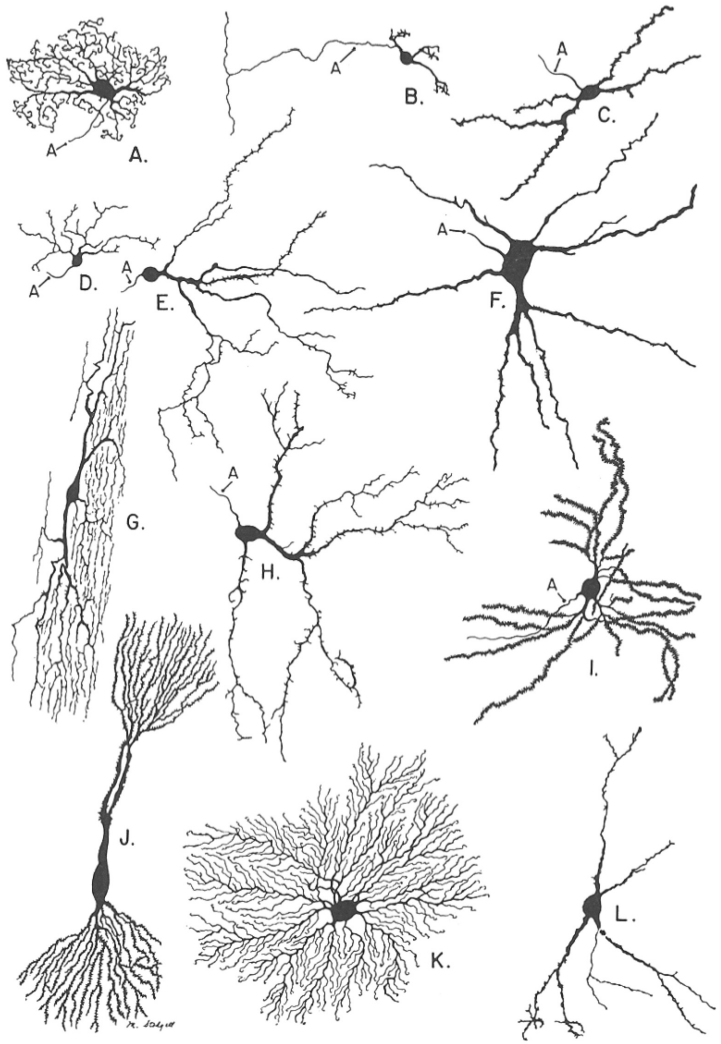
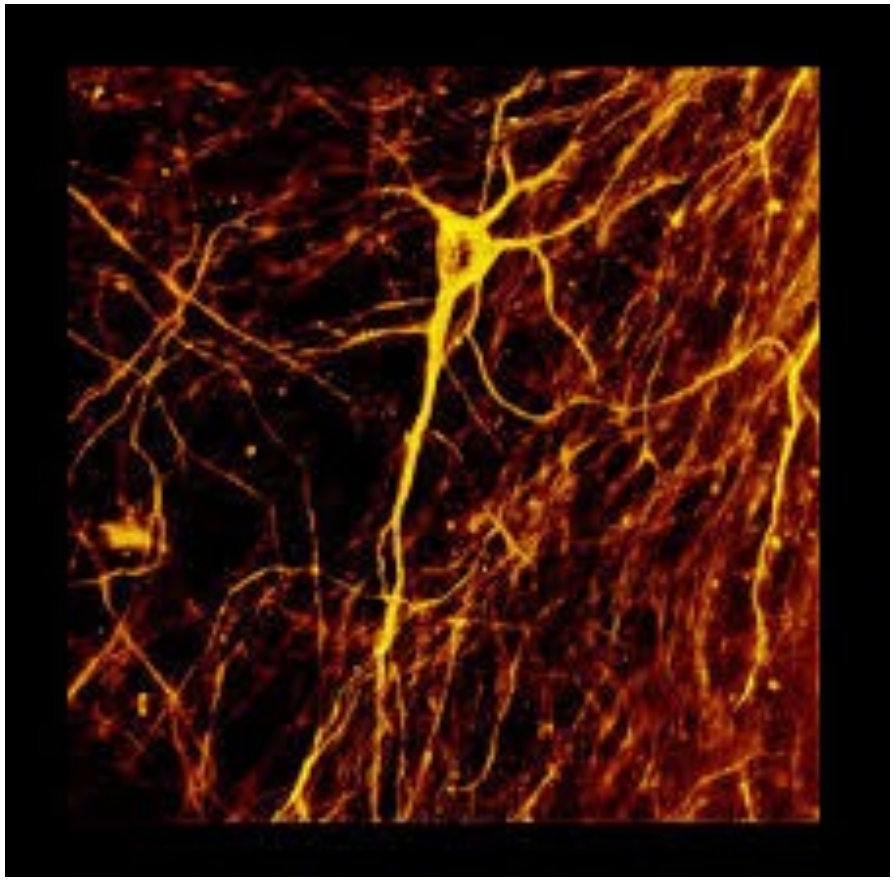
