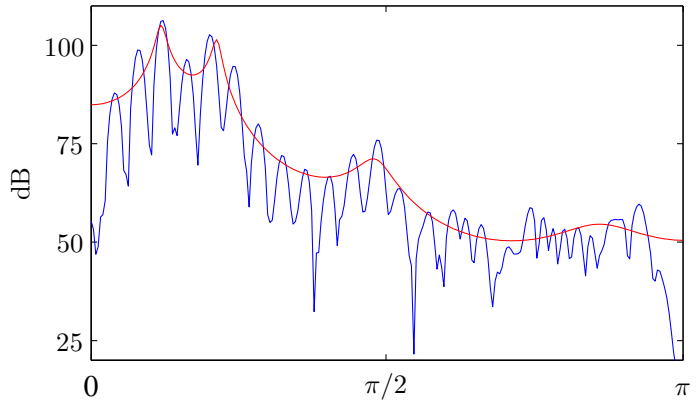

Figure 11.4: A power spectrum of voiced speech for a male speaker (27.5 ms Hann window) and a tenth order linear-prediction fit resulting from minimizing the prediction error.

11.4.1 Linear Prediction-Based Speech Coders

Predictors are generally computed using the optimized using the methods described in section 10.3. This computation is performed on a block-wise basis and, for the case of forward prediction (section 10.4.3), the predictor description must be included in the coded signal description. A pre-condition for making the adaptation practical is that the inter-symbol redundancy removed must be more than the bit rate required to describe the side-information for the predictor in a practically adequate manner.

Short-Term Predictors

A practical low-order linear predictor can remove the empirical inter-symbol redundancy quite effectively and at a reasonable cost in bit rate. Simple models of the vocal tract, which account only for dominant aspects, result in all-pole transfer-functions [119], which explains why adaptive linear prediction can perform efficient decorrelation. In practice, for an 8 kHz sampled speech signal, a tenth order linear predictor performs well. This is consistent with the notion that the power spectrum of speech displays up to four formants (spectral resonances), each with a center frequency and a bandwidth, and a tilt, for a total of 9 variables. The real filter has ten complex poles, which come in five complex conjugate pairs. This allows ten degrees of freedom, which is consistent with the nine degrees of freedom we estimated from a simple vocal-tract model. Since the predictor describing redundancy introduced by the vocal tract describes correlations on a relatively short time-scale, it is usually referred to as the **short-term** predictor, in contrast to a second type of predictor, which will be discussed later.

Since the speech signal is considered as nonstationary, the optimal predictor for a particular speech segment is estimated from speech data. A statistical motivation for the estimation procedure and a practical implementation are described in sections 10.2.1 and 10.3. It is particularly interesting to compare the properties of the resulting esti-

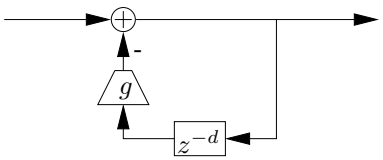

Figure 11.5: Pitch predictor structure. d is the delay corresponding to a pitch period.

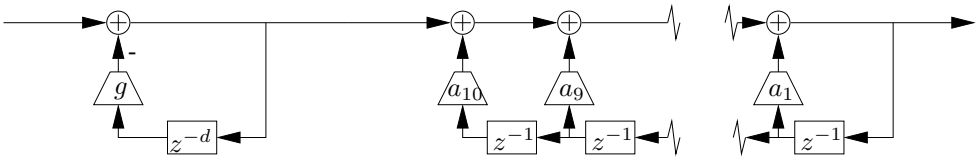

Figure 11.6: Linear-prediction-based synthesis structure including both pitch predictor and short-term predictor.

mate with the properties of the human auditory system, as was done in section 10.2.3. It was noted there that the fitting accuracy is very consistent with human perception: in the power-spectral domain the estimate locally fits the shape of the spectral envelope and globally maintains a constant signal-to-distortion ratio. A typical fit of the linear-predictor spectrum to a short-term speech power spectrum is shown in figure 11.4.

Pitch (Long-Term) Predictors

The above discussion covered short-term predictors. As said, this predictor typically has 10 coefficients and it describes the so-called short-term correlations. The short-term predictor cannot remove the redundancy associated with the nearly periodic character of the voiced speech signal. This periodic character of the voiced sounds is created by oscillations of the vocal cords; vowels are examples of voiced sounds. In low-rate linear-prediction based coders, so-called **long-term predictors** or **pitch predictors** are commonly used to remove the redundancy associated with the nearly periodic character of the signal. The corresponding synthesis structure is shown in figure 11.5. The long-term predictor generally has only one filter tap and both the optimal tap location and the optimal single coefficient are usually estimated using analysis-by-synthesis [120]. It is common practice to optimize the two predictors sequentially: the short-term predictor parameters are estimated first, and the pitch predictor there-after. The corresponding filters are simply concatenated, as illustrated in figure 11.6. For completeness, we mention that a computationally simpler alternative to the pitch predictor is the commonly used and closely-related adaptive codebook structure [121].

The analysis-by-synthesis structure shown in figure 10.7 is easily modified to include the structure with both pitch predictor and short-term predictor. Let the transfer function of the operator in figure 11.6 be denoted by $H(z)$. Then figure 11.7 illustrates the corresponding analysis-by-synthesis structure. (The figure is simplified from figure 10.6.)

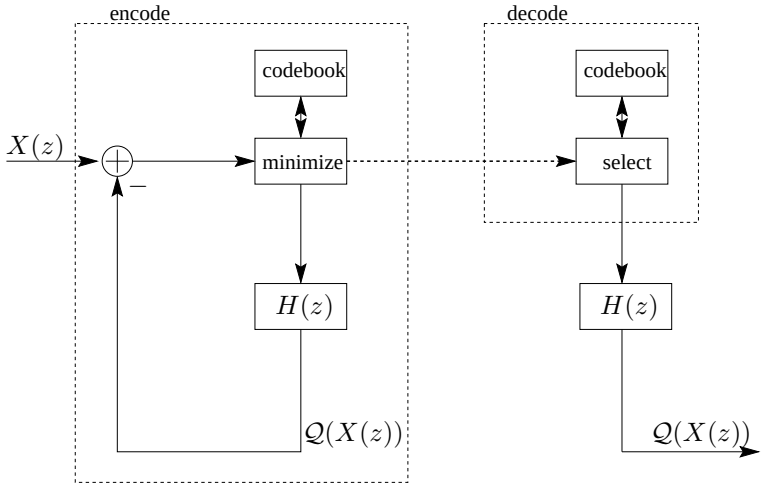

Figure 11.7: General analysis-by-synthesis structure for a synthesis filter with transfer function $H(z)$. This transfer function can include a pitch structure. The loop within the encode box indicates the quantizer selection process for each sample.

Some Disadvantages of Linear Prediction

It can be argued that prediction simplifies codebook design. Since the prediction-error samples are approximately independent and identically distributed (and assuming that the marginal densities are independent of the speech sound) the statistics of the prediction-error sample blocks are essentially independent of the predictor parameters (except for a gain factor). This suggests that a single codebook for the prediction-error blocks can perform well, independently of the power spectral envelope of the segment. Indeed, good performance using this method is obtained in practice with the ubiquitous code-excited linear-prediction (CELP) coder [84, 85]. However, although commonly used, this intuitive argumentation for using a single codebook is not consistent with what we know from rate-distortion theory.

The first inconsistency is related to the space-filling advantage of vector quantization. Analysis-by-synthesis methods are methods that effectively create speech-domain codebook adaptively, and a vector from these adapting codebooks is selected to represent the target vector: the speech minus the zero-input response. While the vectors of the codebook are adapted to model the dependencies of the vectors, through the filtering operation, the local arrangement of the centroids is not selected to be optimal or near optimal. In other words, the shape of the Voronoi regions is generally far from optimal.

The second inconsistency is related to reverse water filling. For Gaussian processes, reverse waterfilling arguments (see section 6.6.2) tell us that the spectrum of the optimal reconstruction signal is often quite different from that of the original signal. However, in analysis-by-synthesis methods, this is generally not accounted for. In principle, the problem can be mitigated by adapting the codebook or by modifying the synthesis filter [122].

Another disadvantage of linear predictive coding of speech, not motivated by rate-distortion theory, is related to long-term prediction. In general, voiced speech has high

periodicity, but its onsets can be abrupt, and the signal shape can change relatively rapidly. It can be shown[1] that this nonstationary and-yet highly periodic character is not efficiently represented with the all-pole structure used for synthesis with a long-term predictor. At low bit rates, this leads to a relatively noisy character of the reconstructed signal.

Enhancements to Linear Prediction

The performance of linear predictive coding is generally enhanced with both perceptual weighting and with postfiltering. In perceptual weighting the analysis-by-synthesis structure is modified so as to account for the relative importance of different spectral regions in auditory perception. In general, this implies that spectral regions with high power are down-weighted (in these regions, masking is strong) and spectral regions with low power carry increased weight (in these regions masking is generally less strong). Postfiltering [123, 124] is used to shape the reconstructed speech for enhanced perceptual quality. In general, postfiltering involves increasing the power of regions that already have strong power and attenuation of regions of low power. Usually, postfiltering attenuates regions of low signal-to-noise ratio.

The phenomenon of reverse waterfilling provides an additional motivation for postfiltering. We know from the theory of reverse waterfilling (section 6.6.2) that the optimal reconstruction, in general, does not have the same power spectrum as the original signal. At low rates, low amplitude frequency bands should be missing. This motivates the attenuation of of low-power frequency regions by the postfilter.

