2.160 System Identification, Estimation, and Learning

Lecture Notes No. 13

March 22, 2006

8. Neural Networks

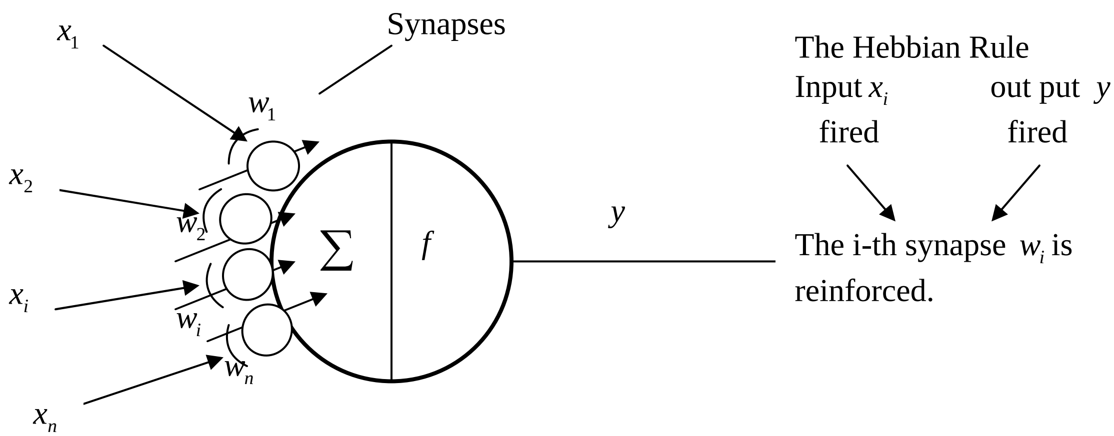
8.1 Physiological Background

Neuro-physiology

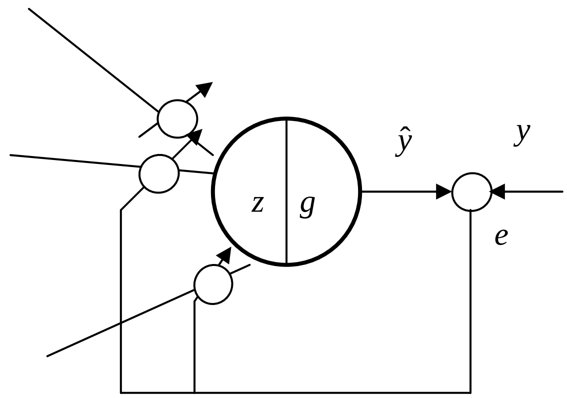
- A Human brain has approximately 14 billion neurons, 50 different kinds of neurons. … uniform
- Massively-parallel, distributed processing
  Very different from a computer (a Turing machine)

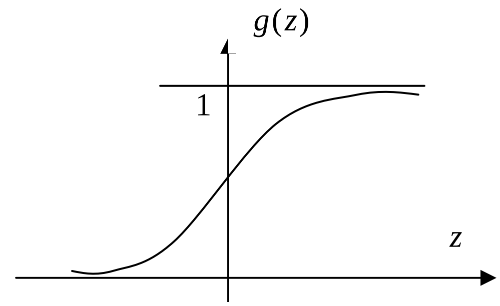
Image removed due to copyright reasons.

McCulloch and Pitts, Neuron Model 1943
Donald Hebb, Hebbian Rule, 1949
…Synapse reinforcement learning
Rosenflatt, 1959
…The perceptron convergence theorem



The electric conductivity increases at w_i





logistic function, or
sinusoid function

(1) Error $e = \hat{y} - y = g(z) - y \qquad z = \sum_{i=1}^{n} w_i x_i$

Gradient Descent method

(2) $\Delta w_i = -\rho \cdot grad_{w_i} e^2 = -\rho 2e \dfrac{\partial e}{\partial w_i}$

ρ : learning rate

(3) $\therefore \quad \Delta w_i = -2\rho g' e x_i$

$\Delta w_i \propto$ (Input x_i).(Error)

Supervised Learning

(4) $g(z) = \dfrac{1}{1 + e^{-z}}$

Unsupervised Learning.