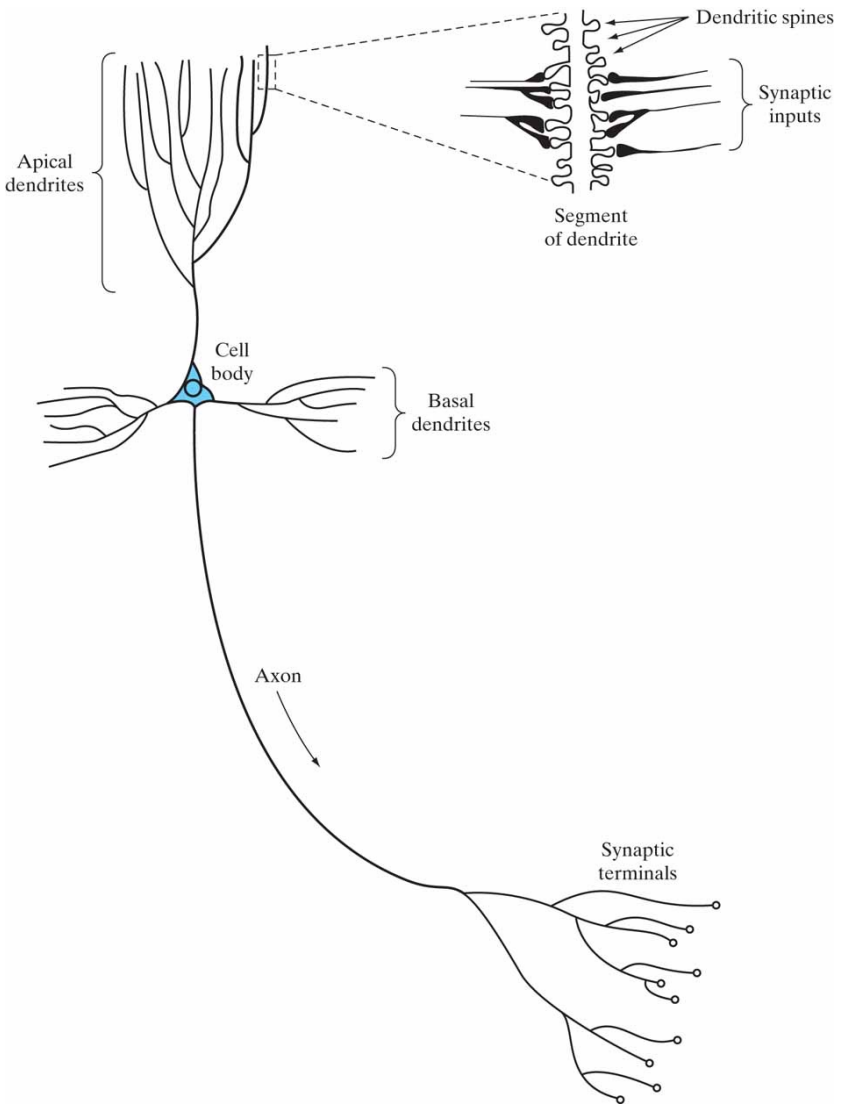
Lecture 7: Perceptron