It is interesting to note that prediction has an obvious path for further improvement in performance. In prediction, the optimality of the quantized excitation samples for the synthesis structure is dependent on neighboring samples. However, in most current coders (including idealized coders that reach the asymptotic limit), the selected quantization value of an excitation sample depends on past samples only. In other words, the excitation samples are not optimized simultaneously and the excitation is nonoptimal. We can get closer to optimality by using delayed-decision methods (e.g., [62]). Such techniques also work for vector excitation methods [125, 126]. In general, such methods require additional computational effort. It is natural to interpret them as a particular form of constrained vector quantization.

Commercial Success of Linear Prediction

Most speech coding standards developed between 1980 and 2000 are based on linear prediction [2]. The European (GSM) and American (TIA) standards for digital mobile telephony are commercially important examples. The success of this method for redundancy removal can be attributed to the advantages listed above: *i*) the inter-symbol redundancy associated with the correlations introduced by the vocal-tract is effectively removed with a low-order linear predictor, *ii*) the estimation of the linear-predictor parameters is consistent with hearing, and *iii*) long-term linear prediction performs rea-

[1] Using a pitch-synchronous signal expansion, the synthesis filter corresponding to the pitch predictor is a first-order AR filter. This time-variance frequency-variance product of the impulse response of such a filter is infinite and displays thus very poor time-frequency resolution.

sonably well in the removal of long-term correlations related to the near periodicity of the voiced speech signal (particularly when a short delay is required).

11.4.2 Nonlinear Prediction and Speech Coding

Nonlinear predictors can exploit all known dependencies between samples. This implies that optimal nonlinear predictors can remove all redundancy from a signal, and this suggests that usage even of nonoptimal nonlinear predictors could lead to improved coding performance. Yet, they are rarely used in current speech coding algorithms. Some possible reasons for this are difficulties with their estimation, high computational complexity, difficulty with encoding their structure in forward-adaptive schemes, and the relatively good performance of linear prediction for speech signals. This relatively good performance is associated with the fact that the vocal-tract is well described by an all-pole filter structure. However, a similar argument is not valid for the signal exciting the vocal tract.

The excitation signal for the vocal tract is a pressure wave that emanates from the vocal cords. During voiced speech (as in vowels), the vocal cords oscillate resulting in a pressure wave with a periodic character. This wave is modified (colored) by the vocal-tract and radiated into space where it is picked up by our microphone and converted into a sampled signal. After removing short-term correlations (the spectral envelope) from this signal using a short-term predictor, a pulse train is observed.

The fact that the signal becomes a pulse train after flattening of the envelope indicates that linear correlations cannot describe all sample dependencies. This is easily illustrated using an ideal filter bank operating on a (time-invariant) pulse train. Let us delay the outputs of each filter by a different amount prior to adding them back together. Clearly, the sample correlations of the input signal are identical to those before the signal. Yet, we have modified the structure of the signal. Thus, there must be dependencies in the signal that are not characterized by linear correlations, and that, therefore, cannot be removed by linear prediction. Note also that the existence of such dependencies also implies that the signal is not Gaussian.

It can be concluded that it is reasonable to use nonlinear prediction to remove the redundancy associated with the "long-term" structure in the speech signal that is created by the oscillation of the vocal cords. That nonlinear prediction is very effective for the long-term structure in speech was confirmed in [127].

11.4.3 Filter-Bank Based Coders

In speech coding, most filter bank-based methods can be included in one of two classes: *i*) filter bank coders with relatively few bands that are always time-invariant, *ii*) filter bank-based coders with an adaptive band structure. In most implementations the reconstruction filter bank is not a perfect inverse of the analysis filter bank.

Low-Resolution Filter bank Based Coders

The usage of low-resolution filter banks is easily motivated for Gaussian stationary signals. For now, we ignore the fact that the Gaussian assumption is not quite right (cf. section 11.4.2). (In fact, it probably forms the actual historical motivation for using the subband coder structure in speech.) According to this motivation, as was discussed in chapter 9, the samples of the individual filters of an ideal critically sampled filter bank are uncorrelated within the band and between bands. This means that distortion is reduced when these coefficients (filter outputs) are quantized rather than the original signals samples (assuming scalar quantization). That this is true for speech signals in practice is a small measure of vindication for using our initial assumptions. The individual bands can be encoded using predictive coders, as, for example, in [128].

In the period between 1975 and 1985, filter banks were commonly used in both research and commercial products (e.g., [128, 129, 130, 131]). In the following years, their usage declined, although they were applied to some specialized applications (e.g., [132]). In the years there-after more and more coders were based on linear prediction.

In speech coding, low-resolution filter banks have two obvious disadvantages compared to linear prediction: *i*) low-resolution subband coders cannot remove redundancy associated with the nearly periodic structure of voiced speech and *ii*) low-resolution filterbanks do not describe the spectral structure of speech as accurately as all-pole filters . The first disadvantage is associated with the inability of the low-resolution filter bank to resolve the harmonic structure present in the power spectrum. The second disadvantage is associated with the fact that the vocal tract instills a particular type of structure on speech spectra that is not reflected by the filter bank.

Orthonormal filter banks do have a strong advantage over linear prediction in that the mean squared-error criterion can be used for quantization of the transformed signals. However, the empirical short-term statistics of the signals in the individual bands vary with the speech sound, which means that different codebooks (at least separate scaling factors) are necessary for different sounds. This is an implementation disadvantage compared to linear prediction, where single codebooks have been found to work well in practice (although they are not optimal, as was discussed earlier).

High-Resolution Filter-Bank Based Coders

High-resolution filter-bank based coders are known as sinusoidal and waveform interpolation coders. Such coders are commonly used at bit rates below 4 kb/s. While it is natural to interpret sinusoidal (e.g., [133, 134]) and waveform interpolation coders (e.g. [135, 136]) as high-resolution orthonormal filter banks or approximations there-of, the filter bank interpretation and terminology is rarely used in the literature.

The significant signal structure that remains within the bands of low-resolution filter banks provides a good reason to simply increase the number of bands. However, for such high-resolution filter banks we can also give several motivations that are not based on the Gaussianity assumption: *i*) narrow-band filter banks generally lead to significant energy concentration for voiced speech, which makes coding with significance maps attractive and *ii*) properly selected adaptive narrow-band filter banks can render each band low-pass in character (again for voiced speech), even when critically sampled, facilitating

simple redundancy removal techniques such as down-sampling and prediction.

Sinusoidal Coding

The relation between sinusoidal coders and high-resolution filter banks is based on the notion that the outputs from narrow-band filters are conveniently represented as modulated sine waves. Let us analyze this for the continuous signal case. Consider a filter bank with N filters with center spacing $2\pi/N$ radians. For the k'th filter with impulse response $h_k(t)$, the amplitude modulation for the filter output can be written as

$$a_k(t) = \exp(\frac{-j2\pi kt}{N}) \int_{-\infty}^{\infty} s(t-t')h_k(t')dt'. \tag{11.14}$$

If the filters $h_k(t)$ satisfy

$$\sum_k h_k(t) = \delta(t), \tag{11.15}$$

which is true for ideal bandpass filters, the original speech signal can be written as

$$\begin{aligned} s(t) &= \sum_k \int_{-\infty}^{\infty} s(t-t')h_k(t')dt' \\ &= \sum_k a_k(t) \exp(\frac{j2\pi kt}{N}), \end{aligned} \tag{11.16}$$

which clearly displays the "sinusoidal" interpretation of the filter bank. This representation is particularly natural for voiced speech. Due to the nearly periodic nature of voiced speech, only a subset of the sinusoids have a significant amplitude at any time instant. Within the frequency resolution of the filter bank, these sinusoids are harmonically related. This implies that the significance map is described efficiently by simply encoding the fundamental frequency. (For unvoiced speech, the sinusoidal representation does not provide these advantages.)

The usage of a signficance map in sinusoidal coding implies that the basis (more generally, frame) functions have a support that is significantly longer than a single pitch period, thus resolving the harmonics. While this is desirable to resolve harmonics, it generally leads to problems at sharp onsets of periodic signal segments, which are clearly identified by the human auditory system, but not by the long-support basis (frame) functions.

The effectiveness of the sinusoidal method depends on the particularities of the implementation. The filters must be practical digital filters and this leads to analysis and synthesis filter banks that fall into the general class of discrete Gabor transforms. The frame functions for these transforms are windowed exponentials. For coding, it is desirable that the filter banks are orthonormal and that the windows on the exponentials are smooth. The first property ensures that the distortion criterion is invariant with the transform and that the number of variables to be coded is small (which would not be the case for a redundant representation). The smoothness property ensures that quantization does not result in discontinuities of the reconstructed signal and increases the frequency resolution of the coding system. We note that human perception is sensitive

to discontinuities in an acoustic signal even when these are of no significant consequence in the commonly used distortion measures based on squared error. Thus, the usage smooth frame functions facilitates the efficiency of distortion measures that are imperfect descriptions of human perception.

Unfortunately, as was discussed in section 9.3.5, the above list of desirable properties can only be reached for oversampling. It can be argued that the usage of analysis and synthesis methods that do not guarantee perfect reconstruction in sinusoidal coding areas is a result of the requirement of oversampling for perfect reconstruction using Gabor frame functions. Particularly at the low bit rates where sinusoidal coding was employed, where the quantization error dominates, perfect reconstruction by the filter bank was not a high priority.

