Forskarskola i medicinsk bioinformatik
Course for CMI PhD programme in Medical Bioinformatics:
Support Vector Machines Part I
Lecture Notes 1: Linear Learning Machines
8-9th of September, 2005
LiTH/Linköping University

Timo Koski
Department of mathematics
LiTH
Overview of the lectures // Part I

Support vector machines (SVM), as invented by V.N. Vapnik and his coworkers in the early 1990’s, are based on a combination of two ideas from 1950-60’s: linear learning machines and kernel mappings. These were put into perspective with statistical learning theory giving the capacity of a learning machine, as defined by Vapnik. In part I of the course we are going to discuss the above techniques in SVM for pattern recognition/classification. There are techniques of SVM regression and SVM time series prediction, too, and other applications not covered in these lectures. However, SVM regression is studied in the computer exercises.
Overview of the lectures //Part II

In part II of the lectures we will deal with

- kernel classifiers from a Bayesian perspective.
- designing kernels (incl. string kernels and Fisher kernels)
- kernel PCA (= principal component analysis)
- kernel Fisher discriminant

This first set of lecture notes deals with the main building blocks of the most simple SVMs, and exhibits the concepts required for understanding more complex SVMs.

We start with the theory of linear learning machines. There are some prerequisites, known as 'the geometry of learning'.
TEXTS ON SUPPORT VECTOR MACHINES

The lecture is mainly based on


OUTLINE OF TOPICS:

• vector spaces

• normed vector spaces, inner product spaces

• properties of inner products

• geometry of linear learning machines

• perceptron

• perceptron convergence

• dual representation of perceptron convergence
Vector Spaces, Norms and Inner Products

We need a mathematical groundwork for a geometry of linear learning. We recall a few facts about vector spaces, norms, and inner products, as found in a plethora of books on, e.g., linear algebra or functional analysis. For a treatment that also includes some of the optimization theory needed for SVM we refer to

Prerequisites (A) on Vector Spaces

A set $X$ with elements $x, y, z \ldots$, referred to as vectors, is called a vector space, if there are two operations called 'addition of two vectors' and 'multiplication of a vector by scalar', $\alpha \in R$. These operations satisfy

- $x + y \in X$

- There is a neutral element $0$ such that $x + 0 = x$

- $\alpha x \in X$

- $1x = x \in X, \ 0x = 0 \in X$
Prerequisites (A) on Vector Spaces

and in addition

- \( x + y = y + x \)

- \( \alpha (x + y) = \alpha x + \alpha y \)

- \( (\alpha + \beta) x = \alpha x + \beta x \)

We take

\[ x - y = x + (-1)y \]

and then

\[ x - x = 0 \]
Prerequisites (A) on Vector Spaces: An Example

The standard example is the set $\mathbb{R}^n$ of real column vectors of fixed dimension $n$. Let $T$ denote transpose

$$x = (x_1, x_2, \ldots, x_n)^T$$

(a transposed row vector is a column vector), where $x_i \in \mathbb{R}, \quad i = 1, \ldots, n.$

$$0 = (0, 0, \ldots, 0)^T$$

We define

$$x + y = (x_1 + y_1, x_2 + y_2, \ldots, x_n + y_n)^T$$

$$\alpha x = (\alpha x_1, \alpha x_2, \ldots, \alpha x_n)^T$$
Prerequisites (A): Vector Spaces and Norms

A vector space $X$ is called a normed linear space, if there is a real-valued function that maps each $x \in X$ to a number $\|x\|$ with the following properties

- $\|x\| \geq 0$ for all $x \in X$
- $\|x\| = 0$ if and only if $x = 0$.
- $\|x + y\| \leq \|x\| + \|y\|$ (triangle inequality)
- $\|\alpha x\| = |\alpha|\|x\|$ (homogeneity)
Linear Vector Spaces and Norms:
Examples

Consider $R^n$. Then $\|x\|_2$ and $\|x\|_\infty$ are norms on $R^n$:

- $\|x\|_2 = \sqrt{\sum_{i=1}^{n} x_i^2}$ (Euclidean norm)
- $\|x\|_\infty = \max_{1 \leq i \leq n} |x_i|$
Prerequisites (A): Norms and Distances

In a normed linear space $\mathbf{X}$ the real-valued function

$$d(x, y) = \|x - y\|$$

is called the distance between $x$ and $y$.