Replacing e by $\hat{y}$ yields the Hebbian Rule

$\Delta w_i \propto$ (Input x_i).(output $\hat{y}$)

8.2 Stochastic Approximation

consider a linear output function for $\hat{y} = g(z)$:

(5) $\hat{y} = \sum_{i=1}^{n} w_i x_i$

Given N sample data $\left\{ \left(y^j, x_1^j, \ldots x_n^j \right) \middle| \; j = 1, \ldots N \right\}$ Training Data

Find $w_1, \ldots, w_n$ that minimize

(6) $J_N = \dfrac{1}{N} \sum_{i=1}^{n} (\hat{y}^j - y^j)$

(7) $\Delta w_i = -\rho \cdot grad_{w_i} J_N = -\rho \dfrac{2}{N} \sum_{j=1}^{N} (\hat{y}^j - y^j) \dfrac{\partial \hat{y}^j}{\partial w_i}$

This method requires to store the gradient $\left(\hat{y}^j - y^j \right) \dfrac{\partial \hat{y}^j}{\partial w_i}$ for all the sample date j=1,…N:

It is a type of batch processing.

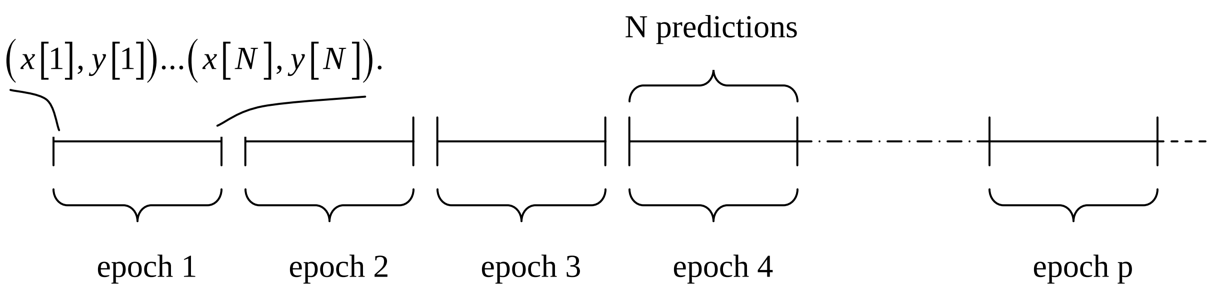
A simpler method is to execute updating the weight Δw_i every time the training data is presented.

(8) $\Delta w_i[k] = \rho\delta[k]x_i[k]$ for the *k-th* presentation

(9) Where $\delta(k) = y[k] - \sum w_l[k]x_l[k]$

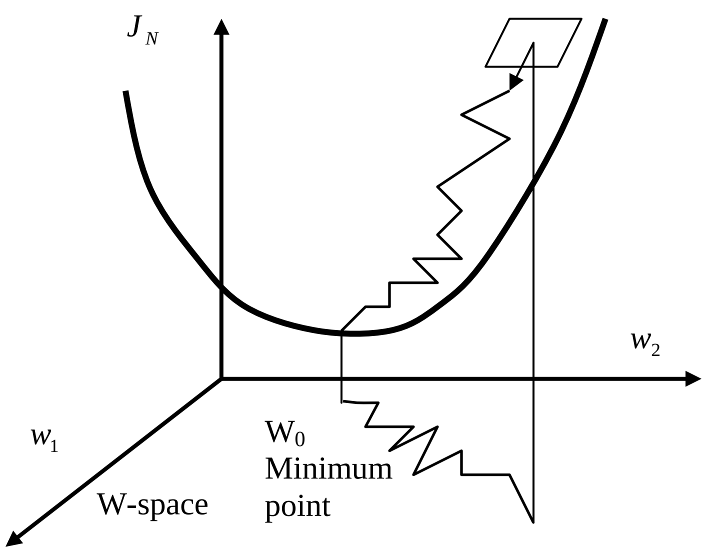
Correct output for the training data presented at the k -th time	Predicted output based on the weights $w_i[k]$ for the training data presented at the k -th time

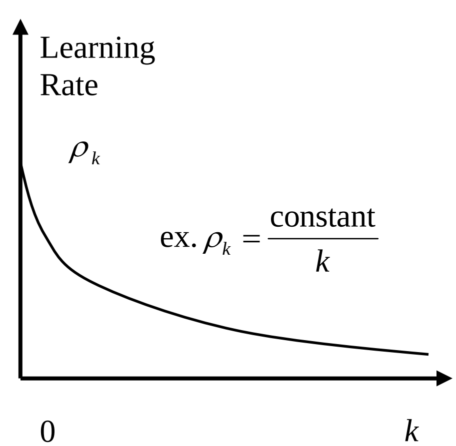
Learning procedure
Present all the *N* training data in any sequence, and repeat the *N* presentations, called an "epoch", many times… Recycling.