Waveform Interpolation

An alternative narrowband filter bank approach is waveform interpolation (WI). In this method, the support of the basis (frame) functions is sufficiently short to resolve the onsets of voiced speech. Sparseness can be obtained by a second-stage processing procedure [137], but the procedure is in this respect more flexible than the sinusoidal coding method. In WI the speech signal is thought of as being based on an evolving characteristic waveform $s(t, \phi)$, where t is time and ϕ displays the waveform. $s(t, \phi)$ is periodic in ϕ with a fixed period of 2π. The speech signal is then $s(t, \phi(t))$, where $\phi(t)$ is a particular pitch track.

If $s(t, \phi)$ is represented as a Fourier series,

$$s(t, \phi) = \sum_k a_k(t) \exp(\frac{jk\phi}{N}), \tag{11.17}$$

a similar representation as is used for sinusoidal coders is obtained. However, the notion that equation 11.17 displays one pitch cycle along ϕ leads to a particular filter bank resolution. In WI, the filter bank is selected such that its frequency resolution is exactly the fundamental frequency of the signal. No significance map is used in WI. Since the filter bank is tailored to the signal properties, it can be interpreted as an example of a best-basis procedure.

The filter bank interpretation has been exploited directly in WI coders. In one implementation, the speech signal is time-warped, so that a perfect-reconstruction fixed filter bank can be employed [138]. To improve the time-frequency resolution, the modulated lapped transform (see section 9.3.4) has been used. In contrast to the Gabor transform, this transform allows critical sampling with smooth windows. However, the interpretation of the signal along the ϕ axis is less straightforward.

Because of the particular frequency resolution of the filter bank WI interpolation, the bands of this filter bank (the coefficients of a the basis/frame functions for a particular pitch-synchronous channel) are particularly convenient for the separation into a noise-like component and a nearly-periodic component. For voiced speech, most bands are essentially low-pass in character, facilitating further down sampling even when a critically sampled filter bank is used. In contrast, for noise-like (wide band) signals, each signal band will have a flat power spectral density. By applying straightforward a two-band filter bank (with high-pass and low-pass filters) for each of the bands, signals

corresponding to the noise and the nearly-periodic components of the speech signal can be separated. In combination with distortion criteria that correspond to a more accurate representation of perception, this facilitates the removal of irrelevancy by using separate quantizers for the voiced and unvoiced components.

11.5 Problems

1. Consider the analysis-by-synthesis structure of figure 11.7. Consider the block-wise selection of an excitation vector from a codebook.

   (a) Let the impulse response of the synthesis structure be $h_0, h_1, \cdots$ and the original signal vector be $\mathbf{s}$. Determine the criterion for the selection of a codebook vector $\mathbf{c}_i$, where i is the index of the vector in the codebook. Write the criterion in matrix notation.

   (b) Consider the case where the vector has a separate gain factor. That is, the excitation is of the form $\lambda_k \mathbf{c}_i$ where k is the index in the gain codebook. Derive an expression for the ideal gain λ for a given excitation vector.

   (c) Simplify the criterion for the excitation vector, under the assumption of optimal gain.

   (d) Derive the simplest expression for the criterion to be used in the gain quantizer given the excitation vector.

2. In this problem we review and contrast the basic properties of prediction-based coding schemes and filter-bank based coding schemes. We consider Gaussian processes only.

   (a) Is a block-by-block discrete Fourier transform a linear operator? Is adaptive linear predictive filtering, also operating on a block-by-block basis, a linear operator?

   (b) Starting from the expression for redundancy (see equations), show that an "ideal" predictor (explain what "ideal" is) can remove all redundancy from a stationary Gaussian process.

   (c) Provide a qualitative reasoning that shows that, asymptotically, the discrete Fourier transform can also remove all redundancy from a stationary Gaussian process.

   (d) The Fourier transform is a unitary transform that implements an approximate minimization of $\prod_{i=1}^{k} \sigma_i^2$. Show that such a minimization (under the constraint of unitarity) corresponds to concentration of signal energy.

   (e) Argue that, for a stationary signal, prediction results in a uniform variance of the prediction residual samples.

3. Prove Jensen's inequality for convex functions (equation 11.9).

4. Like some famous entrepeneurs, you and your friend try to sell your product before it is ready. You have a customer who wants you to encode a stationary signal with the power spectrum shown in figure 11.8. You do the better sales job and tell the customer it will take about 1/6 bits to half the distortion per sample (assuming squared-error criterion). Unfortunately, your friend had earlier told the same customer that it will take about 1/2 bit to half the distortion per sample.

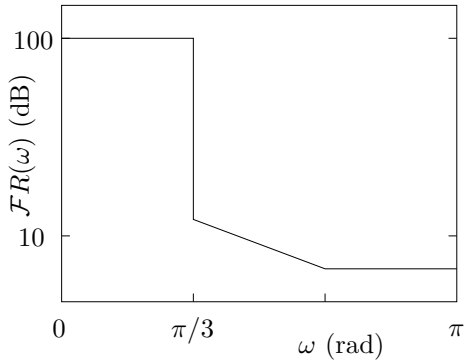

Figure 11.8: Power spectrum for problem 4. Note that the vertical axis is logarithmic.

(a) Show that both results are correct (or very close to it) under certain conditions. Make sure to list all assumptions made.

(b) *Outline* the principles of a good high-rate coder that you would design for this signal. Assume the distortion is a squared-error criterion and use scalar quantizers. No further constraints are imposed.

5. We usually consider Euclidean distance based distortion criteria. For such criteria and high-rate conditions, a scalar quantizer can not perform as well as a vector quantizer. These conclusions change dramatically if we change the distortion criterion.

   (a) Describe a distortion criterion for which scalar quantization can perform as well as vector quantization for certain densities. What must the densities be?

   (b) For the criterion you selected and for a Gaussian distribution, would the Karhunen-Loeve transform be useful for improving the efficiency of scalar quantization? Why?

6. Lossless codes can be seen as the result of minimizing the mean codeword length $L = \sum_{x\in\mathcal{A}} p_X(x)l(x)$ subject to the constraint that the code is uniquely decodable. Such *entropy coding* leads to a mean codeword length that is close to the entropy. The minimization of the mean-codeword-length cost criterion can lead to individual codewords that are very long and, thus, to long delay. If delay is more important than mean rate, alternative cost criteria can be considered. In the following, you can first assume that the codeword lengths $l(x)$ can take any value on the real line (as was done in the derivation of the Shannon code) and upon completion of the derivation introduce suitable rounding to make the code discrete.

   (a) Find a codeword length that is optimal (or close to -) for a uniquely decodable code and the cost $L = \max_{x\in\mathcal{A}}(l(x))$. Motivate your solution.

   (b) For a cost $L = \sum_{x\in\mathcal{A}} p_X(x)\mathrm{e}^{al(x)}$ find a near-optimal codeword length as a function of the probability (use methods similar as those used to motivate the Shannon code). Naturally, your code must be uniquely decodable. We call the resulting code an *exp-length* code.

(c) Your objective is a good quantizer for a given set of data, a squared error criterion, and an *exp-length* code. Define a complete iteration step of an iterative quantizer optimization method that, for each iteration, improves the quantizer that assumes an *exp-length* based code (rather than constrained-entropy or constrained-resolution codes). (The algorithm is a Lloyd algorithm.)

7. Consider a continuous-amplitude, discrete-time, Gaussian, white-noise (flat power spectrum) process, X_i, sampled at 8 kHz. The process X_i is upsampled to 16 kHz, then low-pass filtered with 4 kHz cut-off (assume an ideal filter) with as output the process V_i. Consider the squared-error criterion.

   (a) Provide the rate-distortion function for the process V_i.

   (b) Design a *practical* efficient coding system for the process V_i that gets close to the rate-distortion function.

   (c) Design a practical and efficient coding system for the process V_i under the constraint that the overall delay of your coder (consider delay in *both* encoder and decoder!) design is zero (0) samples. As always motivate your choice.

Appendix A

Probability Theory Basics and Notation

In this appendix, we review the basic probability theory that is assumed known throughout the book. We start with the notion of **sample space** or **alphabet**, which is a collection of **elementary events**, also called elementary experimental outcomes, or simply points.

The discrete values that an output sample of an A/D (analog-to-digital) converter can take form an alphabet. For the output of a 16-bit linear A/D converter, the sample space is all integers from -32768 to 32767. Alphabets can contain a finite number of elements, a countable infinite number of elements (for example the set of integer numbers, $\mathcal{Z}$) or an uncountably infinite number of elements (such as the set of points in any interval of the real line $\mathbb{R}$). The number of points in an alphabet is called its **cardinality**. Alphabets are often referred to as **discrete** or **continuous**. A discrete alphabet has countably finite or infinite elements while a continuous alphabet has uncountably infinite elements.

An **event** is a subset of the sample space (this is consistent with a sample being an elementary event). An event has a particular **probability**. The basic axioms of probability theory are that the probability must satisfy three properties:

1. The probability of any event is in $[0, 1]$.
2. The probability of the entire alphabet is unity.
3. The probability of the countable union of disjoint events (subsets of the sample space which share no elements) is the sum of their probabilities.

We now have defined the three aspects of a **probability space**: the sample space, the event space, and the probability of the events.

For our example of the A/D output, an event can be the subset of positive values of the alphabet. This event can have a probability of 0.5, for example.