Andreas Wichert
Department of Computer Science and Engineering
Técnico Lisboa

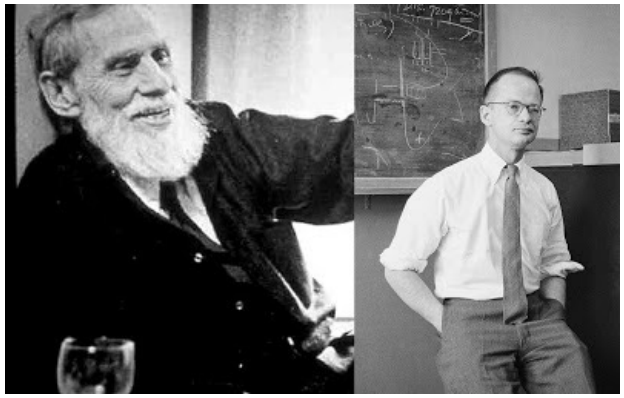
- 10^{40} Neurons
- $10^{4\text{-}5}$ connections per neuron



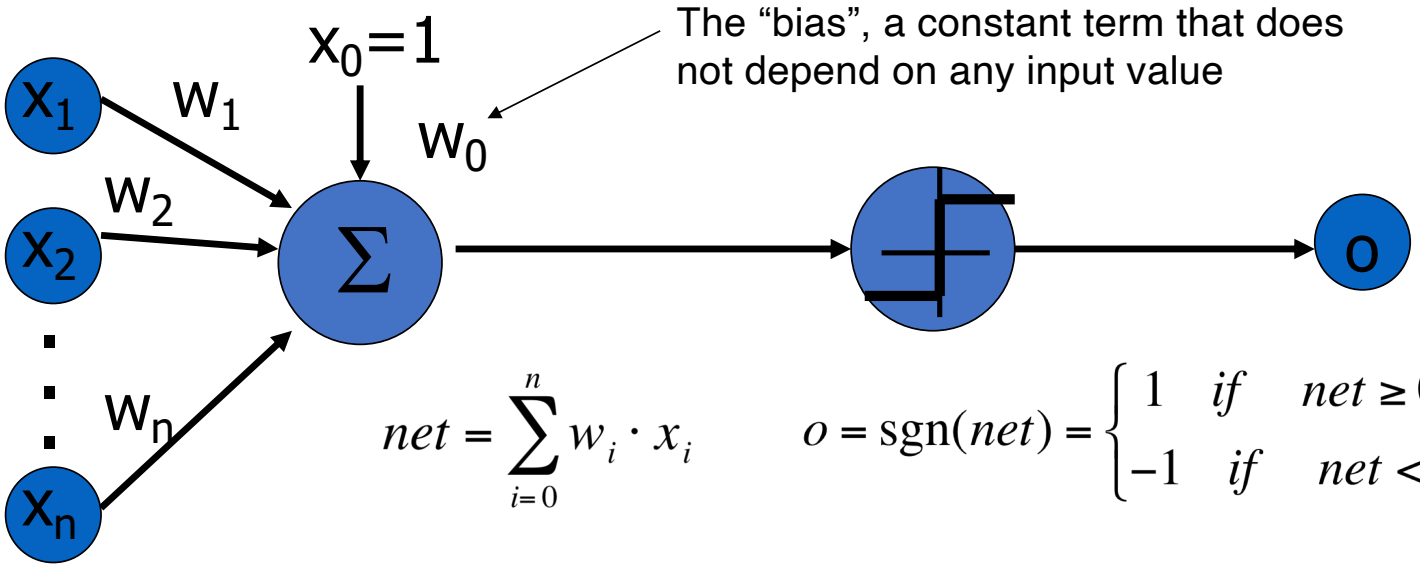



Dendritic spines
Synaptic inputs
Apical dendrites
Segment of dendrite
Cell body
Basal dendrites
Axon
Synaptic terminals

Perceptron (1957)