- $d(x, y) \geq 0$ for all $x \in \mathbf{X}$
- $d(x, y) = d(y, x)$.
- $d(x, y) = 0$ if and only if $x = y$.
- $d(x, y) \leq d(x, z) + d(y, z)$ (triangle inequality)
- $d(\alpha x, \alpha y) = |\alpha|d(x, y)$ (homogeneity)
Lengths and Balls in Normed Spaces

Then the *length* of $x$ is the distance from $x$ to 0, i.e.,

$$d(x, 0) = ||x - 0|| = ||x||$$

The **open ball** $B_\tau(x) \subset X$ of radius $\tau$ around $x \in X$ is

$$B_\tau(x) \overset{\text{def}}{=} \{y \in X ||y - x|| < \tau\}$$
Prerequisites (A): Inner Product Spaces

A vector space $X$ is called an *inner product space*, if there is a function, called inner product*, that maps each pair $x, y$ of vectors in $X$ to a number $\langle x, y \rangle$ with the following properties

- $\langle x, y \rangle = \langle y, x \rangle$

- $\langle x, x \rangle \geq 0$ with $\langle x, x \rangle = 0$, if and only if $x = 0$.

- $\langle x + y, z \rangle = \langle x, z \rangle + \langle y, z \rangle$

- $\langle x, y + z \rangle = \langle x, y \rangle + \langle x, z \rangle$

- $\langle x, \alpha y \rangle = \alpha \langle x, y \rangle$

*Much of learning theory literature talks about dot products*
Prerequisites (A): Inner Product Spaces

Inner Product Spaces: Examples

1. Take $X = \mathbb{R}^n$, and

$$\langle x, y \rangle = \sum_{i=1}^{n} x_i y_i$$

2. Take $X = \mathbb{R}^n$, and

$$\langle x, y \rangle_A = \sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij} x_i y_j = \langle x, Ay \rangle$$

where $A = (a_{ij})_{i,j=1}^{n,n}$ is a symmetric and non-negative definite matrix*

*$$\langle x, Ay \rangle = \sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij} x_i x_j \geq 0$$ for all $x$. 

16
Inner Product Spaces are Normed Spaces

An inner product space $X$ is automatically a normed linear space, since we can put

$$||x|| = \sqrt{\langle x, x \rangle}$$

and, since $\langle x, x \rangle \geq 0$,

$$||x||^2 = \langle x, x \rangle$$

Example: $X = \mathbb{R}^n$

$$||x|| = \sqrt{\langle x, x \rangle} = \sqrt{\sum_{i=1}^{n} x_i^2}$$
Examples: Mahalanobis distance

When we take $A = \Sigma^{-1}$ in $\langle x, y \rangle_A$ and set

$$\|x - y\|_{\Sigma^{-1}}^2 = \langle x - y, \Sigma^{-1} (x - y) \rangle$$

we obtain a useful distance in pattern recognition known as the (squared) Mahalanobis distance. Regions of constant Mahalanobis distance to a fixed vector $y$ are ellipsoids.
Properties of Inner Products

• Cauchy-Schwarz inequality

\[ \langle x, y \rangle \leq \|x\| \cdot \|y\| \]

• \( \|x + y\|^2 = \|x\|^2 + \|y\|^2 + 2 \langle x, y \rangle \)

• \( \|x - y\|^2 = \|x\|^2 + \|y\|^2 - 2 \langle x, y \rangle \)
Properties of Inner Product Spaces

The angle $\theta$ between $x$ and $y$ in an inner product space is given by

\[ \cos(\theta) = \frac{\langle x, y \rangle}{\|x\| \|y\|} \]

If $\langle x, y \rangle = 0$, we say that $x$ and $y$ are orthogonal, since then $\cos(\theta) = 0$, and then $\theta = \pi/2$ (within period). Also, we have then the Pythagorean relations

\[ \|x + y\|^2 = \|x\|^2 + \|y\|^2 \]
\[ \|x - y\|^2 = \|x\|^2 + \|y\|^2 \]
Orthonormal Vectors in Inner Product Spaces

Let $\phi_1, \ldots, \phi_n$ be a sequence of orthonormal vectors of an inner product space $X$. Orthonormality means that

1. $\langle \phi_i, \phi_j \rangle = 0$, for $i \neq j$.