Next, we define the random variable. A **random variable** is a function that maps each point in the sample space onto a finite real number. A condition is that the set of

points in sample space corresponding to the random variable being less than any real number should be an event. The condition, together with the defined properties for the probability, ensures that the probability that a random variable takes a certain set of values is defined.

The function $F_X(x) = P(X \leq x)$ is called the **cumulative distribution function**, often abbreviated to distribution function. For **discrete-alphabet random variables** or simply discrete random variables the distribution function has a staircase pattern, while for continuous-alphabet random variables the continuous distribution is continuous. Random variables outside these two classes have a mixed-alphabet. For continuous alphabets, the derivative of the distribution function, the **probability density**, is generally used.

A random discrete-time signal, also called a discrete-time **process**, consists of a **sequence** of random variables. Let us consider an infinite sequence. It makes it easier to analyze such a sequence if we can assume that the statistics of the sequence are the same for all random variables of the sequence. An infinite sequence for which all probability functions (joint and/or marginal) are independent of the time index, is called **stationary**. We extend this definition to a finite sequence, to mean that all probability functions are independent of the time index, as far as they apply to the finite sequence.

A process is considered **wide-sense stationary** (also called **weakly stationary**) if the first and second order moments, $\mathrm{E}[X_i]$ and $R_{i,j} \equiv \mathrm{E}[X_i X_j]$ are invariant with time shifts (i.e., $\mathrm{E}[X_i] = \mathrm{E}[X_{i+q}]$ and $R_{ij} = R_{i+q,j+q}$).

A special case of a stationary sequence is a sequence with **independent, and identically distributed (iid)** variables, which are defined by the property

$$F_{X_1,X_2}(x_1, x_2) = F_{X_1}(x_1)\, F_{X_2}(x_2), \tag{A.1}$$

where $F_{X_1,X_2}(x_1, x_2) = P(X_1 \leq x_1, X_2 \leq x_2)$. Property A.1 simplifies the evaluation of information-theoretic measures, and iid variables are, therefore, often used in models of natural signals.

For a stationary process that is also **ergodic**, the time average of the functions of the random variables approach the expected value of the random variable, i.e.,

$$\mathrm{E}[f(X_i)] = \lim_{k\to\infty} \frac{1}{k} \sum_{n=0}^{n=k-1} f(X_n). \tag{A.2}$$

In our applications, we will not need to define ergodicity in a more general context. However, we note that a process can be ergodic without being stationary.

Finally some words on notation. Following a common convention, random variables are capitalized, whereas events are not. Thus $P(X = x)$ is to be interpreted as the probability that the random variable X takes the value x, or simply the probability of the outcome x for an event. A discrete process is an indexed set of variables $\{X_i : i \in \mathcal{K}\}$ where $\mathcal{K} \subset \mathcal{Z}$. Except in a few cases where this is obvious from the context, or explicitly stated, the index of the process indicates time, $\mathcal{K} = \mathcal{Z}$, and the process can be interpreted as a signal. In general, we distinguish a process from a variable by labeling it with a subscript; X_i denotes a process. The superscript of a variable can mean either the dimensionality or the power of a variable. Generally, the letter k is reserved for the dimensionality and so x^k usually indicates a k-dimensional vector variable and x^{kT} the transpose of this vector variable.

Appendix B

The Lagrange Multiplier Method

The Lagrange-multiplier method aids in the solution of constrained optimization problems. Consider a function $f(x^k)$ and a constraint $g(x^k) = 0$ where f and g are functions of the k-dimensional vector $x^k = [x_0, \cdots, x_{k-1}]^T$ and where f and g have continuous first partial derivatives. The Lagrange multiplier method finds the stationary points (a minimum, a maximum, or an inflection point) of $f(x^k)$ under a constraint of the form

$$g(x^k) = 0. \tag{B.1}$$

In the more general form, the method also deals with inequality constraints of the form

$$g(x^k) \leq 0, \tag{B.2}$$

In the first two sections, the Lagrange multiplier method is derived for equality constraints. The second section concludes with a summary of the method. The third subsection describes how the method can be used for inequality constraints. Before starting with the derivations, we note that the commonly heard interpretation that the Lagrange multiplier method consists of the addition of a zero component $\lambda g(x^k)$ which has a nonzero derivative is an oversimplification.

B.1 Equality Constraints: Derivation Using Differentials

We denote partial derivatives as $f_i(x^k) = \frac{\partial f(x^k)}{\partial x_i}$ which we sometimes abbreviate to f_i. At a stationary point of $f(x^k)$ we have that

$$df = f_0 dx_0 + f_1 dx_1 + \cdots + f_{k-1} dx_{k-1} = 0. \tag{B.3}$$

Without the constraint, the infinitesimal quantities dx_i can vary *independently* and for a stationary point all $f_i(x^k)$ must vanish. However with the constraint $g(x^k) = 0$ this

is not the case. Then, any changes in the dx_i are dependent, and this dependency is defined by

$$dg = g_0 dx_0 + g_1 dx_1 + \cdots + g_{k-1} dx_{k-1} = 0. \tag{B.4}$$

Assuming $g_q \neq 0$, equation B.4 implies that

$$dx_q = -\sum_{i \neq q} \frac{g_i}{g_q} dx_i \tag{B.5}$$

(the sum is over all dimensions except q) and, thus, that

$$df = \sum_i (f_i - \frac{f_q}{g_q} g_i) dx_i. \tag{B.6}$$

Equation B.6 allows independent variation of all dx_i, since the coefficient of dx_q is always zero. It is seen that for a stationary point x^k

$$\frac{f_0(x^k)}{g_0(x^k)} = \frac{f_1(x^k)}{g_1(x^k)} = \cdots = \frac{f_{k-1}(x^k)}{g_{k-1}(x^k)}. \tag{B.7}$$

To find the constrained extrema of $f(x^k)$, i.e. to find the k components of x^k for the constrained extrema, we can solve the equations B.7 ($k-1$ equations) and equation B.1.

This solution can be rewritten in a simpler manner. Let us first define the **Lagrange multiplier**:

$$\lambda \equiv \frac{f_0(x^k)}{g_0(x^k)} = \frac{f_1(x^k)}{g_1(x^k)} = \cdots = \frac{f_{k-1}(x^k)}{g_{k-1}(x^k)}, \tag{B.8}$$

where x^k is assumed to be a stationary point. Then, equations B.7 become

$$\begin{aligned} f_0(x^k) - \lambda g_0(x^k) &= 0 \\ &\vdots \\ f_{k-1}(x^k) - \lambda g_{k-1}(x^k) &= 0. \end{aligned} \tag{B.9}$$

We now have $k+1$ variables: the k components of x^k and the Lagrange multiplier λ. We solve for these variables with the k equations B.9 and the constraint equation B.1 to obtain a stationary point (if any exists).

The last method for finding the constrained extrema can be interpreted as finding a stationary point of the function $f(x^k) + \lambda g(x^k)$ or, alternatively, of the function $f(x^k) + \lambda(g(x^k) - c)$ with the constraint equation B.1 as an auxiliary equation, making the Lagrange multiplier method easy to remember. This formulation can also be derived in a more direct manner, as is shown in the next subsection.

B.2 Equality Constraints: Direct Derivation

The direct derivation of the Lagrange multiplier method can be described by three steps:

1. The problem is to find a stationary point of $f(x^k)$ under the constraint $g(x^k) = 0$.

2. An identical problem is to find the stationary point of $f(x^k) + \lambda g(x^k)$ under the constraint that $g(x^k) = 0$, where λ is an arbitrary constant.

3. If we can find a λ such that the stationary point of $f(x^k) + \lambda g(x^k)$ also satisfies $g(x^k) = 0$, then we have found the desired stationary point.

Thus, the Lagrange multiplier method consists of:

1. construct the function $f(x^k) + \lambda g(x^k)$

2. differentiate this function towards the components of x^k,

3. solve the resulting k equations $B.9$ and the constraint $g(x^k) = 0$ for the vector x^k (the additional unknown λ can be discarded).

B.3 Inequality Constraints

B.3.1 General Case

For simplicity, we consider the scalar case for the inequality constraint. Let us consider the case where we want to *minimize* $f(x^k)$ under the constraint that $g(x^k) \leq 0$. We distinguish two cases:

1. The case where $f(x^k)$ has a minimum with $g(x^k) < 0$. The solution is straightforward.

2. The case where $g(x^k) = 0$. In this case, $\frac{\partial f(x)}{\partial x^k}^T dx^k \geq 0$ for all admissable dx^k. From the constraint, it follows that the set of admissable dx^k satisfies $\frac{\partial g(x^k)}{\partial x^k}^T dx^k \leq 0$. For this case, a solution must satisfy $\frac{\partial f(x)}{\partial x^k} + \lambda \frac{\partial g(x^k)}{\partial x^k} = 0$ and $g(x^k) = 0$.

Both cases can be written as one optimization problem by setting constraints on the multiplier λ: minimize

$$f(x^k) + \lambda g(x^k) \tag{B.10}$$

with the constraint

$$g(x^k) \leq 0 \tag{B.11}$$

and where

$$\lambda = 0, \quad g(x^k) < 0, \tag{B.12}$$
$$\lambda \geq 0, \quad g(x^k) = 0. \tag{B.13}$$

Equation B.12 corresponds to case 1 and equation B.13 corresponds to case 2.