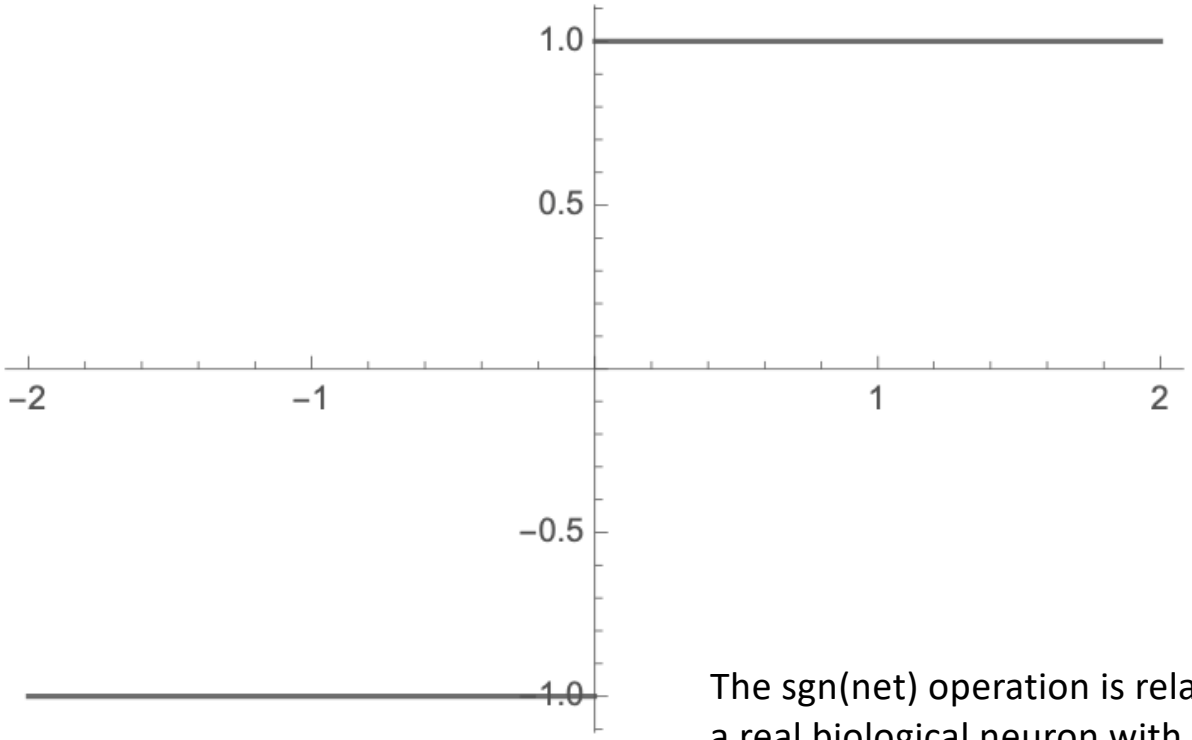


- Linear threshold unit (LTU)



$$net = \sum_{i=0}^{n} w_i \cdot x_i \qquad o = \mathrm{sgn}(net) = \begin{cases} 1 & if \quad net \geq 0 \\ -1 & if \quad net < 0 \end{cases}$$

McCulloch-Pitts model of a neuron (1943)



The sgn(net) operation is related to the threshold operation of a real biological neuron with −1 not firing and 1 firing. It should be noted that sgn is non continuous

$$o = sgn\left(\sum_{j=0}^{D} w_j \cdot x_j\right) = sgn\left(\langle \mathbf{x}|\mathbf{w}\rangle + w_0 \cdot x_0\right)$$

Simplified, the perceptron tries to learns the function $f(\mathbf{x})$

$$f(\mathbf{x}) = \begin{cases} 1 & \text{if} \quad \langle \mathbf{x} | \mathbf{w} \rangle + w_0 \geq 0 \\ -1 & \text{otherwise,} \end{cases}$$

