A summary of
"Deep Learning without Poor Local Minima"

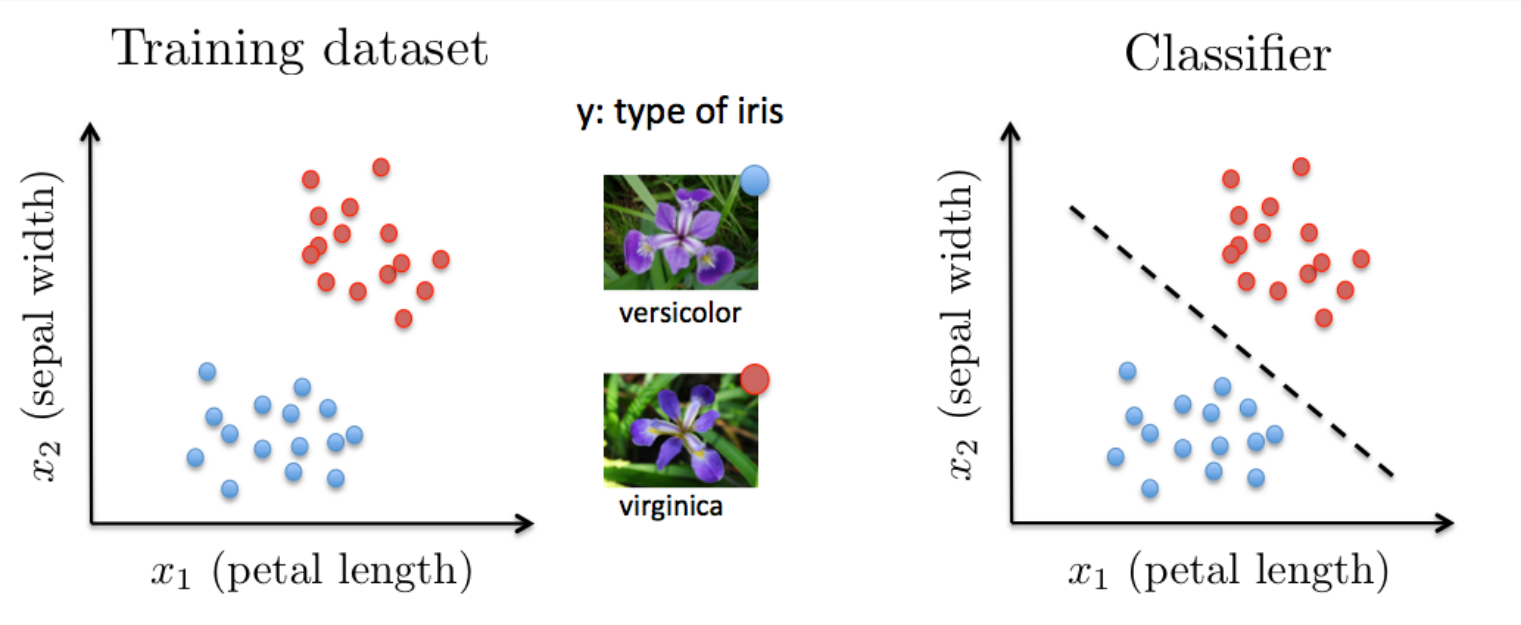
by Kenji Kawaguchi

MIT – oral presentation at NIPS 2016

Learning

Supervised (or Predictive) learning

Learn a mapping from inputs x to outputs y, given a labeled set of input-ouput pairs (the training set)
$D_n = \{(X_i, Y_i), i = 1, \ldots, n\}$



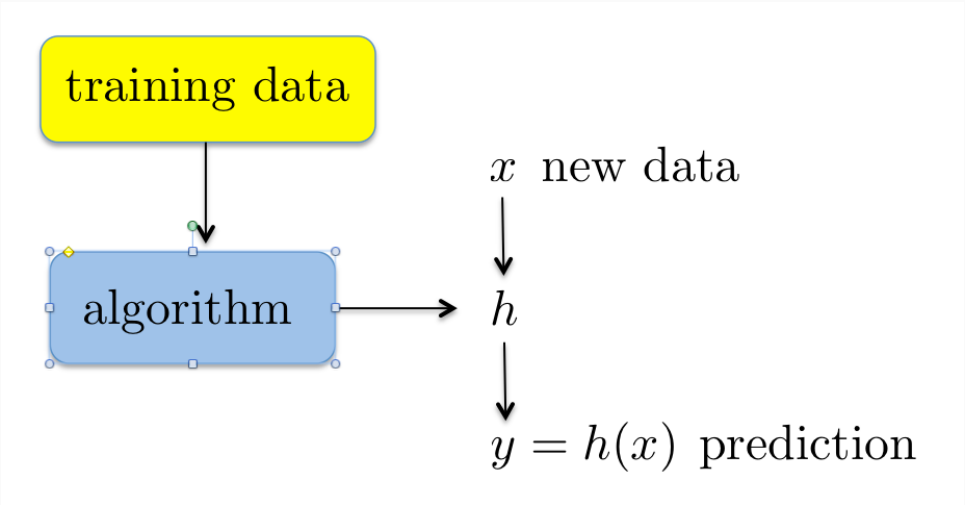
We learn the classification function $f = 1$ if versicolor, $f = -1$ if virginica

Supervised learning

- Training set: $D_n = \{(X_i, Y_i), i = 1, \ldots, n\}$
  - Input features: $X_i \in \mathbb{R}^d$
  - Output: Y_i

$$Y_i \in \mathcal{Y} \begin{cases} \mathbb{R} & \textbf{regression} \text{ (price, position, etc)} \\ \text{finite} & \textbf{classification} \text{ (type, mode, etc)} \end{cases}$$

- y is a non-deterministic and complicated function of x
  i.e., $y = f(x, z)$ where z is unknown (e.g. noise). Goal: learn f.
- Learning algorithm:



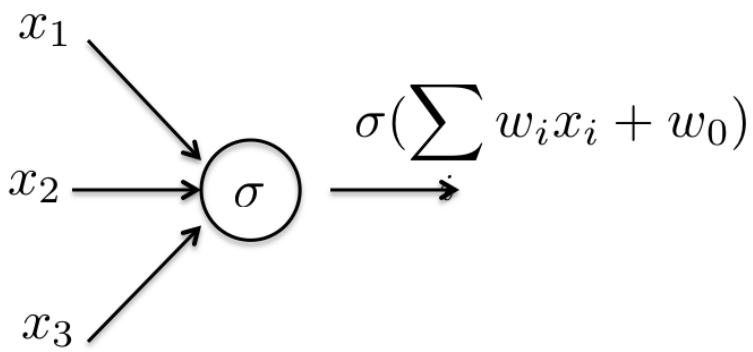
Objectives

- **Empirical risk:** $\hat{R}(f) := \frac{1}{n} \sum_{i=1}^{n} \|f(X_i) - Y_i\|_2^2$.