2. $\|\phi_i\| = 1$ for all $j$.

We note an example.

1. $X = \mathbb{R}^n$, and $\langle x, y \rangle = \sum_{i=1}^{n} x_i y_i$.

Then

$$
\phi_i = \begin{pmatrix}
0, 0, \ldots, 1, 0, \ldots, 0
\end{pmatrix}, \quad i = 1, \ldots, n
$$

are orthonormal vectors in $X = \mathbb{R}^n$. 

21
**Basis and Dimension**

We say that the orthonormal vectors $\phi_1, \ldots, \phi_n$ form an orthonormal *basis* for $X$, if every $x$ can be written

$$x = \langle x, \phi_1 \rangle \phi_1 + \ldots + \langle x, \phi_n \rangle \phi_n$$

and that this is unique, also in the sense $\phi_1, \ldots, \phi_n$ cannot be appended or reduced by a single orthonormal vector. Then $n$ is the *dimension* of the inner product space $X$.

We shall later on need to discuss inner product spaces $X$ of infinite dimension, i.e., where formally speaking

$$x = \sum_{i=1}^{\infty} \langle x, \phi_i \rangle \phi_i$$

so that the basis has infinitely many orthonormal vectors.
Inner Product Spaces

An inner product space supports thus geometric notions. Inner product spaces of finite dimension are often called Euclidean spaces. A Euclidean space $X$ equipped with $\langle x, y \rangle_X$ is isomorphic to $\mathbb{R}^n$ with $\langle x, y \rangle = \sum_{i=1}^{n} x_i y_i$ in the sense that there is an invertible map $\psi : X \mapsto \mathbb{R}^n$ such that

$$\langle x, y \rangle_X = \langle \psi(x), \psi(y) \rangle$$
LINEAR LEARNING MACHINES: NOTATIONS

Next we start studying linear learning machines, also known as perceptrons.

\[ X = \text{INPUT SPACE} \]

which is also an inner product space (of finite dimension).

\[ Y = \text{OUTPUT DOMAIN} \]

Examples of output domains are \{+1, −1\} (output domain for binary classification), or \{1, 2, ..., m\} (multiple classification).

We consider functions \( X \overset{f}{\to} R \) such that

\[ f(x) = \langle w, x \rangle + b \]

where \( w \) is a vector called weight vector, and \( b \) is a real number called bias.
Output domains are \( \{+1, -1\} \)

Most of the theory to be presented deals with the binary output domain represented as \( \{+1, -1\} \). This is, of course, restrictive, but this is not completely without practical relevance, e.g., in bioinformatics.

To quote a problem: the prediction of constitutive and immunoproteasome cleavage sites in antigenic sequences *

See also the supporting material on the webpage of N. Christiani & J. Shawe-Taylor (2000):

http://www.support-vector.net/bioinformatics.html

LINEAR LEARNING MACHINES: Some intuition

\[ |\langle \mathbf{w}, \mathbf{x} \rangle - \langle \mathbf{w}, \mathbf{y} \rangle| \]
\[ = |\langle \mathbf{w}, \mathbf{x} - \mathbf{y} \rangle| \]
by rules of inner product, and then
\[ \leq ||\mathbf{w}|| ||\mathbf{x} - \mathbf{y}|| \]
by Cauchy-Schwartz.

Hence whenever the difference \( ||\mathbf{x} - \mathbf{y}|| \) is small, the difference in the real-valued outputs is also small.
LINEAR LEARNING MACHINES:
Some geometry

A hyperplane is an affine* subspace of dimension \( n - 1 \), if \( n \) is the dimension of \( X \), which divides the space into two half spaces. We consider the following hyperplane \( D \subset X \):

\[
D = \{ x \in X \mid f(x) = 0 \}
\]

where

\[
f(x) = \langle w, x \rangle + b
\]

Take \( x_1 \in D \) and \( x_2 \in D \). Then

\[
\langle w, x_1 \rangle = \langle w, x_2 \rangle \iff \langle w, x_1 - x_2 \rangle = 0.
\]

In other words, \( w \) is orthogonal to any vector in \( D \).