B.3.2 Common Source Coding Inequality Constraints

For source coding, it is common to have to minimize a function $f(x^k)$ subject to a set of inequality constraints of the form $x_i < c_i, i \in \{0, \cdots k-1\}$ (where $x^k = [x_0, \cdots, x_{k-1}]$. (The function $f(x^k)$ generally contains a Lagrange equality constraint.) In this case, the problem can be formulated as: minimize

$$f(x^k) + \sum_i \lambda_i x_i \tag{B.14}$$

with the constraints

$$x_i - c_i \leq 0, \qquad i \in \{0, \cdots, k-1\} \tag{B.15}$$

where

$$\lambda_i = 0, \quad x_i - c_i < 0, \tag{B.16}$$

$$\lambda_i \geq 0, \quad x_i - c_i = 0. \tag{B.17}$$

Differentiating equation B.14 and setting it to zero results in

$$\frac{\partial f(x^k)}{\partial x_i} + \lambda_i = 0 \tag{B.18}$$

When combined with equations B.16 and B.17 this gives us the set of equations

$$\frac{\partial f(x^k)}{\partial x_i} = 0, \qquad x_i - c_i < 0, \tag{B.19}$$

$$\frac{\partial f(x^k)}{\partial x_i} \leq 0, \qquad x_i - c_i = 0. \tag{B.20}$$

which are straightforward to solve if a minimum exists.

Appendix C

Variational Calculus

The objective of variational calculus is to find a function, $f(x^k)$, which results in a **functional**, $\eta(f(\cdot))$, (a function of this function) to be invariant with small variations in the function $f(\cdot)$. For these trajectories of $f(\cdot)$ the functional has a stationary value. Typically, the functional is an integral of the function:

$$\eta(f(\cdot)) = \int H(f(x^k))dx^k, \tag{C.1}$$

where $H(\cdot)$ is some function. Now consider a perturbation of $f(\cdot)$ by a function $\epsilon w(x^k)$ (with continuous second partial derivatives) where ϵ is small:

$$\begin{aligned} \eta(f(\cdot)) &= \int H(f(x^k) + \epsilon w(x^k))dx^k \\ &= \int H(f(x^k))dx^k + \epsilon \int \frac{\partial H(f(x^k))}{\partial f(x^k)} w(x^k)dx^k. \end{aligned} \tag{C.2}$$

The function $f(\cdot)$ represents a stationary point of the functional $\eta(f(\cdot))$ if

$$\frac{\partial H(f(x^k))}{\partial f(x^k)} = 0 \tag{C.3}$$

on the trajectory of $f(x^k)$. Equation C.3 is a simple form of the **Euler-Lagrange equation**. This equation is easily generalized to the case where $H(\cdot)$ is also a function of the partial derivatives of $f(x^k)$.

Generalization of variational calculus to the case with an integral constraint of the form

$$\int G(f(x^k))dx^k = c, \tag{C.4}$$

where c is a constant, is straightforward. It is most convenient to use the same line of reasoning as in section B.2. Thus,

1. The problem is to find a stationary point of $\int H(f(x^k))dx^k$ under the constraint $\int G(f(x^k))dx^k = 0$.

2. An identical problem is to find the stationary point of $\int H(f(x^k))dx^k + \lambda \int G(f(x^k))dx^k$.
3. If we can find a λ such that the stationary point of $\int H(f(x^k))dx^k + \lambda \int G(f(x^k))dx^k$ also satisfies $\int G(f(x^k))dx^k = 0$, then we have found the desired stationary point.

Appendix D

Determinant Inequalities

Consider a symmetric positive-definite matrix A of dimension k. We prove that $\prod_{i\in\{1,\cdots,k}A_{ii} \geq \det(A)$ holds for symmetric positive definite matrices with a recursion. We then prove that $\mathrm{tr}(A) \geq \det(A)$.

Proof that $\prod_i A_{ii} \geq \det(A)$

To prove that $\prod_i A_{ii} \geq \det(A)$, let us partition the matrix A into a submatrix B, a column vector b, and a scalar c. Furthermore, we select an x such that $Bx = -b$. Then we have

$$\begin{aligned}
\det\begin{bmatrix} B & b \\ b^H & c \end{bmatrix} &= \det\left(\begin{bmatrix} I & 0 \\ x^H & 1 \end{bmatrix}\begin{bmatrix} B & b \\ b^H & c \end{bmatrix}\begin{bmatrix} I & x \\ 0 & 1 \end{bmatrix}\right) \\
&= \det\begin{bmatrix} B & Bx+b \\ x^HB+b^H & x^HBx+x^Hb+b^Hx+c \end{bmatrix} \\
&= \det\begin{bmatrix} B & 0 \\ 0 & b^Hx+c \end{bmatrix} \\
&= \det(B)\,(b^Hx+c) \\
&\leq \det(B)\,c,
\end{aligned}$$

where we used that $x^HBx = -x^Hb$, so that x^Hb must be negative. We prove our statement by decomposing the matrix B in a similar manner, and so-on until we have only scalars.

Proof that $\frac{1}{k}\mathrm{tr}(A) \geq \det(A)^{\frac{1}{k}}$

Given the first results of this Appendix, to prove that $\frac{1}{k}\mathrm{tr}(A) \geq \det(A)^{\frac{1}{k}}$ for a symmetric positive-definite matrix, it suffices to prove $\frac{1}{k}\mathrm{tr}(A) \geq (\prod_i(A_{ii}))^{\frac{1}{k}}$. The latter relation is the well-known arithmetic-geometric inequality, which states that for real positive

numbers $\{a_i\}_{i\in\{1,\cdots,k\}}$

$$\frac{1}{k}\sum_{i=1}^{k} a_i \geq (\prod_{i=1}^{k} a_i)^{\frac{1}{k}}. \tag{D.1}$$

We start with the simple case $k = 2$ and consider

$$c = \sqrt{a_1} - \sqrt{a_2}. \tag{D.2}$$

Since c is real we have

$$0 \leq c^2 = a_1 + a_2 - 2\sqrt{a_1 a_2} \tag{D.3}$$

Thus it follows that

$$\frac{1}{2}a_1 + a_2 \geq \sqrt{a_1 a_2}, \tag{D.4}$$

which completes the proof for $k = 2$.

We now outline the generalization of the proof to the case where the number of dimensions, k, is arbitrary. We start with proving that equation D.1 holds for $k = 2^n$ with n a postive integer. This can be done with mathematical induction. Given that the relation holds for $k = 2^n$, it is easy to show that it also holds for $k = 2^{n+1}$. Since we have already shown that it holds for $k = 1$, we have then proven that it holds for all $k = 2^n$ with n a positive integer.

Next, we show that if equation D.1 holds for k then it also holds for $k - 1$. To this purpose, we choose

$$a_k = \frac{1}{k-1}\sum_{i=1}^{k-1} a_i \tag{D.5}$$

With this convenient choice for a_k, the left-hand side of equation D.1 becomes

$$\frac{1}{k}\sum_{i=1}^{k} a_i = \frac{1}{k-1}\sum_{i=1}^{k-1} a_i. \tag{D.6}$$

Equation D.1 can then be written as

$$\frac{1}{k-1}\sum_{i=1}^{k-1} a_i = (\frac{1}{k-1}\sum_{i=1}^{k-1} a_i \prod_{i=1}^{k-1} a_i)^{\frac{1}{k}}, \tag{D.7}$$

which is equivalent to the desired $k - 1$ relation:

$$\frac{1}{k-1}\sum_{i=1}^{k-1} a_i = (\prod_{i=1}^{k-1} a_i)^{\frac{1}{k-1}}. \tag{D.8}$$

Thus, since equation D.1 holds for all $k = 2^n$ with n a positive integer and for $k - 1$ if it holds for k, it must hold for all k (k positive integers).

Appendix E

An All-Pass Transfer Function

Let $H(z)$ be an all-pass transfer function of the form

$$H(z) \equiv \frac{A^{(p)}(z)}{A^{(p)\#}(z)}, \tag{E.1}$$

where $A^{(p)}(z) = 1 + a_1^{(p)} z^{-1} \cdots + a_p^{(p)} z^{-p}$ is minimum phase and where $A^{(p)\#}(z) = z^{-p} + a_1^{(p)} z^{-p+1} \cdots + a_p^{(p)}$.