- Look for the mapping in a class of functions $\mathcal{F}$ that minimizes the risk (or a regularized version of it):

$$f^\star \in \arg\min_{f \in \mathcal{F}} \hat{R}(f).$$

Neural networks

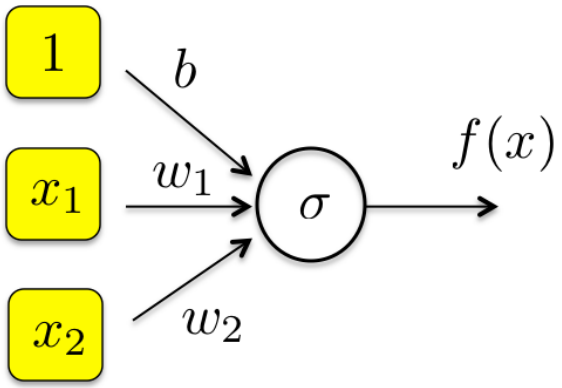


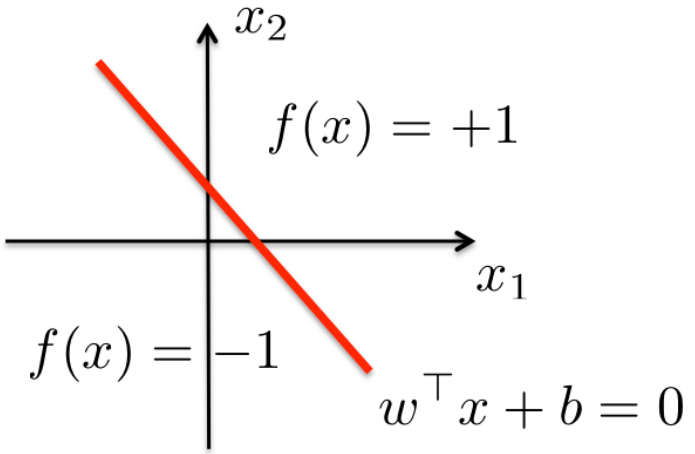
Loosely inspired by how the brain works[1]. Construct a network of simplified neurones, with the hope of approximating and learning any possible function

[1] Mc Culloch-Pitts, 1943

The perceptron

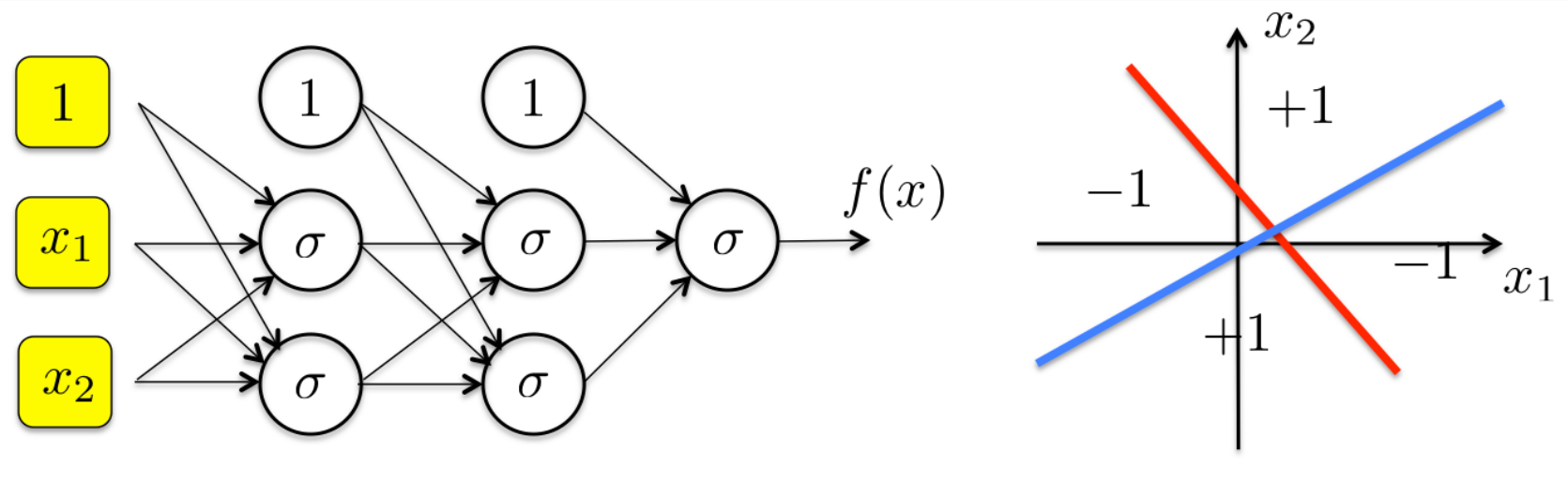
The first artificial neural network with one layer, and $\sigma(x) = \text{sgn}(x)$ (classification)
Input $x \in \mathbb{R}^d$, output in $\{-1, 1\}$. Can represent separating hyperplanes.