*\( = \) parallel to a linear subspace, a linear subspace contains 0
LINEAR LEARNING MACHINES: Some geometry

\[ f(x) = \langle w, x \rangle + b \]

\[ D = \{ x \in X \mid f(x) = 0 \} \]

A hyperplane is an affine subspace of dimension \( n - 1 \), which divides the space into two half spaces, here \( R_1 \) and \( R_2 \), as

\[ R_1 = \{ x \in X \mid f(x) > 0 \} \]

\[ R_2 = \{ x \in X \mid f(x) < 0 \} \]

By the preceding \( w \) is orthogonal to any vector in \( D \) and points to \( R_1 \).
LINEAR LEARNING MACHINES:
Some geometry

\[ f(x) = 0 \]
\[ f(x) > 0 \]
\[ f(x) < 0 \]
LINEAR LEARNING MACHINES: More geometry

Let $x_p = P_D(x_p)$ be the orthogonal projection of $x$ onto $D$, i.e.,

$$\langle x_p, x - x_p \rangle = 0$$
LINEAR LEARNING MACHINES: 
More geometry

Write*

\[ x = x_p + r_x \cdot \frac{w}{\|w\|} \]

where \( r_x \) is a number (to be determined) and \( \frac{w}{\|w\|} \) is a unit length vector in the direction of \( w \). We compute the squared length

\[ \|x - x_p\|^2 = \left\| r_x \cdot \frac{w}{\|w\|} \right\|^2 = \left( \frac{r_x}{\|w\|} \right)^2 \|w\|^2 \]

\[ \iff \]

\[ \|x - x_p\|^2 = r_x^2 \]

\[ \iff \]

\[ \|x - x_p\| = |r_x| \]

Hence \( r_x \) is the signed distance of \( x \) to \( D \).

LINEAR LEARNING MACHINES: More geometry

In addition, with

\[ f(x) = \langle w, x \rangle + b = \]

\[ = \langle w, x_p + r_x \cdot \frac{w}{\|w\|} \rangle + b = \langle w, x_p \rangle + \langle w, r_x \cdot \frac{w}{\|w\|} \rangle + b \]

\[ = \underbrace{\langle w, x_p \rangle + b + \langle w, r_x \cdot \frac{w}{\|w\|} \rangle}_{=f(x_p)} \]

\[ = f(x_p) + \langle w, r_x \cdot \frac{w}{\|w\|} \rangle \]

\[ = \langle w, r_x \cdot \frac{w}{\|w\|} \rangle \]
LINEAR LEARNING MACHINES:
Signed distance

\[ \langle \mathbf{w}, r_x \cdot \frac{\mathbf{w}}{\|\mathbf{w}\|} \rangle = r_x \cdot \frac{1}{\|\mathbf{w}\|} \langle \mathbf{w}, \mathbf{w} \rangle = r_x \|\mathbf{w}\|. \]

In other words we have obtained

\[ f(x) = r_x \|\mathbf{w}\| \]

\[ \iff \]

\[ r_x = \frac{f(x)}{\|\mathbf{w}\|} \]

Furthermore, \( f(0) = \langle \mathbf{w}, 0 \rangle + b = 0 + b = 0 \), i.e.,

\[ r_0 = \frac{f(0)}{\|\mathbf{w}\|} = \frac{b}{\|\mathbf{w}\|} \]
LINEAR LEARNING MACHINES:
Some geometry

Recalling $f(x) = r_x \|w\|$, if $r_x > 0$, then $x$ is on the positive side, and if $r_x < 0$, then $x$ is on the negative side. Here we take

$$\mathcal{R}_1 = \{x \in X \mid f(x) > 0\}$$

as the positive side of $D$

$$\mathcal{R}_2 = \{x \in X \mid f(x) < 0\}$$

as the negative side of $D$. 
LINEAR LEARNING MACHINES:
More geometry

Since $r_0$ is the signed distance of 0 to $D$, and

$$r_0 = \frac{b}{\|w\|}$$

we see that varying $b$ moves $D$ parallell to itself. If $b > 0$, then 0 is in $\mathcal{R}_1$, and if $b < 0$, then 0 is in $\mathcal{R}_2$. 
LINEAR LEARNING MACHINES:
More geometry

\[ r_0 = \frac{b}{\|w\|} \] is the signed distance of 0 to \( D \)
LINEAR LEARNING MACHINE a.k.a. PERCEPTRON