In a D dimensional space, this straight line becomes a hyperplane with equation

$$-w_0 = \langle \mathbf{x} | \mathbf{w} \rangle$$

$$0 = \langle \mathbf{x} | \mathbf{w} \rangle + w_0.$$

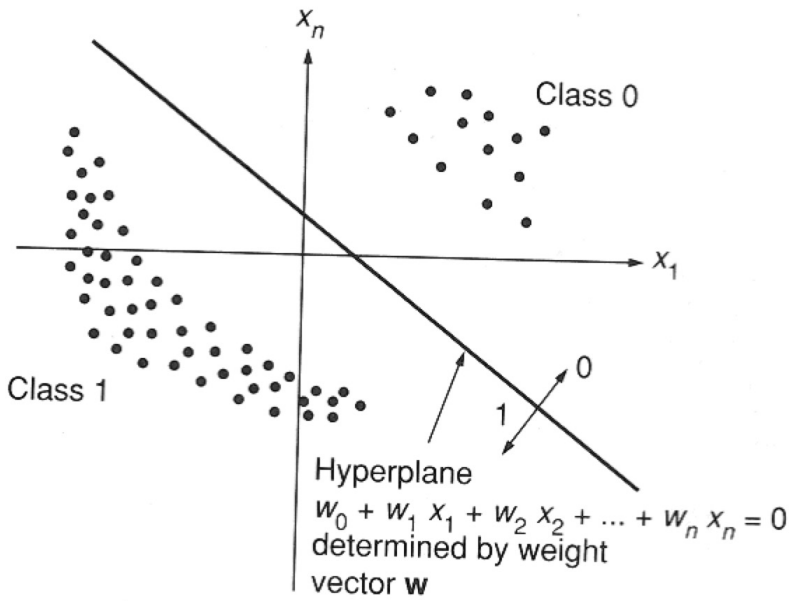
Linearly separable patterns

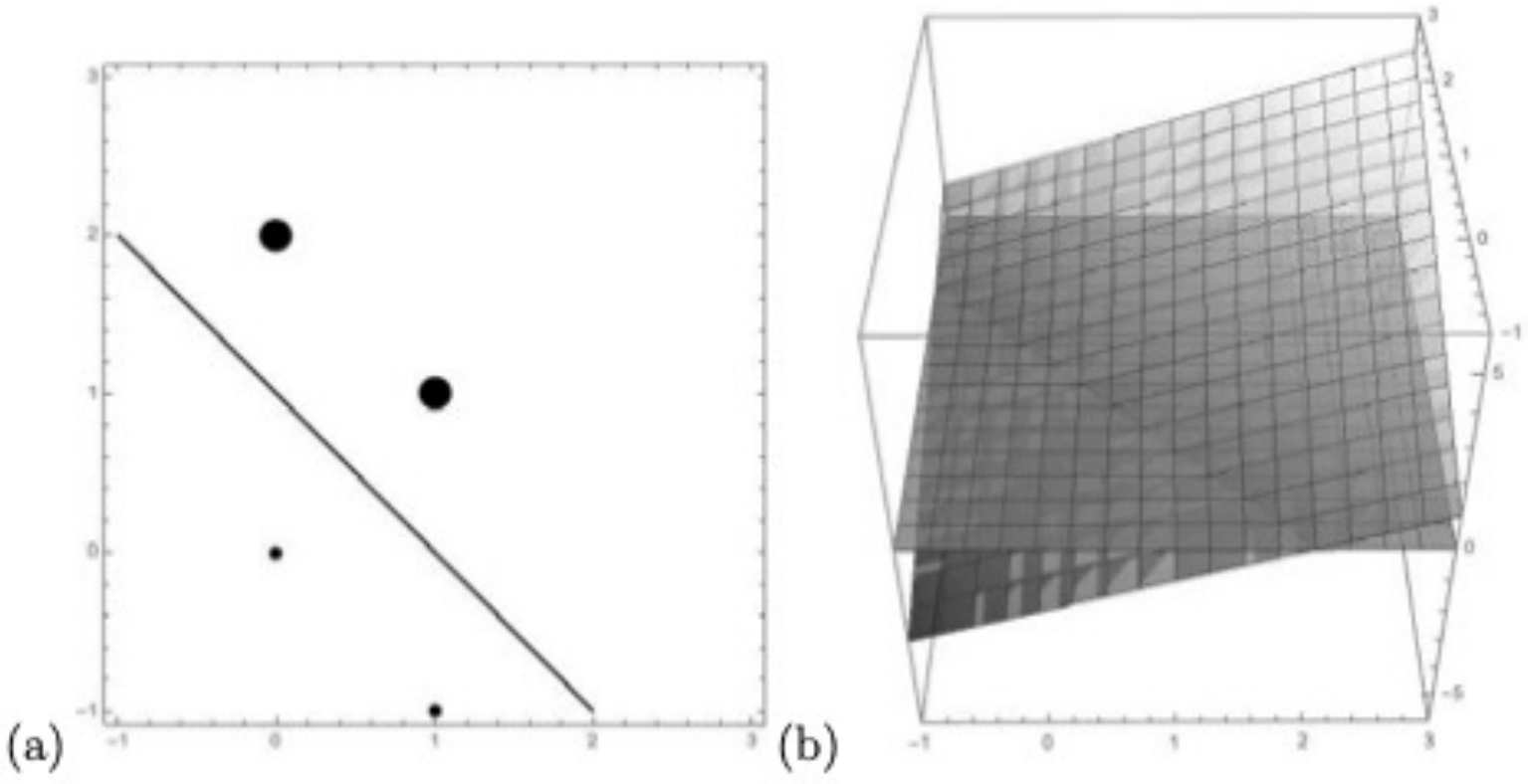
$$o = \mathrm{sgn}(\sum_{i=0}^{n} w_i x_i)$$

$$\sum_{i=0}^{n} w_i x_i > 0 \quad for \quad C_0$$

$$\sum_{i=0}^{n} w_i x_i \leq 0 \quad for \quad C_1$$

$X_0=1$, bias...