This procedure is called the Widrow-Hoff algorithm.
Convergence: As the recycling is repeated infinite times, does the weight vector converge to the optimal one: $\arg\min_{w} J_N(w_1 ... w_n)$?... Consistency

If a constant learning rate $\rho > 0$ is used, this does not converge, unless $\min J_N = 0$





If the learning rate is varied, eg. $\rho_k = \frac{\text{constant}}{k}$, convergence can be guaranteed.
This is a special case of the Method of Stochastic Approximation.

Expected Loss Function

(10) $E[L(w)] = \int L(x, y|w)p(x)dx$

(11) $\int L(x, y|w) = \frac{1}{2}\left(y - \hat{y}(x|w)\right)^2$

The stochastic approximation procedure for minimizing this expected loss function with respect to weight $w_1, \ldots, w_n$ is

(12) $w_i[k+1] = w_i[k] - \rho[k]\frac{\partial}{\partial w_i}L(x[k], y[k]|w[k])$

Where $x[k]$ is the training data presented at the k-th iteration. This estimate is proved consistent if the learning rate $\rho[k]$ satisfies.

1) $\lim_{k\to\infty} \rho[k] = 0$

(13) 2). $\lim_{k\to\infty} \sum_{i=1}^{k} \rho[i] = +\infty$ ⟶ This condition prevents all the weights from converging so fast that error will remain forever uncorrected.

$\lim_{k\to\infty} \sum_{i=1}^{k} \rho[i]^2 < \infty$ ⟶ This condition ensures that random fluctuations are eventually suppressed

(14) $\lim_{k\to\infty} E[(w_i[k] - w_{i0})^2] = 0$;

The estimated weights converge to their optimal values with probability of 1.
Robbins and Monroe, 1951

This stochastic Approximation method in general needs more presentation of data, i.e. the convergence process is slower than the batch processing. But the computation is very simple.