Multilayer perceptrons

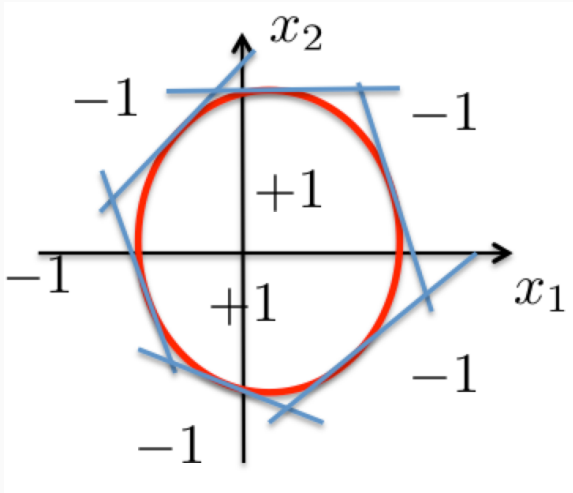
They can represent any function of $\mathbb{R}^d$ to $\{-1, 1\}$



... but the structure depends on the **unknown** target function f, and is difficult to optimise

From perceptrons to neural networks

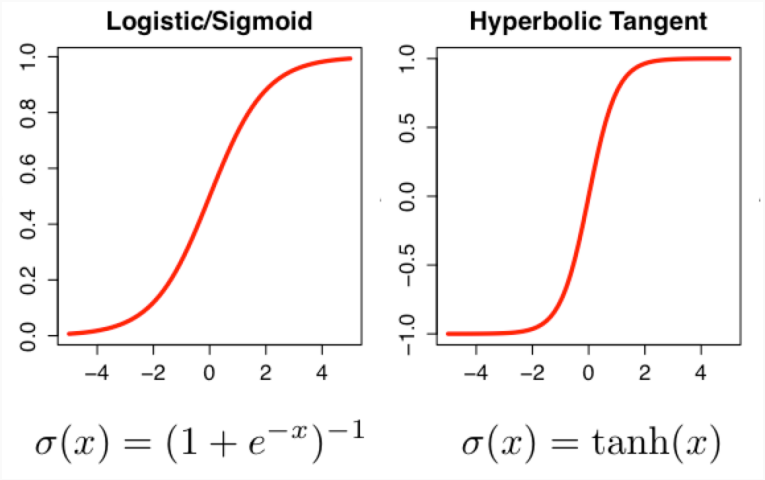
... and the number of layers can rapidly grow with the complexity of the function



A key idea to make neural networks practical: **soft-thresholding** ...

Soft-thresholding

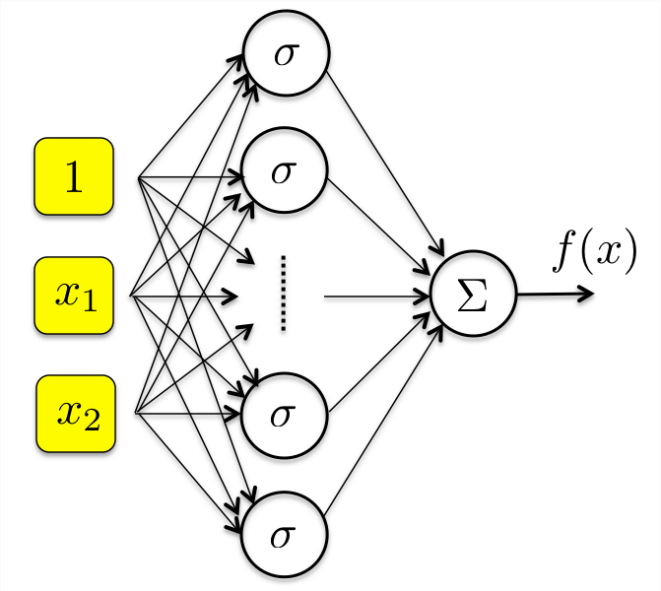
Replace hard-thresholding function σ by smoother functions



Theorem (Cybenko 1989) *Any continuous function f from $[0,1]^d$ to $\mathbb{R}$ can be approximated as a function of the form: $\sum_{j=1}^{N} \alpha_j \sigma(w_j^\top x + b_j)$, where σ is any sigmoid function.*

Soft-thresholding

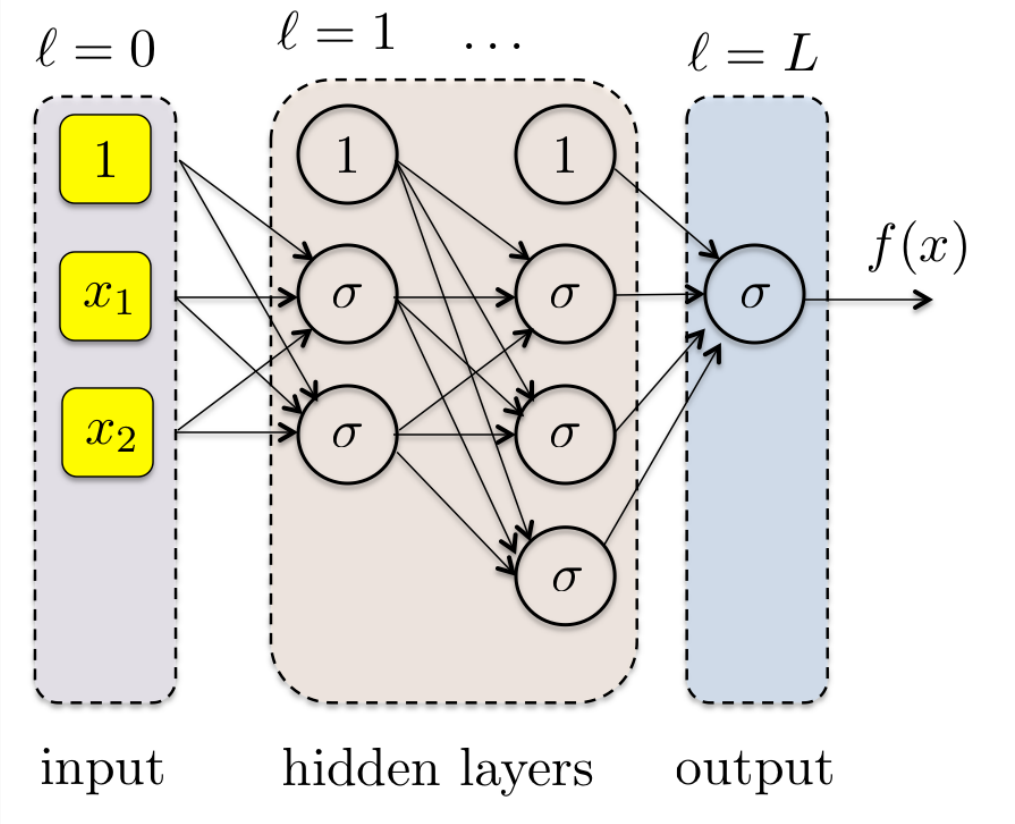
Cybenko's theorem tells us that f can be represented using a single hidden layer network ...