Let

$$\text{sign}(x) = \begin{cases} +1 & 0 \leq x, \\ -1 & x < 0 \end{cases}$$

(Note that the definition of sign for $x = 0$ is arbitrary.) Then a linear learning machine for a binary classification task (or for a binary target concept), a.k.a. perceptron, processes an input $x$ by giving the output

$$y = \text{sign}(f(x)), \quad y \in \{+1, -1\}$$

This requires, of course, that we have, somehow, set the values of $w$ and $b$. The learning of the weights $w$ and the bias $b$ from examples is our next topic of study.
LINEAR LEARNING MACHINE a.k.a. PERCEPTRON

Frank Rosenblatt* developed in 1957 the perceptron, i.e.,

\[ y = \text{sign}(f(x)), \quad y \in \{+1, -1\} \]

as a model to understand human memory, learning, and cognition. The perceptron was based on biological ideas about networks of neurons in the brain. In 1960 Rosenblatt demonstrated a piece of electromechanical hardware, named Mark I Perceptron, the first machine that could learn to recognize and identify optical patterns.


LINEAR LEARNING MACHINE a.k.a. PERCEPTRON

The perceptron theory, as it stood in the 1960's, is very lucidly presented in


A thorough presentation and at the same time a devastating criticism of the performance of perceptrons, and then a reverse reappraisal, is in


When sign \((f(x))\) was replaced by smooth functions, e.g. \(\tanh(f(x))\), and several such perceptrons were layered after each other, and backpropagation algorithm was invented, the theory of neural networks or feedforward multilayer perceptrons, was launched.
PERCEPTRON & NEURAL NETWORKS

A very clearly written survey of learning theory (with an electrical engineering point of view) is found in

PERCEPTRON TRAINING

Let $x_i \in X$, $y_i \in Y = \{+1, -1\}$, $i = 1, \ldots, l$ be $l$ pairs of examples of a binary target concept. Then

$$S = \{(x_1, y_1), \ldots, (x_l, y_l)\}$$

is called a training set. Then we consider the following learning task

**Supervised learning of perceptrons**

Find $(w, b)$ using $S$. 
PERCEPTRON TRAINING

\[ S = \{(x_1, y_1), \ldots, (x_l, y_l)\} \]

**Supervised learning of perceptrons**

Task: Find \((w, b)\) using \(S\) so that \(f(x_i) = y_i\) for all \(i\), i.e., all inputs are correctly classified.

Questions:

1. Does a solution to this task exist?

2. If a solution exists, is there an algorithm that finds it in a finite number of steps?
Linear Separability

We assume that there exists a hyperplane that correctly classifies $S$. More formally this can be stated as

**Linear Separability of $S$**

There exist $(w_*, b_*)$ such that if

$$f_*(x) = \langle w_*, x \rangle + b_*$$

then for all $(x_i, y_i) \in S$ we have $f_*(x) > 0$ if and only if $y_i = +1$.

When this assumption is true, we call $S$ *linearly separable*. 
Linear Separability

Linear separability of $S$ is easy to illustrate with hyperplane geometry: The examples $(x_i, y_i)$ are represented by points with two labels $\circ$ and $\dagger$. All points with labels $\dagger$ lie in $\mathcal{R}_1$ and all points with labels $\circ$ lie in $\mathcal{R}_2$ with regard to the hyperplane $f_*(x) = 0$. 
Linear Separability

There are theorems giving sufficient conditions for separability. E.g., two disjoint convex sets in $\mathbb{R}^k$ are linearly separable. Examples of convex sets are halfplanes, and open balls.

This and other relevant results are found, e.g., in pp. 46–53 of

CONVEXITY

CONVEX

NONCONVEX
A LINEARLY NONSEPARABLE SITUATION

The examples correspond to points with the labels $x$ and $\circ$. The training set cannot be separated by a hyperplane without a considerable number of errors.
Rosenblatt’s perceptron algorithm

An algorithm that finds a perceptron that classifies correctly a linearly separable training set can be constructed as follows. One assumes a weight vector and a bias, and then sweeps through the training set with the corresponding perceptron. For every incorrectly classified example the hyperplane direction is changed and the hyperplane is shifted parallell to itself.