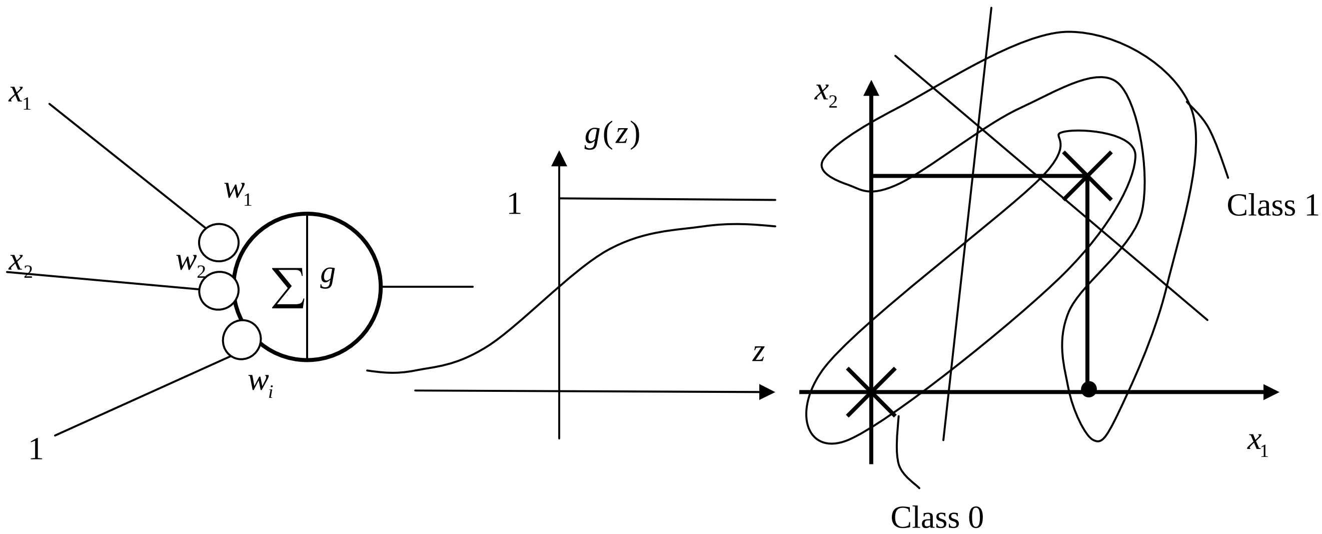
8.3 Multi-Layer Perceptrons

The Exclusive OR Problem

Input		Output
0	0	0
0	1	1
1	0	1
1	1	0
X_1	X_2	y

Can a single neural unit (perceptron) with weights w_1, w_2, w_3, produce the XOR truth table?

No, it cannot



(15) $z = w_1 x_1 + w_2 x_2 + w_3$

Set z=0, then $0 = w_1 x_1 + w_2 x_2 + w_3$ represents a straight line in the $x_1 - x_2$ plane.

(16) $g(z) = \begin{cases} 1 & z > 0 \\ 0 & z \le 0 \end{cases}$

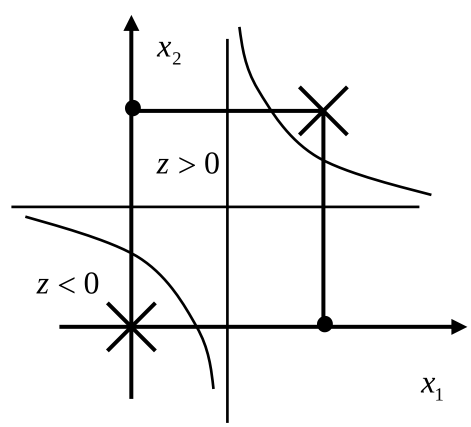
Class 0 and class 1 cannot be separated by a straight line. … Not linearly separable.

Consider a nonlinear function in lieu of (15)

(17) $z = f(x_1, x_2) = x_1 + x_2 - 2x_1 x_2 - \frac{1}{3}$

$f(0,0) = -\frac{1}{3}$, $f(1,1) = -\frac{1}{3}$ → Class 0

$f(1,0) = f(0,1) = \frac{2}{3} > 0$ → Class 1



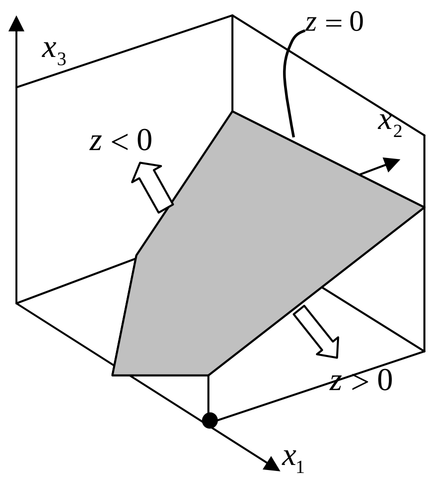
Next, replace $x_1\ x_2$ by a new variable x_3

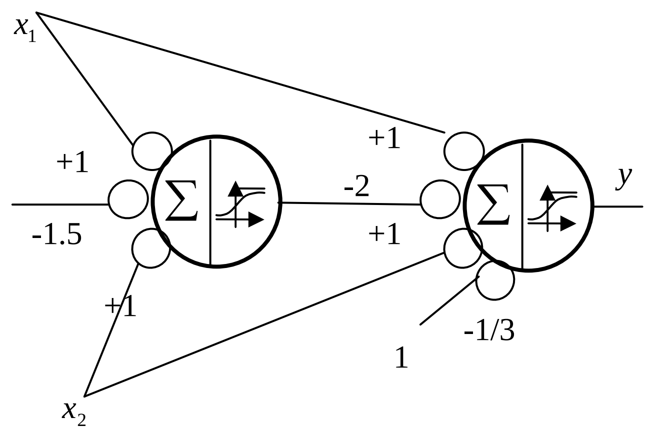
(18) $z = x_1 + x_2 - 2x_3 - \frac{1}{3}$

This is apparently a linear function: Linearly Separable.

Hidden Units

- Augment the original input patterns
- Decode the input and generate tan internal representation





Hidden Unit
Not directly visible from output

Extending this argument, we arrive at a multi-layer network having multiple hidden layers between input and output layers.

Multi-Layer Perception