A non-constructive proof: how many neurones do we need? Might depend on f ...

Neural networks

A feedforward layered network (deep learning = enough layers)



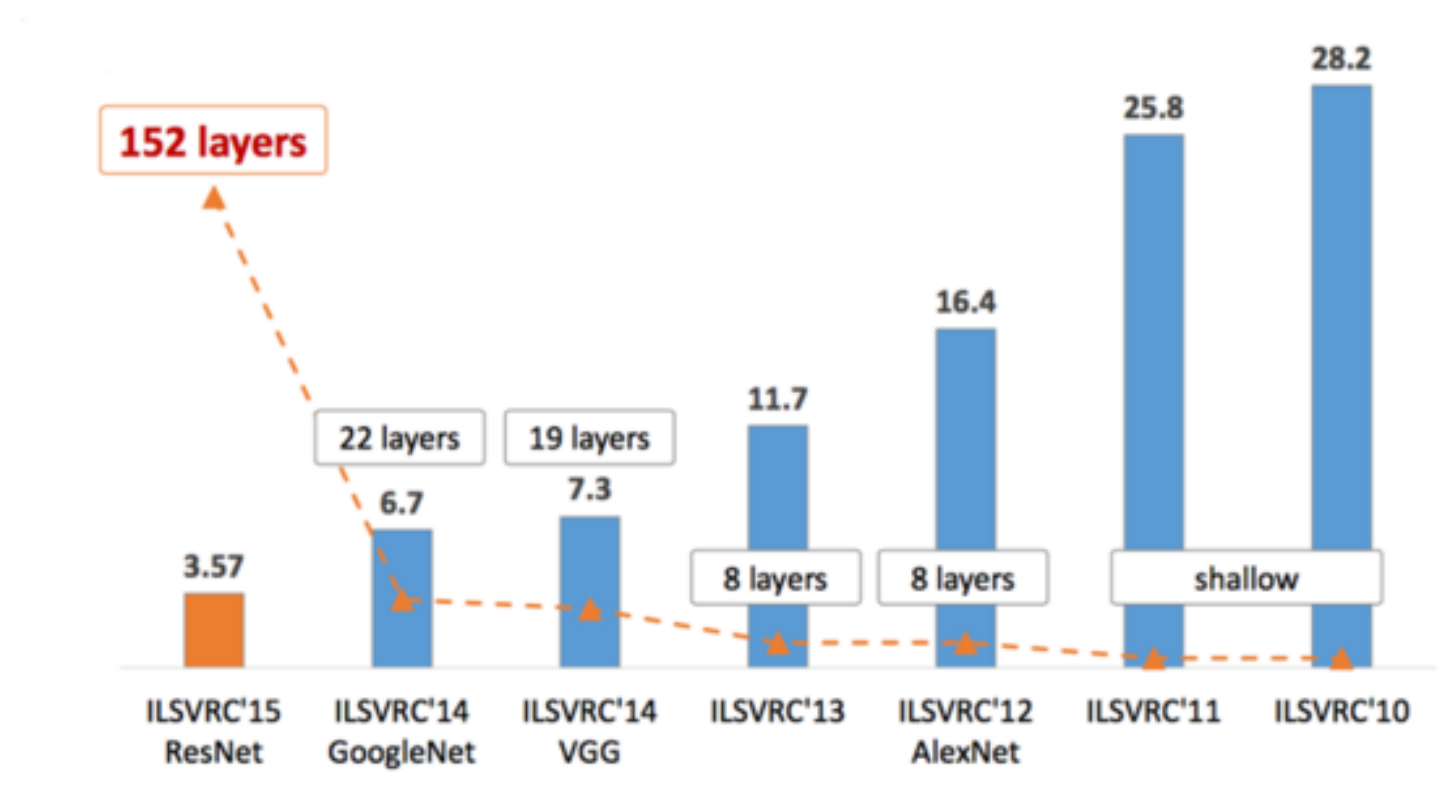
Deep Learning and the ILSVR challenge

Deep learning outperformed any other techniques in all major machine learning competitions (image classification, speech recognition and natural language processing)

The ImageNet Large Scale Visual Recognition Challenge (ILSVRC).