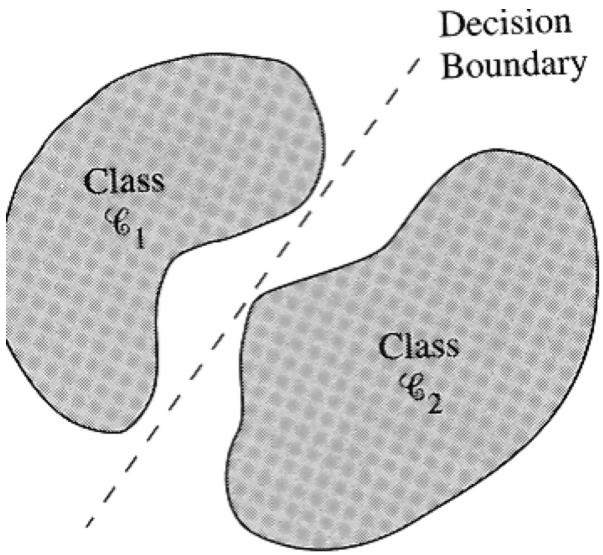


(a) The two classes 1 (indicated by a big point) and −1 (indicated by a small point) are separated by the line *−1 + x1 + x2 = 0.*

(b) The hyperplane *−1+x1+x2 =y* defines the line for y=0.

- The goal of a perceptron is to correctly classify the set of pattern $\boldsymbol{D}=\{\boldsymbol{x}_1,\boldsymbol{x}_2,..\boldsymbol{x}_m\}$ into one of the classes C_1 and C_2
- The output for class C_1 is $o=1$ and for C_2 is $o=-1$

- For $n=2$ ➔



Perceptron learning rule

- Consider linearly separable problems
- How to find appropriate weights
  - Initialize each vector ***w*** to some small *random* values
- Look if the output pattern *o* belongs to the desired class, has the desired value *d*

$$w^{new} = w^{old} + \Delta w \qquad \Delta w = \eta \cdot (d - o) \cdot x$$

- η is called the **learning rate**
- $0 < \eta \leq 1$

- In supervised learning the network has its output compared with known correct answers
  - Supervised learning
  - Learning with a teacher
- *(d-o)* plays the role of the error signal
- We use different notation: *t* for target or *d* for the desired value

Algorithm

1. iterations=0;
2. $\eta \in (0, 1]$;
3. Initialise all the weights $w_0, w_, \cdots, w_D$ to some random values;
4. Choose a pattern $\mathbf{x}_k$ out of the training set;
5. Compute $net_k = \sum_{i=1}^{D} w_j \cdot x_{k,j} + w_0 = \langle \mathbf{x}_k | \mathbf{w} \rangle + w_0 \cdot x_0$;
6. Compute the output by the activation function $o_k = sgn(net_k)$;
7. Compute $\Delta w_j = \eta \cdot (t_k - o_k) \cdot x_{k,j}$;
8. Update the weights $w_j = w_j + \Delta w_j$;
9. iterations++;
10. If no change in weights for all training set or maximum number of iteration THEN STOP ELSE GOTO 4;

$$\Delta w_j = \eta \cdot (t_k - o_k) \cdot x_j,$$

$$w_j^{new} = w_j^{old} + \Delta w_j,$$

where $\delta_k = (t_k - o_k)$ plays the role of the error signal with being either zero, 2 or -2

$$\delta_k = \begin{cases} 2 & \text{if } t_k = 1 \text{ and } o_k = -1 \\ -2 & \text{if } t_k = -1 \text{ and } o_k = 1 \\ 0 & \text{if } t_k = o_k \end{cases}$$