It turns out that this algorithm halts after a finite number of runs, i.e., it finds a correctly classifying perceptron in a finite time, and that there is an explicit upper bound on the number of sweeps through $S$. 
Rosenblatt’s perceptron algorithm

Every time an example is classified incorrectly, the weight vector is updated by adding a term that is proportional to the error margin (with sign). We note that

- If

\[ y_i \left( \langle w_k, x_i \rangle + b_k \right) > 0 \]

then \( x_i \) is correctly classified.

Hence it would seem to be a good idea to update, i.e. rotate the hyperplane, by adding to the weight vector a vector proportional to \( y_i \left( \langle w_k, x_i \rangle + b_k \right) \), as soon as this is negative.
Rosenblatt’s perceptron algorithm

The bias is updated (the hyperplane is shifted parallel to itself) by adding a term that is proportional to the square of the maximum length of the input vectors in the training set. The algorithm is run until no examples in the training set are longer misclassified.

\[ R = \max_{1 \leq i \leq l} \|x_i\| \]

Clearly \( R \) gives radius for the ball \( B_R (0) \), where all training inputs lie. In addition, we must have \( b < R \), since otherwise all of \( S \) will lie on one side of the hyperplane, which is meaningful only for trivial training sets, i.e. those that contain only one label.
Algorithm 1 Rosenblatt’s perceptron algorithm. Given linearly separable $S$ and a learning rate $\eta > 0$

1: $w_0 \leftarrow 0$, $b_0 \leftarrow 0$, $k \leftarrow 0$ $R \leftarrow \max_{1 \leq i \leq l} \|x_i\|$

2: repeat

3: for $i = 1$ to $l$ do

4: if $y_i (\langle w_k, x_i \rangle + b_k) \leq 0$ then

5: $w_{k+1} \leftarrow w_k + \eta y_i x_i$,

$b_{k+1} \leftarrow b_k + \eta y_i R^2$,

$k \leftarrow k + 1$

6: end if

7: end for

8: until no mistakes made in the loop

9: return $w_k, b_k$, where $k$ is the number of mistakes.
Functional margin

The properties of the perceptron algorithm as well as of SVMs will be clearly understood, once we introduce the concept of a *functional margin*. Here we set for some \((w, b)\)

\[
\gamma_i := y_i (\langle w, x_i \rangle + b_k) = y_i f(x_i)
\]

We call \(\gamma_i\) the functional margin of \((x_i, y_i)\) with regard to \((w, b)\). We set

\[
\gamma := \min (\gamma_1, \ldots, \gamma_l)
\]

and call \(\gamma\) the *functional margin* of \(S\) with regard to \((w, b)\).
Functional margin & the geometric margin

Recall from the above, $r_x = \frac{f(x)}{\|w\|}$, then

$$\gamma_i = y_i f(x_i) = y_i \cdot r_x \|w\|$$

We obtain in view of the preceding geometric analysis, that the distance from $x_i$ to $D$ is

$$\|x_i - P_D(x_i)\| = |r_x|$$

If the input $x_i$ is correctly classified, then

$$|r_x| = y_i \cdot r_x$$

Hence, in case $\|w\| = 1$, and all $x_i$ are correctly classified,

$$\gamma = \min (\gamma_1, \ldots, \gamma_l) = \min y_i \cdot r_x$$

Hence, here $\gamma$ is the geometric margin.

The geometric margin of $S$ is the maximum of $\gamma$ over all hyperplanes.
Functional margin & the geometric margin

\[ f(x) = 0 \]
Perceptron Convergence Theorem  
(Novikoff 1961)

Let $S$ be a non-trivial linearly separable training set and let for all $i$

$$\gamma_i = y_i (\langle w_*, x_i \rangle + b_*) \geq \gamma$$

where $\|w_*\| = 1$.

Then number of repeats made by Rosenblatt’s perceptron algorithm is at most

$$\left( \frac{R^2}{\gamma} \right)$$
Proof: Perceptron Convergence (1)

We extend the inner product space to $X \times R$; for $\hat{x} = (x, x)$ and $\hat{y} = (y, y)$. We take $\hat{x} + \hat{y} = (x + y, x + y)$, $\alpha \hat{x} = (\alpha x, \alpha x)$ and we set

$$\langle \hat{x}, \hat{y} \rangle_{ext} = \langle x, y \rangle + xy$$

This is an inner product on $X \times R$.