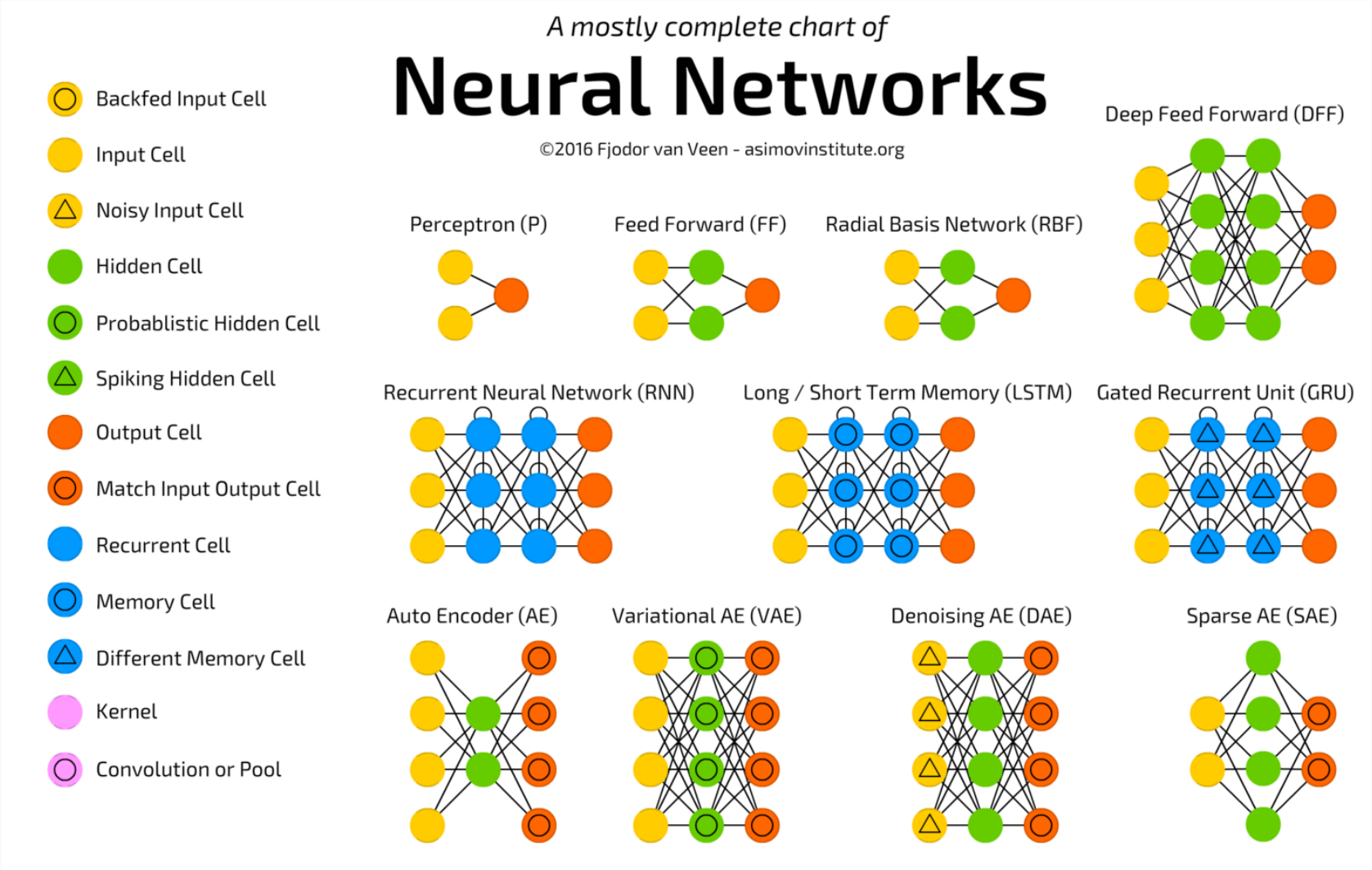
1. Training: 1.2 million images (227×227), labeled one out of 1000 categories
2. Test: 100.000 images (227×227)
3. Error measure: The teams have to predict 5 (out of 1000) classes and an image is considered to be correct if at least one of the predictions is the ground truth. [2]