In this example a linearly separable training set is described by four vectors

$$\mathbf{x}_1 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \mathbf{x}_2 = \begin{pmatrix} 0 \\ 2 \end{pmatrix}, \mathbf{x}_3 = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \mathbf{x}_4 = \begin{pmatrix} 1 \\ -1 \end{pmatrix},$$

and the corresponding targets

$$t_1 = -1, t_2 = 1, t_3 = 1, t_4 = -1.$$

The weights are initialized to 1 and the learning rate η for simplicity is set to 1 as well

$$w_0 = 1, w_1 = 1, w_2 = 1, \quad \eta = 1$$

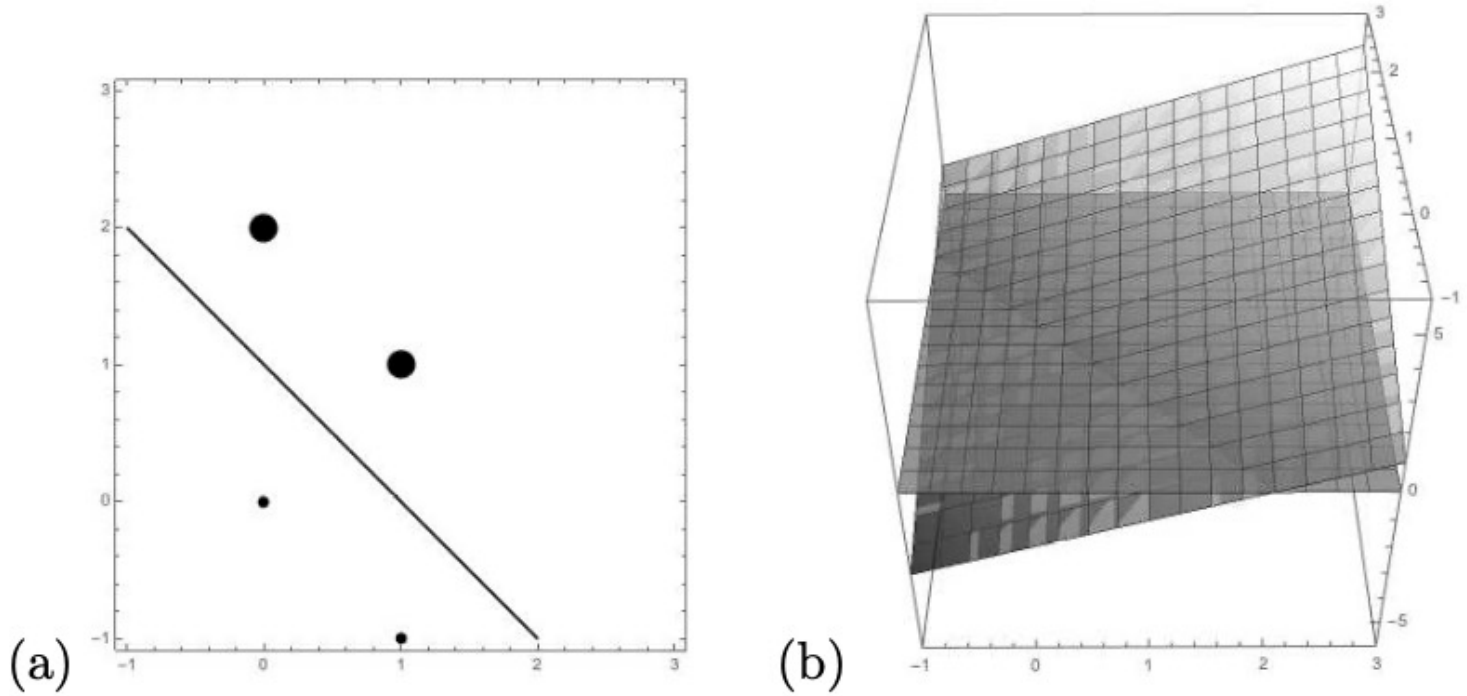
$$o_1 = sgn(1 \cdot 0 + 1 \cdot 0 + 1) = sgn(1) = 1; \quad \delta_1 = -2; \quad \mathbf{w} = \begin{pmatrix} -1 \\ 1 \\ 1 \end{pmatrix},$$

$$o_2 = sgn(1 \cdot 0 + 1 \cdot 2 - 1) = sgn(1) = 1; \quad \delta_2 = 0; \quad \mathbf{w} = \begin{pmatrix} -1 \\ 1 \\ 1 \end{pmatrix},$$

$$o_3 = sgn(1 \cdot 1 + 1 \cdot 1 - 1) = sgn(1) = 1; \quad \delta_3 = 0; \quad \mathbf{w} = \begin{pmatrix} -1 \\ 1 \\ 1 \end{pmatrix},$$

$$o_4 = sgn(1 \cdot 1 + 1 \cdot (-1) - 1) = sgn(-1) = -1; \quad \delta_4 = 0; \quad \mathbf{w} = \begin{pmatrix} -1 \\ 1 \\ 1 \end{pmatrix}$$