We set

$$\hat{x}_i = (x_i, R)$$

$$\hat{w} = \left( w, \frac{b}{R} \right)$$

By our extended inner product above we get

$$y_i \langle \hat{w}, \hat{x}_i \rangle_{ext} = y_i (\langle w, x_i \rangle + b)$$

Let now $\hat{w}_{t-1}$ be the weight vector prior to the $t$th mistake. The $t$th mistake/update happens when $(x_i, y_i)$ is classified incorrectly by $\hat{w}_{t-1}$, i.e.,

$$y_i \langle \hat{w}_{t-1}, \hat{x}_i \rangle_{ext} \leq 0$$
Proof: Perceptron Convergence (2)

In view of the pseudocode above the updates can be written as

$$\left( w_t, \frac{b_t}{R} \right) \leftarrow \left( w_{t-1} + \eta y_i x_i, \frac{b_{t-1}}{R} + \eta y_i R \right)$$

$$\iff$$

$$\hat{w}_t = \hat{w}_{t-1} + \eta y_i \hat{x}_i$$

Let

$$\hat{w}_* = \left( w_*, \frac{b_*}{R} \right)$$

correspond to a separating hyperplane with $w_*$ and $b_*$. Then we get

$$\langle \hat{w}_t, \hat{w}_* \rangle_{ext} = \langle \hat{w}_{t-1}, w_* \rangle_{ext} + \eta y_i \langle \hat{x}_i, \hat{w}_* \rangle_{ext}$$

and

$$\langle \hat{x}_i, \hat{w}_* \rangle_{ext} = \langle x_i, w_* \rangle + \frac{b_*}{R}$$

$$= \langle x_i, w_* \rangle + b_*$$
Proof: Perceptron Convergence (3)

Hence linear separability of $S$ w.r.t. $w_*, b_*$ entails

$$\eta y_i \ll \hat{x}_i, \hat{w}_* = \eta y_i (\ll x_i, w_* + b_*) \geq \eta \gamma.$$  

In other words we have obtained that

$$\ll \hat{w}_t, \hat{w}_* \gg_{ext} \ll \hat{w}_{t-1}, \hat{w}_* \gg + \eta \gamma$$

By successive application of this inequality we get

$$\ll \hat{w}_t, \hat{w}_* \gg_{ext} \geq t \eta \gamma$$
Proof: Perceptron Convergence (4)

We use

\[ \hat{w}_t = \hat{w}_{t-1} + \eta y_i \hat{x}_i \]

and the rules for norms and inner products and get

\[ \|\hat{w}_t\|_{ext}^2 = \|\hat{w}_{t-1}\|_{ext}^2 + \eta^2 \|\hat{x}_{t-1}\|_{ext}^2 + 2\eta y_i \ll \hat{w}_{t-1}, \hat{w}_* \gg_{ext} \]

by incorrect classification of \((x_i, y_i)\), and

\[ \leq \|\hat{w}_{t-1}\|_{ext}^2 + \eta^2 \|\hat{x}_{t-1}\|_{ext}^2 \]

by incorrect classification of \((x_i, y_i)\), and

\[ \leq \|\hat{w}_{t-1}\|_{ext}^2 + \eta^2 (\|x_{t-1}\|^2 + R^2) \]

\[ \leq \|\hat{w}_{t-1}\|_{ext}^2 + \eta^2 (R^2 + R^2) \]

\[ \leq \|\hat{w}_{t-1}\|_{ext}^2 + 2\eta^2 R^2 \]

I.e., by successive application

\[ \|\hat{w}_t\|_{ext}^2 \leq 2t\eta^2 R^2 \]
Proof: Perceptron Convergence (5)

By the results
\[ \langle \hat{w}_t, \hat{w}_* \rangle_{ext} \geq t \eta \gamma \]
and
\[ \| \hat{w}_t \|_{ext}^2 \leq 2 t \eta^2 R^2 \]
we get by Cauchy-Schwarz’ inequality
\[ t \eta \gamma \leq \langle \hat{w}_t, \hat{w}_* \rangle_{ext} \leq \| \hat{w}_t \|_{ext} \cdot \| \hat{w}_* \|_{ext} \leq \sqrt{2 t \eta R} \| \hat{w}_* \|_{ext} \]

We need to bound $\| \hat{w}_* \|_{ext}$. 