ILSVR challenge

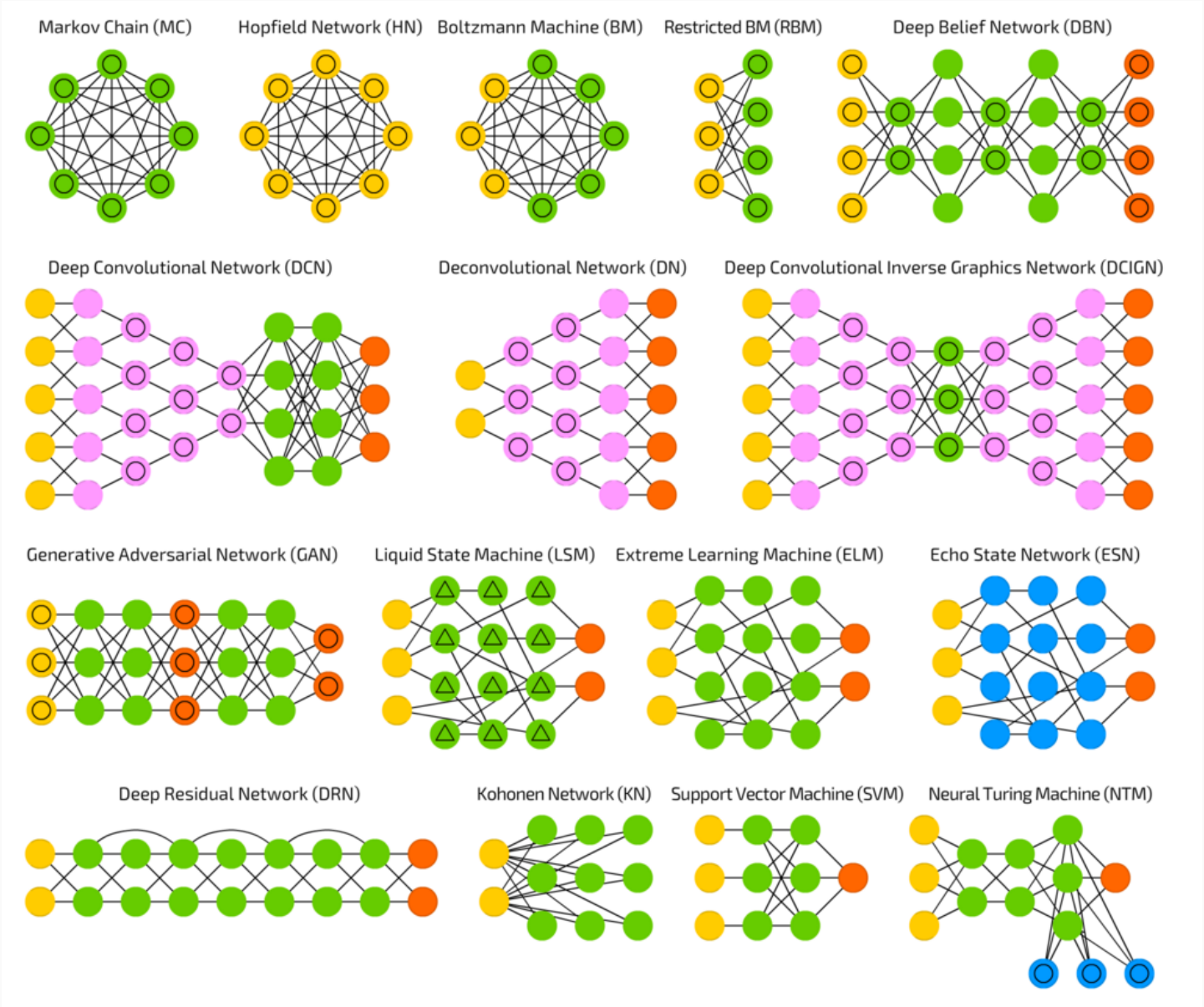


[1] From Stanford CS231n lecture notes

Architectures

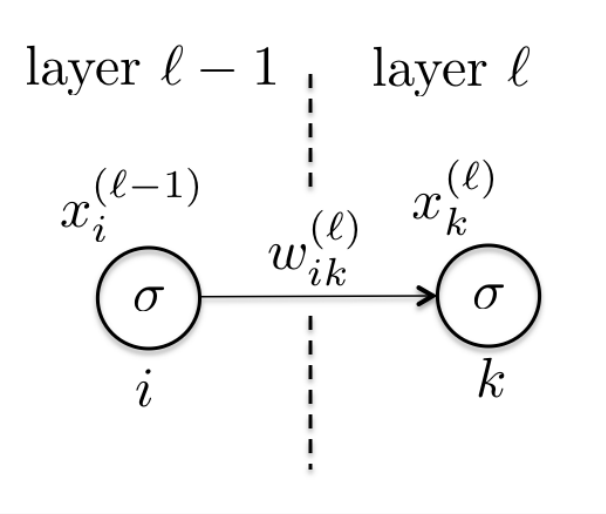


Architectures



Computing with neural networks

- Layer 0: inputs $x = (x_1^{(0)}, \ldots, x_d^{(0)})$ and $x_0^{(0)} = 1$
- Layer $1, \ldots, L-1$: hidden layer ℓ, $d^{(\ell)} + 1$ nodes, state of node i, $x_i^{(\ell)}$ with $x_0^{(\ell)} = 1$
- Layer L: output $y = x_1^{(L)}$



Signal at k: $s_k^{(\ell)} = \sum_{i=0}^{d^{(\ell-1)}} w_{ik}^{(\ell)} x_i^{(\ell-1)}$

State at k: $x_k^{(\ell)} = \sigma(s_k^{(\ell)})$

Output: the state of $y = x_1^{(L)}$

Training neural networks

The output of the network is a function of $\mathbf{w} = (w_{ij}^{(\ell)})_{i,j,\ell}$: $y = f_{\mathbf{w}}(x)$
We wish to optimise over $\mathbf{w}$ to find the most accurate estimation of the target function

Training data: $(X_1, Y_1), \ldots, (X_n, Y_n) \in \mathbb{R}^d \times \{-1, 1\}$

Objective: find $\mathbf{w}$ minimising the empirical risk:

$$E(\mathbf{w}) := R(f_{\mathbf{w}}) = \frac{1}{2n} \sum_{l=1}^{n} |f_{\mathbf{w}}(X_l) - Y_l|^2$$

Stochastic Gradient Descent

$E(\mathbf{w}) = \frac{1}{2n} \sum_{l=1}^{n} E_l(\mathbf{w})$ where $E_l(\mathbf{w}) := |f_{\mathbf{w}}(X_l) - Y_l|^2$
In each iteration of the SGD algorithm, only one function E_l is considered ...

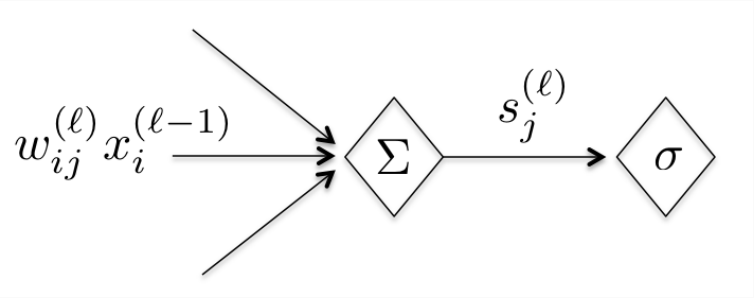
Parameter. learning rate $\alpha > 0$

1. **Initialization.** $\mathbf{w} := \mathbf{w}_0$
2. **Sample selection.** Select l uniformly at random in $\{1, \ldots, n\}$
3. **GD iteration.** $\mathbf{w} := \mathbf{w} - \alpha \nabla E_l(\mathbf{w})$, go to 2.

Is there an efficient way of computing $E_l(\mathbf{w})$?

Backpropagation

We fix l, and introduce $e(\mathbf{w}) = E_l(\mathbf{w})$.
Let us compute $\nabla e(\mathbf{w})$:



$$\frac{\partial e}{\partial w_{ij}^{(\ell)}} = \underbrace{\frac{\partial e}{\partial s_j^{(\ell)}}}_{:=\delta_j^{(\ell)}} \times \underbrace{\frac{\partial s_j^{(\ell)}}{\partial w_{ij}^{(\ell)}}}_{=x_i^{(\ell-1)}}$$

The sensitivity of the error w.r.t. the signal at node j can be computed recursively ...

Backward recursion

Output layer. $\delta_1^{(L)} := \frac{\partial e}{\partial s_1^{(L)}}$ and $e(\mathbf{w}) = (\sigma(s_1^{(L)}) - Y_l)^2$

$$\delta_1^{(L)} = 2(x_1^{(L)} - Y_l)\sigma'(s_1^{(L)})$$

From layer ℓ to layer $\ell - 1$.

$$\delta_i^{(\ell-1)} := \frac{\partial e}{\partial s_i^{(\ell-1)}} = \sum_{j=1}^{d^{(\ell)}} \underbrace{\frac{\partial e}{\partial s_j^{(\ell)}}}_{:=\delta_j^{(\ell)}} \times \underbrace{\frac{\partial s_j^{(\ell)}}{\partial x_i^{(\ell-1)}}}_{=w_{ij}^{(\ell)}} \times \underbrace{\frac{\partial x_i^{(\ell-1)}}{\partial s_i^{(\ell-1)}}}_{=\sigma'(s_i^{(\ell-1)})}$$