Let $\alpha_i^{(p)}, i = 1 \cdots p$ denote the poles of $A_p(z)$. Since $A^{(p)}(z)$ is minimum phase we have that $|\alpha_i^{(p)}| < 1, i = 1 \cdots p$. It is convenient to write $H(z)$ as follows:

$$H(z) = z^{-p} \frac{\prod_{i=1}^{i=p} (z - \alpha_i^{(p)})}{\prod_{i=1}^{i=p} (1 - \alpha_i^{(p)*} z)}. \tag{E.2}$$

Using polar coordinates we note that

$$|z - \alpha_i^{(p)}|^2 - |1 - \alpha_i^{(p)*} z|^2 = (1 - |z|^2)(1 - |\alpha_i^{(p)}|^2). \tag{E.3}$$

From the minimum-phase property of $A^{(p)}(z)$ it then follows that

$$|z - \alpha_i^{(p)}|^2 - |1 - \alpha_i^{(p)*} z|^2 \begin{array}{lll} > & 0, & |z| > 1 \\ = & 0, & |z| = 1 \\ < & 0, & |z| < 1 \end{array} \tag{E.4}$$

and, therefore,

$$\frac{|z - \alpha_i^{(p)}|}{|1 - \alpha_i^{(p)*} z|} \begin{array}{lll} > & 1, & |z| > 1, \\ = & 1, & |z| = 1, \\ < & 1, & |z| < 1 \end{array} \tag{E.5}$$

Obviously, the same equality and inequalities are valid for the complete all-pass filter of equation E.2. Thus, it follows that

$$|H(z)| \begin{array}{lll} > & 1, & |z| > 1, \\ = & 1, & |z| = 1, \\ < & 1, & |z| < 1. \end{array} \tag{E.6}$$

Appendix F

The Levinson, Schur, and Step-Up Algorithms in Matlab

In this Appendix, simple Matlab implementations of the Levinson and Schur algorithms for obtaining the predictor coefficients are given. Note that these implementations are only for demonstration purposes; Matlab is not very efficient when using explicit loops.

```
function[a,k]=levinson(R,order)
% Levinson algorithm
% a: predictor coefficients
% k: reflection coefficients
% R: first column of autocorrelation matrix
% order: predictor order
k = [];
a = 1;                                  % initialize predictor
sigma2 = R(1);                          % initialize prediction error variance
for i=1:order
 at = flipud(a);                        % flip the predictor vector
 delta = R(2:(size(a,1)+1))' * at;      % perform inner product
 k = [k, -delta / sigma2];              % compute k(i)
 a = [[a;0],[0;at ]] * [1; k(i)];       % compute new predictor vector
 sigma2 = sigma2 + delta * k(i);        % update prediction error variance
%sigma2 = sigma2 * (1-k(i)^2);          % altern update prediction error var
end

function[k]=schuri(R,order)
% inefficient Schur algorithm;
% this version retains the size of the generator matrix G
% k: reflection coefficients
% R: first column autocorrelation matrix
% order: desired dimension of k
k = [];
G = [R(1:order+1), R(1:order+1)];           % initialize generator matrix
```

```
for i=1:order
 Gtilde = [G(:,1),[G(i+1,1);G(1:order,2)]];  % perform shift on second column
 k = [k,  -Gtilde(i+1,1) /Gtilde(i+1,2)  ];  % compute k(i)
 G = Gtilde * [[1,k(i)];[k(i),1]];           % multiply by 2x2 "theta" matrix
end

function[k]=schure(R,order)
% efficient Schur algorithm
% this version removes the top from the generator matrix G
% k: reflection coefficients
% R: first column autocorrelation matrix
% order: desired dimension of k
k = [];
G = [R(1:order+1), R(1:order+1)];            % initialize generator matrix
for i=1:order
 q = order+2-i;                              % 1 extra because we count from 1
 Gtilde = [G(2:q,1),G(1:q-1,2)];             % perform shift; chop off top
 k = [k,  -Gtilde(1,1)/Gtilde(1,2)];         % compute k(i)
 G = Gtilde * [[1,k(i)];[k(i),1]];           % multiply by 2x2 "theta" matrix
end

function [a]=stepup(k,order)
% step-up algorithm
% a: predictor coefficients
% k: reflection coefficients
% order: prediction order (less or equal to size(k,1))
a = 1;                                       % initialize predictor
for i=1:order
 a = [[a;0],[0;flipud(a)]] * [1;k(i)];   % compute new predictor vector
end
```

Bibliography

[1] M. Bosi, K. Brandenburg, S. Quackenbush, L. Fielder, K. Akagiri, H. Fuchs, M. Diets, J. Herre, G. Davidson, and Y. Oikawa., "ISO/IEC MPEG-2 Advanced Audio Coding," *J. Audio Eng. Soc.*, vol. 45, no. 10, pp. 789–812, 1997.

[2] R. V. Cox, "Speech coding standards," in *Speech Coding and Synthesis* (W. B. Kleijn and K. K. Paliwal, eds.), pp. 49–78, Elsevier, 1995.

[3] C. E. Shannon, "A mathematical theory of communications I," *Bell Syst. Techn. J.*, vol. 27, pp. 379–423, 1948.

[4] C. E. Shannon, "A mathematical theory of communications II," *Bell Syst. Techn. J.*, vol. 27, pp. 623–656, 1948.

[5] F. W. Sears and G. Salinger, *Thermodynamics, Kinetic Theory and Statistical Thermodynamics*. London: Addison-Wesley, 1975.

[6] L. Brillouin, *Science and Information Theory*. New York: Academic Press, 1962.

[7] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. New York: Wiley, 1991.

[8] J. Karush, "A simple proof of an inequality of McMillan," *IRE Trans. Inform. Theory*, vol. IT-7, p. 118, 1961.

[9] A. Papoulis, *Probability, Random Variables and Stochastic Processes, 2nd ed.* New York: McGraw-Hill, 1984.

[10] G. Strang, *Linear algebra and its applications*. Orlando: Academic Press, 1980.

[11] U. Grenander and G. Szego, *Toeplitz Forms and their Applications*. New York: Chelsea Publishing Company, 1984.

[12] R. M. Gray, "Toeplitz and circulant matrices: a review," technical report, Stanford University, 1971.

[13] G. Fant, *The acoustic theory of speech production*. The Hague: Mouton, 1970.

[14] O. Vasicek, "A test for normality based on sample entropy," *J. Royal Statistical Soc., Ser. B*, vol. 38, no. 1, pp. 54–59, 1976.

[15] J. Beirlant, E. Dudewicz, L. Györfi, and E. van der Meulen, "Nonparametric entropy estimation: an overview," *Intern. J. Math. Stat. Sci.*, vol. 1, no. 6, pp. 17–30, 1997.

[16] I. S. Gradshteyn and I. M. Ryzhik, *Table of Integrals, Series and Products*. New York: Academic Press, 1980.

[17] M. Abramowitz and I. Stegun, *Handbook of mathematical Functions*. New York: Dover Publications, 1965.

[18] J. Burg, *Maximum entropy spectral analysis*. PhD thesis, Stanford University, 1975.

[19] D. Percival and A. Walden, *Spectral Analysis for Physical Applications*. Cambridge: Cambridge University Press, 1993.

[20] D. A. Huffman, "A method for the construction of minimum-redundancy codes," *Proc. IRE*, vol. 40, no. 9, pp. 1098–1101, 1952.

[21] R. Pasco, *Source coding algorithms for fast data compression.* PhD thesis, Stanford University, 1976.

[22] J. J. Rissanen, "Generalized Kraft inequality and arithmetic coding," *IBM J. Res. Devel.*, vol. 20, pp. 198–203, 1976.

[23] I. A. Witten, R. M. Neal, and J. Cleary, "Arithmetic coding for data compression," *Comm. Assoc. Computing Machinery*, vol. 30, no. 6, pp. 520–540, 1987.

[24] A. Gersho and R. M. Gray, *Vector Quantization and Signal Compression.* Dordrecht, Holland: Kluwer Academic Publishers, 1991.

[25] R. Blahut, "Computation of channel capacity and rate distortion functions," *Trans. Inform Theory*, no. 18, pp. 410–421, 1972.

[26] W. Bennett, "Spectra of quantized signals," *Bell Syst. Tech. J.*, vol. 27, pp. 446–472, 1948.

[27] J. Johnston, "Transform coding of audio signals using perceptual noise criteria," *IEEE J. Select. Areas Comm.*, vol. 6, no. 2, pp. 314–323, 1988.

[28] R. Gray, "Gauss mixture vector quantization," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, (Salt Lake City), pp. 1769–1772, 2001.

[29] R. M. Gray and T. Linder, "Mismatch in high-rate entropy-constrained vector quantization," *IEEE Trans. Inform. Theory*, vol. 49, no. 5, pp. 1204–1217, 2003.

[30] J. Ziv, "Universal quantization," *IEEE Trans. Inform. Theory*, vol. 31, pp. 344–347, 1985.

[31] T. Eriksson, *Vector quantization in speech coding.* PhD thesis, Chalmers University, Goteborg, Sweden, 1996.

[32] S. Na and D. Neuhoff, "Bennett's integral for vector quantizers," *IEEE Trans. Inform. Theory*, vol. 41, no. 4, pp. 886–900, 1995.

[33] A. Gersho, "Asymptotically optimal block quantization," *IEEE Trans. Inform. Theory*, vol. 25, pp. 373–380, 1979.

[34] N. Ueda and R. Nakano, "A new learning approach based on equidistortion principle for optimal vector quantizer desing," in *Neural Networks for Signal Processing III, Proc. IEEE-SP workshop*, pp. 362–371, 1993.

[35] J. Conway and N. Sloane, "A lower bound on the average error of vector quantizers," *IEEE Trans. Inform. Theory*, vol. 31, pp. 106–109, 1985.

[36] T. Lookabough and R. Gray, "High-resolution theory and the vector quantizer advantage," *IEEE Trans. Inform. Theory*, vol. IT-35, no. 5, pp. 1020–1033, 1989.

[37] P. Hedelin and J. Skoglund, "Vector quantization based on Gaussian mixture models," *IEEE Trans. Speech Audio Process.*, vol. 8, no. 4, pp. 385 – 401, 2000.

[38] A. Subramaniam and B. D. Rao, "PDF optimized parametric vector quantization of speech line spectral freuencies," in *IEEE Speech Coding Workshop*, (Delavan, WI), pp. 87–89, 2000.

[39] A. Subramaniam and B. Rao, "Speech LSF quantization with rate independent complexity, bit scalability, and learning," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, (Istanbul), pp. 705–708, 2001.