Proof: Perceptron Convergence (5)

We need to bound $\|\hat{w}_*\|_{ext}$ and therefore we check

$$\|\hat{w}_*\|_{ext}^2 = \|w_*\|^2 + \frac{b_*^2}{R^2}$$

and since $S$ is non-trivial,

$$\leq \left\| w_* \right\|^2 + \frac{R^2}{R^2} = 2$$

In all we have obtained

$$t\eta \gamma \leq \sqrt{2t\eta R^2}\|\hat{w}_*\|_{ext} \leq \sqrt{t\eta}2R$$

i.e.,

$$\sqrt{t} \leq \frac{2R}{\gamma}$$

and

$$t \leq \left( \frac{2R}{\gamma} \right)^2$$

which verifies the assertion as claimed.
Comments on Perceptron Convergence

The number of errors $t$ was found to bounded by

$$t \leq \left( \frac{2R}{\gamma} \right)^2$$

- The bound does not depend on the learning rate (counterintuitive)
- A large geometric margin promises a fast convergence
- A small radius on inputs promises a fast convergence
The Scaling Freedom

If $\alpha \neq 0$, then

$$D = \{ x \in X \mid = 0 \}$$

$$= \{ x \in X \mid \ll w, x \gg + b = 0 \}$$

$$= \{ x \in X \mid \ll \alpha w, x \gg + \alpha b = 0 \}$$

Hence $\alpha w, \alpha b$ corresponds to the same hyperplane as $w, b$. This was used in the proof above by taking $\alpha = \frac{1}{\|w_*\|}$, since then

$$\|\alpha w\| = \left\| \frac{1}{\|w_*\|} w \right\| = 1.$$ 

Elimination of scaling freedom is discussed in the lecture2.
Dual Form of the Perceptron Algorithm

The perceptron algorithm works by adding misclassified examples with label $+1$, and by subtracting misclassified examples with label $-1$. Hence, as $w_0 = 0$, we have after the algorithm has stopped that

$$w = \sum_{i=1}^{l} a_i y_i x_i$$

where $a_i$ is proportional to the number of times misclassification of $x_i$ has caused the weight to be updated. (The algorithm sweeps through $S$ in some fixed order a finite number of times until the algorithm converges.) $a_i$ is also called the embedding strength of $x_i$.  
Dual Form of the Perceptron Algorithm

Hence we can write

\[ \langle w, x \rangle + b = \langle \sum_{i=1}^{l} a_i y_i x_i, x \rangle + b \]

\[ = \sum_{i=1}^{l} a_i y_i \langle x_i, x \rangle + b \]

and hence the perceptron output is

\[ h(x) = \text{sign} \left( \sum_{i=1}^{l} a_i y_i \langle x_i, x \rangle + b \right) \]

This gives a dual form of the algorithm. (The concept of duality will reappear and plays a big part later.)
Algorithm 2 Dual form of the perceptron algorithm. Given linearly separable $S$

1: $\mathbf{a} = (a_1, a_2, \ldots, a_l) \leftarrow 0$, $b_0 \leftarrow 0$, $k \leftarrow 0$

$R \leftarrow \max_{1 \leq i \leq l} \|x_i\|$ 

2: repeat

3: for $i = 1$ to $l$ do

4: if $y_i \left( \sum_{j=1}^{l} a_j y_j \ll x_j, x_i \gg + b \right) \leq 0$ then

5: $a_i \leftarrow a_i + 1,$

$b \leftarrow b + y_j R^2$

6: end if

7: end for

8: until no mistakes made within the for loop

9: return $\mathbf{a}, b$, define $h(x)$. 
Dual Form of the Perceptron Algorithm

Then we have, by the perceptron convergence theorem,

\[ \sum_{j=1}^{l} a_j \leq \left( \frac{2R}{\gamma} \right)^2 \]

The sum \( \sum_{j=1}^{l} a_j \) measures thus the complexity of the target concept in the dual representation.

NOTE: In the dual form of the perceptron algorithm the inputs appear only through their pairwise inner products \( \ll x_j, x_i \gg \).