Summary.

$$\frac{\partial E_l}{\partial w_{ij}^{(\ell)}} = \delta_j^{(\ell)} x_i^{(\ell-1)}, \quad \delta_i^{(\ell-1)} = \sum_{j=1}^{d^{(\ell)}} \delta_j^{(\ell)} w_{ij}^{(\ell)} \sigma'(s_i^{(\ell-1)})$$

Backpropagation algorithm

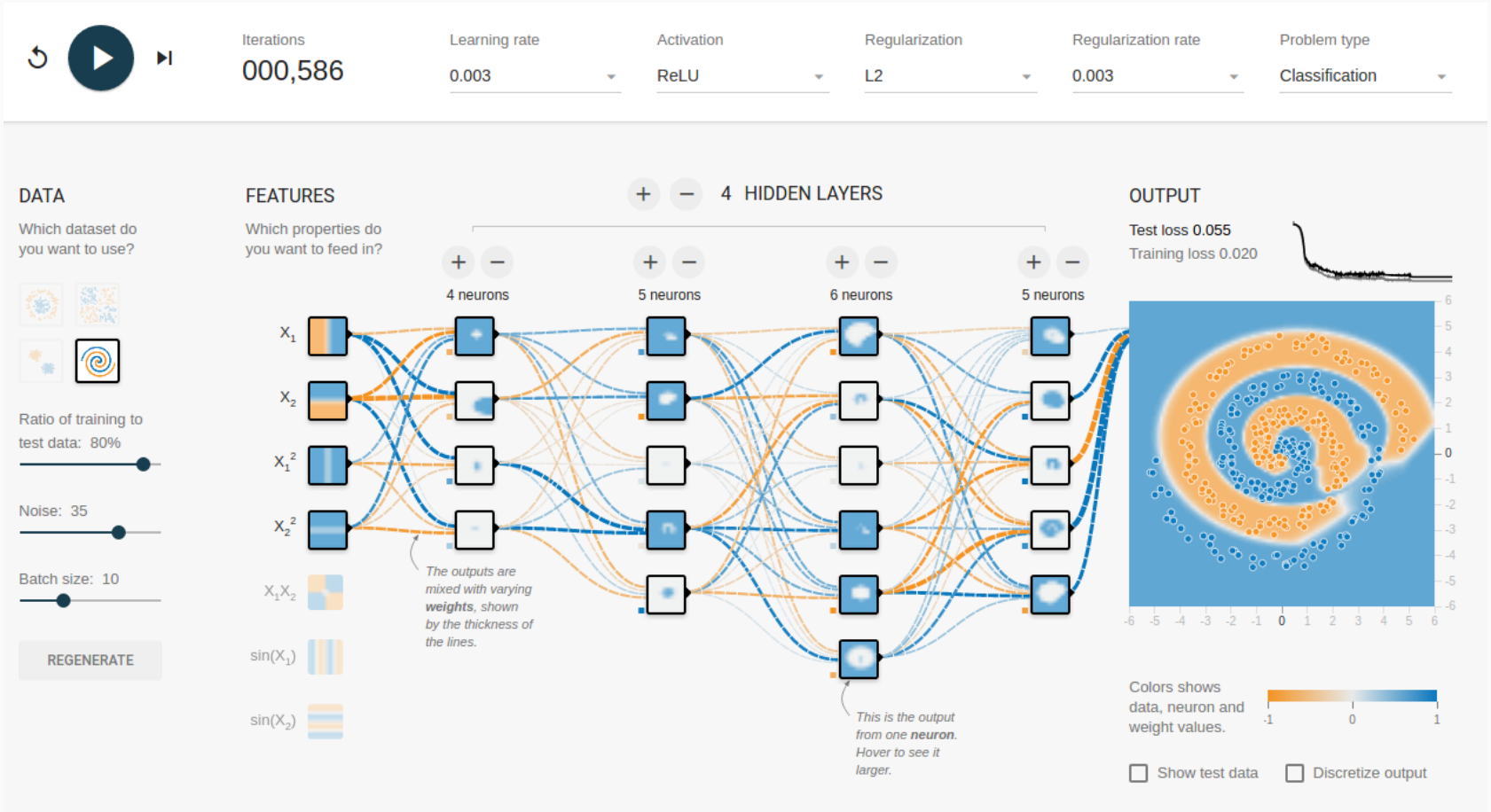
Parameter. Learning rate $\alpha > 0$

Input. $(X_1, Y_1), \ldots, (X_n, Y_n) \in \mathbb{R}^d \times \{-1, 1\}$

1. **Initialization.** $\mathbf{w} := \mathbf{w}_0$
2. **Sample selection.** Select l uniformly at random in $\{1, \ldots, n\}$
3. **Gradient of E_l.**
   - $x_i^{(0)} := X_{li}$ for all $i = 1, \ldots d$
   - Forward propagation: compute the state and signal at each node $(x_i^{(\ell)}, s_i^{(\ell)})$
   - Backward propagation: propagate back Y_l to compute $\delta_i^{(\ell)}$ at each node and the partial derivative $\frac{\partial E_l}{\partial w_{ij}^{(\ell)}}$
4. **GD iteration.** $\mathbf{w} := \mathbf{w} - \alpha \nabla E_l(\mathbf{w})$, go to 2.

Example: tensorflow

http://playground.tensorflow.org/



Deep Learning without Local Minima

Critical question: The SGD algorithm will converge to a global minimum of the risk, if we can guarantee that local minima have the same risk as a global minimum. What does the loss surface look like?

Related work:

- P. Baldi, K. Hornik. Neural Networks and PCA: Learning from Examples without Local Minima. *Neural Networks*, 1989.
- I. Goodfellow, Y. Bengio, A. Courville. Deep Learning, http://www.deeplearningbook.org
- A. Choromanska et al.. The Loss Surface of Multilayer Networks. ICML 2015.