[40] T. Z. Shabestary and P. Hedelin, "Spectral quantization by companding," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, vol. 1, pp. 641–644, 2002.

[41] J. Conway and N. Sloane, "Fast quantizing and decoding algorithms for lattice quantizers and codes," *IEEE Trans. Inform. Theory*, vol. 28, pp. 227–232, 1982.

[42] K. Sayood and S. Blankenau, "A fast quantization algorithm for lattice quantizer design," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, (New York), pp. 1168–1171, 1988.

[43] J. H. Conway and N. J. A. Sloane, *Sphere Packings, Lattices, and Groups.* New York: Springer Verlag, 1993.

[44] T. Senoo and B. Girod, "Vector quantization for entropy coding of image subbands," *IEEE Trans. Image Process.*, vol. 28, pp. 227–232, 1982.

[45] R. Zamir and M. Feder, "On lattice quantization noise," *IEEE Trans. Inform. Theory*, vol. 42, no. 4, pp. 1152–1159, 1996.

[46] J. Bucklew, "A note on optimal multidimensional companders," *IEEE Trans. Inform. Theory*, vol. 29, no. 2, p. 279, 1983.

[47] J. Bucklew, "Companding and random quantization in several dimensions," *IEEE Trans. Inform. Theory*, vol. 27, no. 2, pp. 207–211, 1981.

[48] P. Moo and D. Neuhoff, "Optimal compressor functions for multidimensional companding," in *Proc. IEEE International Symposium on Information Theory*, 1997.

[49] S. P. Lloyd, "Least squares quantization in PCM," *IEEE Trans. Inform. Theory*, vol. IT-28, pp. 129–137, 1982.

[50] Y. Linde, A. Buzo, and R. M. Gray, "An algorithm for vector quantizer design," *IEEE Trans. Comm.*, vol. COM-28, pp. 84–95, 1980.

[51] P. Chou, T. Lookabough, and R. Gray, "Entropy-constrained vector quantization," *IEEE Trans. Acoust. Speech Signal Process.*, vol. 38, no. 1, pp. 31–42, 1989.

[52] K. K. Paliwal and B. S. Atal, "Efficient vector quantization of LPC parameters at 24 bits/frame," *IEEE Trans. Speech Audio Process.*, vol. 1, no. 1, pp. 3–14, 1993.

[53] P. Kroon and E. F. Deprettere, "A class of analysis-by-synthesis predictive coders for high quality speech coding at rates between 4.8 and 16 kbit/s.," *IEEE J. Selected Areas Comm.*, vol. 6, no. 2, pp. 353–363, 1988.

[54] P. Kroon and W. B. Kleijn, "Linear-prediction based analysis-by-synthesis coding," in *Speech Coding and Synthesis* (W. B. Kleijn and K. K. Paliwal, eds.), pp. 79–119, Amsterdam: Elsevier, 1995.

[55] E. Agrell, "Spectral coding by fast vector quantization," in *Proc. IEEE Workshop on Speech Coding for Telecommunications*, (Sainte-Adele, Quebec), pp. 61–62, 1993.

[56] V. Ramasubramanian and K. Paliwal, "Fast k-dimensional tree algorithsm for nearest-neighbor search with application to vector quantization encoding," *IEEE Trans. Signal Process.*, pp. 518–531, 1992.

[57] "Recommendation t.81," tech. rep., ITU, 1992.

[58] H. Bölcskei, F. Hlawatsch, and H. Feichtinger, "Frame-theoretic analysis of oversampled filter banks," *IEEE Trans. Signal Process.*, vol. 46, no. 12, pp. 3256–3268, 1998.

[59] G. Strang and T. Nguyen, *Wavelets and filter banks.* Wellesley, MA: Wellesley Cambridge, 1996.

[60] R. Bracewell, *The Fourier Transform and Its Applications.* New York: McGraw Hill, 1986.

[61] F. Beaufays, "Transform-domain adaptive filters: An analytical approach," *IEEE Trans. Signal Process.*, vol. SP-43, no. 2, pp. 422–431, 1995.

[62] N. S. Jayant and P. Noll, *Digital Coding of Waveforms.* Englewood Cliffs NJ: Prentice-Hall, 1984.

[63] H. Malvar, *Signal Processing with Lapped Transforms.* Norwood, MA: Artech House, 1992.

[64] Z. Cvetkovic, "Oversampled modulated filterbanks and tight Gabor frames in $l^2(Z)$," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, (Detroit), pp. 1456–1459, 1995.

[65] M. Vetterli and J. Kovačević, *Wavelets and Subband Coding*. Signal Process., Englewood Cliffs, NJ: Prentice Hall, 1995.

[66] I. Daubechies, *Ten lectures on wavelets*. Philadelphia: SIAM, Academic Press, 1992.

[67] J. Shapiro, "Embedded image coding using zerotree of wavelet coefficients," *IEEE Trans. Signal Process.*, vol. 41, pp. 3445–3462, 1993.

[68] S. Mallat and F. Falzon, "Analysis of low bit rate image transform coding," *IEEE Trans. Signal Process.*, vol. 46, no. 4, pp. 1027–1042, 1998.

[69] S. Mallat, *A Wavelet Tour of Signal Process.* New York: Academic Press, 1998.

[70] K. Ramchandran, M. Vetterli, and C. Herley, "Wavelets, subband coding and best bases," *Proceedings IEEE*, vol. 84, no. 4, pp. 541–560, 1996.

[71] R. Coifman and M. W. Wickerhauser, "Entropy-based algorithms for best basis selection," *IEEE Trans. Inf. Theory*, vol. IT-38, no. 2, pp. 713–718, 1992.

[72] F. Itakura and S. Saito, "Speech information compression based on the maximum likelihood estimation," *J. Acoust. Soc. Jpn.*, vol. 27, no. 9, p. 463, 1971.

[73] S. Saito and K. Nakata, *Fundamentals of Speech Signal Process.* New York: Academic Press, 1985.

[74] G. Kubin, "On the nonlinearity of linear prediction," in *Proc. IX, Europ. Signal Processing Conference EUSIPCO'98*, (Rhodes, Greece), pp. 557–560, Sep 1998.

[75] H.-P. Bernhard, "Determining the predictability of speech," in *Proc. Digital Signal Processing Workshop*, 1996. preprint.

[76] T. Kailath, *Lectures on Wiener and Kalman Filtering*. New York: Springer Verlag, 1981.

[77] D. Florencio, "Investigating the use of asymmetric windows in CELP vocoders," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, (Minneapolis), pp. II427–II430, 1993.

[78] G. H. Golub and C. v. Loan, *Matrix Computations*. The Johns Hopkins University Press, Baltimore, Maryland, 2 ed., 1983.

[79] B. S. Atal, "Predictive coding of speech signals at low bit rates," *IEEE Trans. Comm.*, vol. COM-30, no. 4, pp. 600–614, 1982.

[80] B. S. Atal and M. R. Schroeder, "Predictive coding of speech signals and subjective error criteria," *IEEE Trans. Acoust. Speech Signal Process.*, vol. 27, no. 3, pp. 247–254, 1979.

[81] Y. Tohkura, F. Itakura, and S. Hashimoto, "Spectral smoothing techniques in PARCOR speech analysis-synthesis," *IEEE Trans. Acoust. Speech, Signal Process.*, vol. 26, no. 6, pp. 587–596, 1978.

[82] R. Viswanathan and J. Makhoul, "Quantization properties of transmission parameters in linear predictive systems," *IEEE Trans. Acoust Speech, Signal Process.*, vol. ASSP-23, pp. 309–321, 1975.

[83] C. G. Bell, H. Fujisaki, J. M. Heinz, K. N. Stevens, and A. S. House, "Reduction of speech spectra by analysis-by-synthesis techniques," *J. Acoust. Soc. Am.*, vol. 33, pp. 1725–1736, 1961.

[84] B. S. Atal and M. R. Schroeder, "Stochastic coding of speech at very low bit rates," in *Proc. Int. Conf. Comm.*, (Amsterdam), pp. 1610–1613, 1984.

[85] M. Schroeder and B. Atal, "Code-excited linear prediction (CELP): high quality speech at very low bit rates," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, (Tampa), pp. 937–940, 1985.

[86] B. S. Atal and J. R. Remde, "A new model of LPC excitation for producing natural sounding speech at low bit rates," in *Proc. Int. Conf. Acoust. Speech Signal Process.*, (Paris), pp. 614–617, 1982.

[87] H. Fehn and P. Noll, "Multipath search coding of stationary signals with applications to speech," *IEEE Trans. Comm.*, vol. COM-30, no. 4, 1982.

[88] T. Barnwell and S. Quackenbush, "An analysis of objectively computable measures for speech quality testing," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, pp. 996–999, 1982.

[89] H. Coetzee and T. Barnwell, "An LSP based speech quality measure," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, pp. 596–599, 1989.

[90] T. Barnwell, "A new objective speech quality measure for speech coding systems," *J. Acoust. Soc. Am.*, vol. 87, p. s13(A), 1990.

[91] S. Wang, A. Sekey, and A. Gersho, "An objective measure for predicting subjective quality of speech coders," *IEEE J. Selected Areas Comm.*, vol. 10, no. 5, pp. 819–829, 1992.