Notations

- Data: $X_i \in \mathbb{R}^{d_x}$, $Y_i \in \mathbb{R}^{d_y}$, m data points
  X: $d_x \times m$ matrix whose columns are the X_i's
  Y: $d_y \times m$ matrix whose columns are the Y_i's
- H hidden layers
- Layer k with d_k neurons, input weight matrix $W_k \in \mathbb{R}^{d_k \times d_{k-1}}$
- $p = \min\{d_1, \ldots, d_H\}$
- Output:

$$\hat{Y}(W, X) = q\sigma_{H+1}(W_{H+1}\sigma(W_H\sigma(W_{H-1} \ldots ...\sigma(W_2\sigma(W_1 X)\ldots)$$

  Linear activation function: $\hat{Y}(W, X) = W_{H+1} \ldots W_1 X$.

Baldi-Hornik: 1-hidden linear networks

- Linear regression: fitting a linear model to the data.
  $X_i \in \mathbb{R}^{d_x}$, $Y_i \in \mathbb{R}^{d_y}$.
  Find the matrix $L^\star \in \mathbb{R}^{d_y \times d_x}$ minimizing
  $\mathcal{L}(L) = \sum_{i=1}^m \|Y_i - LX_i\|_2^2$.
  When $XX^\top$ is invertible, L is equal to $L^\star = YX^\top(XX^\top)^{-1}$.
  Convexity of $\mathcal{L}$.

- Now in a 1-hidden layer network, we are looking for L that can be factorized as W_2W_1 where $W_1 \in \mathbb{R}^{p \times d_x}$ and $W_2 \in \mathbb{R}^{d_y \times p}$.
  In particular the rank of L is at most p.
  Non uniqueness: $W_1' = CW_1$ and $W_2' = W_2C^{-1}$ work as well.

Baldi-Hornik: 1-hidden linear networks

- Define the $d_y \times d_y$ matrix $\Sigma = YX^\top (XX^\top)^{-1} XY^\top$ (covariance matrix of the best unconstrained linear approximation of Y).
  $d_x = d_y$.

Theorem (Baldi-Hornik 1989) Assume that Σ is full rank, and has d_y distinct eigenvalues $\lambda_1 > \ldots > \lambda_{d_y}$. Let W_1 and W_2 define a critical point of $\mathcal{L}(W_1, W_2)$. Then there exists a subset Γ of p (orthonormal) eigenvectors of Σ, and a $p \times p$ invertible matrix C such that:

$$W_2 = U_\Gamma C, \quad W_1 = C^{-1} U_\Gamma^\top YX^\top (XX^\top)^{-1},$$

where U_Γ is the matrix formed by the eigenvectors in Γ.
Moreover $\mathcal{L}(W_1, W_2) = \text{trace}(YY^\top) - \sum_{i \in \Gamma} \lambda_i$.

Baldi-Hornik: 1-hidden linear networks

Theorem (Baldi-Hornik 1989) Assume that Σ is full rank, and has d_y distinct eigenvalues $\lambda_1 > \ldots > \lambda_{d_y}$. Let W_1 and W_2 define a critical point of $\mathcal{L}(W_1, W_2)$. Then there exists a subset Γ of p (orthonormal) eigenvectors of Σ, and a $p \times p$ invertible matrix C such that:

$$W_2 = U_\Gamma C, \quad W_1 = C^{-1} U_\Gamma^\top Y X^\top (X X^\top)^{-1},$$

where U_Γ is the matrix formed by the eigenvectors in Γ.
Moreover $\mathcal{L}(W_1, W_2) = \text{trace}(YY^\top) - \sum_{i \in \Gamma} \lambda_i$.

Up to C, the global minimizer is unique, and is the projection on the subspace spanned by the p top eigenvectors of Σ of the ordinary least square regression matrix!
Taking an other set of eigenvectors for the projection yields a saddle point.

This paper: networks with any depth and width

Theorem (Kawaguchi 2016) Assume that $XX^\top$ and $XY^\top$ are full rank, $d_x \geq d_y$. Assume that Σ is full rank, and has d_y distinct eigenvalues. The loss function $\mathcal{L}(W_1, \ldots, W_{H+1})$ satisfies:

(i) it is non-convex and non-concave.

(ii) Every local minimum is a global minimum.

(iii) Every critical point that is not a minimum is a saddle point.

(iv) If $\text{rank}(W_H, \ldots, W_2) = p$, then the Hessian at any saddle point has at least one strictly negative eigenvalue.

Proof sketch: example

Assume that W is a critical point and a local minimum, and that $\text{rank}(W_H \dots W_2) = p$

- Necessary conditions: $\nabla \mathcal{L} = 0$ and $\nabla^2 \mathcal{L}$ positive semidefinite.
- From the latter conditions, we deduce that $X(\hat{Y}(W, X) - Y)^\top = 0$.
- Go back to the unconstrained linear case: $f(W') = \|W'X - Y\|_F^2$ for $W' \in \mathbb{R}^{d_y \times d_x}$. Let $r' = (W'X - Y)^\top$ denote the error matrix. By convexity, if $Xr' = 0$ then W' is a global minimizer of f. Now with $W' = W_{H+1} \dots W_1$, we have $Xr = Xr' = 0$, and hence W' is a global minimizer of f.

This paper: non-linear networks with any depth and width

Rectified linear activation function: $\sigma(x) = \max(0, x)$.
Output:

$$\hat{Y}(W, X) = q\sigma_{H+1}(W_{H+1}\sigma(W_H\sigma(W_{H-1} \ldots ...\sigma(W_2\sigma(W_1 X)\ldots)$$

An other way of writing the output:

$$\hat{Y}(W, X) = q \sum_{i=1}^{d_x} \sum_{j=1}^{\gamma} X_{i,j} A_{i,j} \prod_{k=1}^{H} W_{i,j}^{(k)},$$

where the first sum is voer the input coordinates, the second sum is on the path from the i-th input to the output, $X_{i,j} = X_{i,1}$ for all j is the i-th input, and $A_{i,j}$ is the activation (binary variable) of the path j for input i

This paper: non-linear networks with any depth and width

Critical simplification: the $A_{i,j}$'s are independent Bernoulli r.v. with mean ρ!
Under this assumption (among others), there is an equivalence with a linear network.

The previous theorem holds ...

Summary

SGD could well find global minimizer of the empirical risk, under some conditions ...
What is the impact of regularization?
What about other activation functions?