[92] F. K. Soong and B. Huang, "Line spectrum pair (LSP) and speech data compression," in *Proc. Int. Conf. Acoust. Speech Signal Process.*, (San Diego), pp. 1.10.1–1.10.4, 1984.

[93] N. Sugamura and F. Itakura, "Speech analysis and synthesis methods developed at ECL in NTT – from LPC to LSP," *Speech Communications*, vol. 5, pp. 199–215, 1986.

[94] B. Atal, R. Cox, and P. Kroon, "Spectral quantization and interpolation for CELP coders," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, (Glasgow), pp. 69–72, 1989.

[95] R. Hagen and P. Hedelin, "Low bit-rate spectral coding in CELP, a new LSP method," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, (Albuquerque), pp. 189–192, 1990.

[96] B. Juang, D. Y. Wong, and A. H. Gray, "Distortion performance of vector quantization for LPC voice coding," *IEEE Trans. Acoust. Speech Signal Process.*, vol. ASSP-30, no. 2, pp. 294–304, 1982.

[97] T. Moriya and M. Honda, "Speech coder using phase equalization and vector quantization," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, (Tokyo), pp. 1701–1704, 1986.

[98] W. F. LeBlanc, B. Bhattacharya, S. A. Mahmoud, and V. Cuperman, "Efficient search and design procedures for robust multi-stage VQ of LPC parameters for 4 kb/s speech coding," *IEEE Trans. Speech and Audio Process.*, vol. 1, no. 4, pp. 373–385, 1993.

[99] R. Laroia, N. Phamdo, and N. Farvardin, "Robust and efficient quantization of speech LSP parameters using structured vector quantizers," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, (Toronto), pp. 641–644, 1991.

[100] F. Soong and B. Juang, "Line spectrum pair (LSP) and speech data compression," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, (San Diego), pp. 1.10.1–1.10.4, 1984.

[101] F. Soong and B. Juang, "Optimal quantization of LSP parameters," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, (New York), pp. 394–397, 1988.

[102] N. Sugamura and N. Farvardin, "Quantizer design in LSP speech analysis and synthesis," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, (New York), pp. 398–401, 1988.

[103] F. Soong and B. Juang, "Optimal quantization of LSP parameters using delayed decisions," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, (Albuquerque), pp. 185–188, 1990.

[104] K. K. Paliwal and W. B. Kleijn, "Quantization of LPC parameters," in *Speech Coding and Synthesis* (W. B. Kleijn and K. K. Paliwal, eds.), pp. 433–466, Amsterdam: Elsevier, 1995.

[105] A. Gray and J. Markel, "Quantization and bit allocation in speech processing," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. ASSP-24, pp. 459–473, 1976.

[106] F. Itakura, "Line spectrum representation of linear predictive coefficients," *J. Acoust. Soc. Am.*, vol. 57 Supplement, no. 1, p. S35, 1975.

[107] B. S. Atal, R. V. Cox, and P. Kroon, "Spectral quantization and interpolation for CELP coders," in *Proc. Int. Conf. Acoust. Speech Signal Process.*, (Glasgow), pp. 69–72, 1989.

[108] J. Erkelens and P. Broersen, "Interpolation of autoregressive process at discontinuities: application to LPC based speech coding," in *Signal Processing VII, Theories and Applications, Proc. of EUSIPCO 94* (M. Holt, C. Cowan, P. Grant, and W. Sandham, eds.), (Edinburgh), pp. 935–938, 1994.

[109] W. R. Gardner and B. D. Rao, "Optimal distortion measures for the high rate vector quantization of LPC parameters," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, (Detroit), pp. 752–755, 1995.

[110] J. Erkelens and P. Broersen, "On the statistical properties of line spectrum pairs," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, (Detroit), pp. 768–771, 1995.

[111] J. L. Kelly and C. Lochbaum, "Speech synthesis," in *Proc. Fourth Int. Congress Acoust.*, (Copenhagen), pp. G42,1–4, 1962.

[112] J. Schur, "Uber Potenzreihen, die im Inneren des Einheitskreises beschränkt sind," *Z. fuer die Reine and Angewandte Mathematik*, vol. 147, pp. 205–232, 1917.

[113] T. Kailath, "A theorem of I. Schur and its impact on modern signal processing," in *Operator Theory:Advances and Applications Vol. 18* (I. Gohberg, ed.), (Basel), pp. 9–30, Birkhäuser Verlag, 1986.

[114] J. LeRoux and C. Gueguen, "A fixed point computation of partial correlation coefficients," *IEEE Trans. Acoust. Speech, Signal Process*, vol. 25, pp. 257–259, 1977.

[115] W. B. Kleijn and K. K. Paliwal, "An introduction to speech coding," in *Speech Coding and Synthesis* (W. B. Kleijn and K. K. Paliwal, eds.), pp. 1–47, Elsevier, 1995.

[116] A. Gersho, "Advances in speech and audio compression," *Proceedings IEEE*, vol. 82, pp. 900–918, 1994.

[117] A. Spanias, "Speech coding: a tutorial review," *Proceedings of the IEEE*, vol. 82, no. 10, pp. 1541–1582, 1994.

[118] B. S. Atal and S. L. Hanauer, "Speech analysis and synthesis by linear prediction of the speech wave," *J. Acoust. Soc. Amer.*, vol. 50, no. 2, pp. 637–655, 1971.

[119] L. R. Rabiner and R. W. Schafer, *Digital Processing of Speech Signals*. Englewood Cliffs, NJ: Prentice-Hall, 1978.

[120] S. Singhal and B. S. Atal, "Improving performance of multi-pulse LPC coders at low bit rates," in *Proc. Int. Conf. Acoust. Speech Signal Process.*, (San Diego), pp. 1.3.1–1.3.4, 1984.

[121] W. B. Kleijn, D. J. Krasinski, and R. H. Ketchum, "Improved speech quality and efficient vector quantization in SELP," in *Proc. Int. Conf. Acoust. Speech Signal Process.*, (New York), pp. 155–158, 1988.

[122] S. V. Andersen and W. B. Kleijn, "Reverse water-filling in predictive encoding of speech," in *IEEE Speech Coding Workshop*, (Porvoo), pp. 105–107, 1999.

[123] V. Ramamoorthy and N. S. Jayant, "Enhancement of ADPCM speech by adaptive postfiltering," *AT&T Bell Labs. Tech. J.*, pp. 1465–1475, 1984.

[124] J. Chen and A. Gersho, "Adaptive postfiltering for quality enhancement of coded speech," *IEEE Trans. Speech Audio Process.*, vol. 3, no. 1, pp. 59–71, 1995.

[125] K. Mano and T. Moriya, "4.8 kbit/s delayed decision CELP code using tree coding," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, (Albuquerque), pp. 21–24, 1990.

[126] H. Su and P. Mermelstein, "Delayed decision coding of pitch and innovation signals in code-excited linear prediction coding of speech," in *Speech and audio coding for wireless and network applications* (B. S. Atal, V. Cuperman, and A. Gersho, eds.), (Boston), pp. 69–76, Kluwer Academic Publishers, 1993.

[127] M. Birgmeier, H. Bernhard, and G. Kubin, "Nonlinear long-term prediction of speech signals," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, (Munich), pp. 1283–1286, 1997.

[128] R. E. Crochiere, S. A. Webber, and J. L. Flanagan, "Digital coding of speech in subbands," *Bell Syst. Techn. J.*, vol. 55, no. 8, pp. 1069–1085, 1976.

[129] A. Gersho, T. Ramstad, and I. Versvik, "Fully vector-quantized subband coding with adaptive codebook allocation," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, pp. 10.7.1–10.7.4, 1984.

[130] I. Versvik and H. C. Guren, "Subband coding with vector quantization," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, (Tokyo), pp. 3099–3102, 1986.

[131] H. Abut and S. Ergezinger, "Low-rate speech encoding using vector quantization and subband coding," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, (Tokyo), pp. 449–452, 1986.

[132] B. Tang, A. Shen, A. Alwan, and G. Pottie, "A perceptually-based embedded subband speech coder," *IEEE Trans. Speech and Audio Process.*, vol. 5, no. 2, pp. 131–140, 1997.

[133] R. J. McAulay and T. F. Quatieri, "Speech analysis/synthesis based on a sinusoidal representation," *IEEE Trans. Acoust. Speech Signal Process.*, vol. 34, pp. 744–754, 1986.

[134] R. J. McAulay and T. Quatieri, "Sinusoidal coding," in *Speech Coding and Synthesis* (W. B. Kleijn and K. K. Paliwal, eds.), pp. 121–173, Amsterdam: Elsevier, 1995.

[135] W. B. Kleijn, "Encoding speech using prototype waveforms," *IEEE Trans. Speech and Audio Process.*, vol. 1, no. 4, pp. 386–399, 1993.

[136] W. B. Kleijn and J. Haagen, "Waveform interpolation for speech coding and synthesis," in *Speech Coding and Synthesis* (W. B. Kleijn and K. K. Paliwal, eds.), pp. 175–208, Amsterdam: Elsevier, 1995.

[137] W. B. Kleijn, "A frame interpretation of sinusoidal coding and waveform interpolation," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process.*, vol. 3, (Istanbul), pp. 1475–1478, 2000.

[138] W. B. Kleijn, H. Yang, and E. F. Deprettere, "Waveform interpolation with pitch-spaced subbands," in *Proc. Int. Conf. Spoken Lang. Process.*, (Sydney), pp. 1795–1798, 1998.