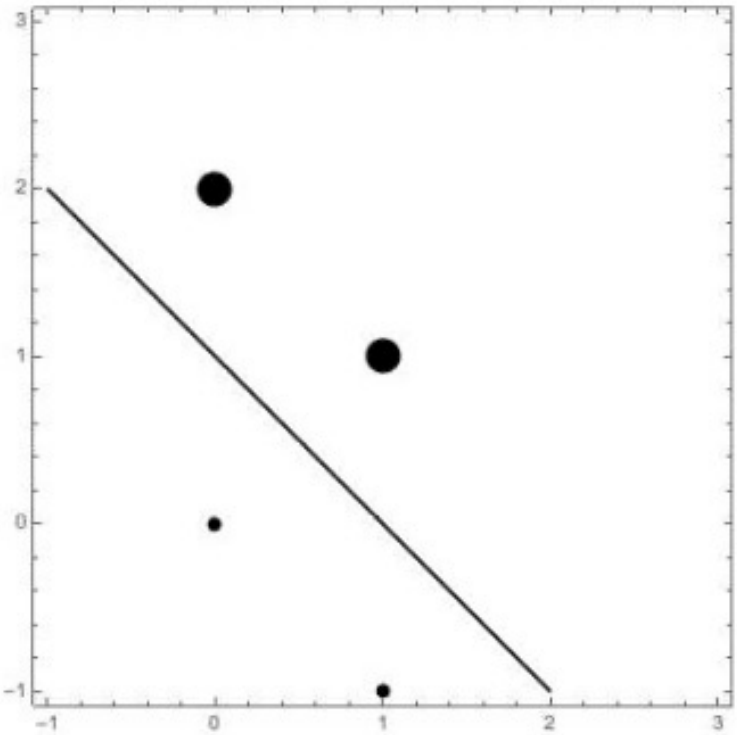
For additional epochs, the weights do not change



(a) The two classes 1 (indicated by a big point) and −1 (indicated a small point) are separated by the line $-1+x_1+x_2=0$. (b) The hyperplane $-1+x_1+x_2=y$ defines the line for $y = 0$.

Initializxation:

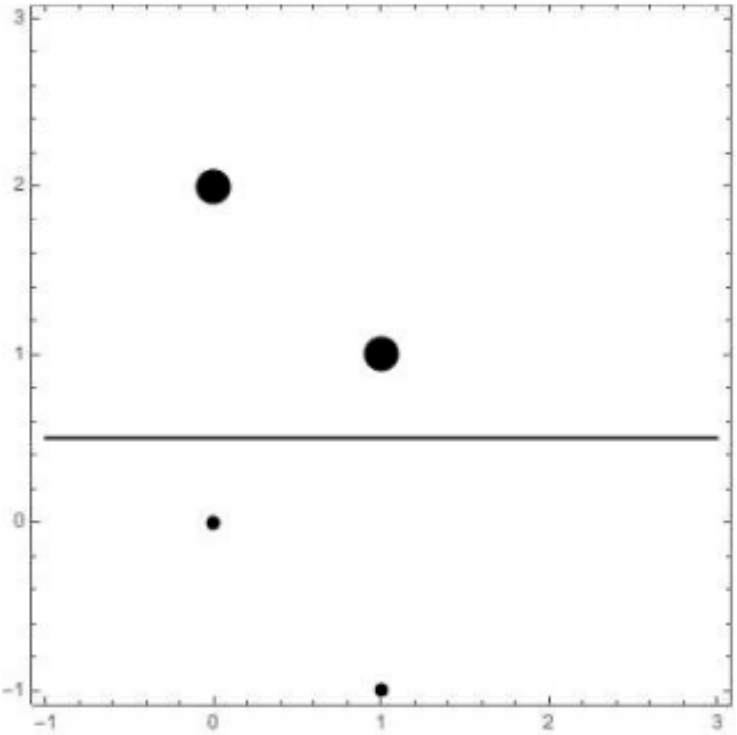
$$w_0 = 1, w_1 = 1, w_2 = 1, \quad \eta = 1$$



The hyperplane $-1+x_1+x_2 = y$ defines the line for y = 0 after the first epoch

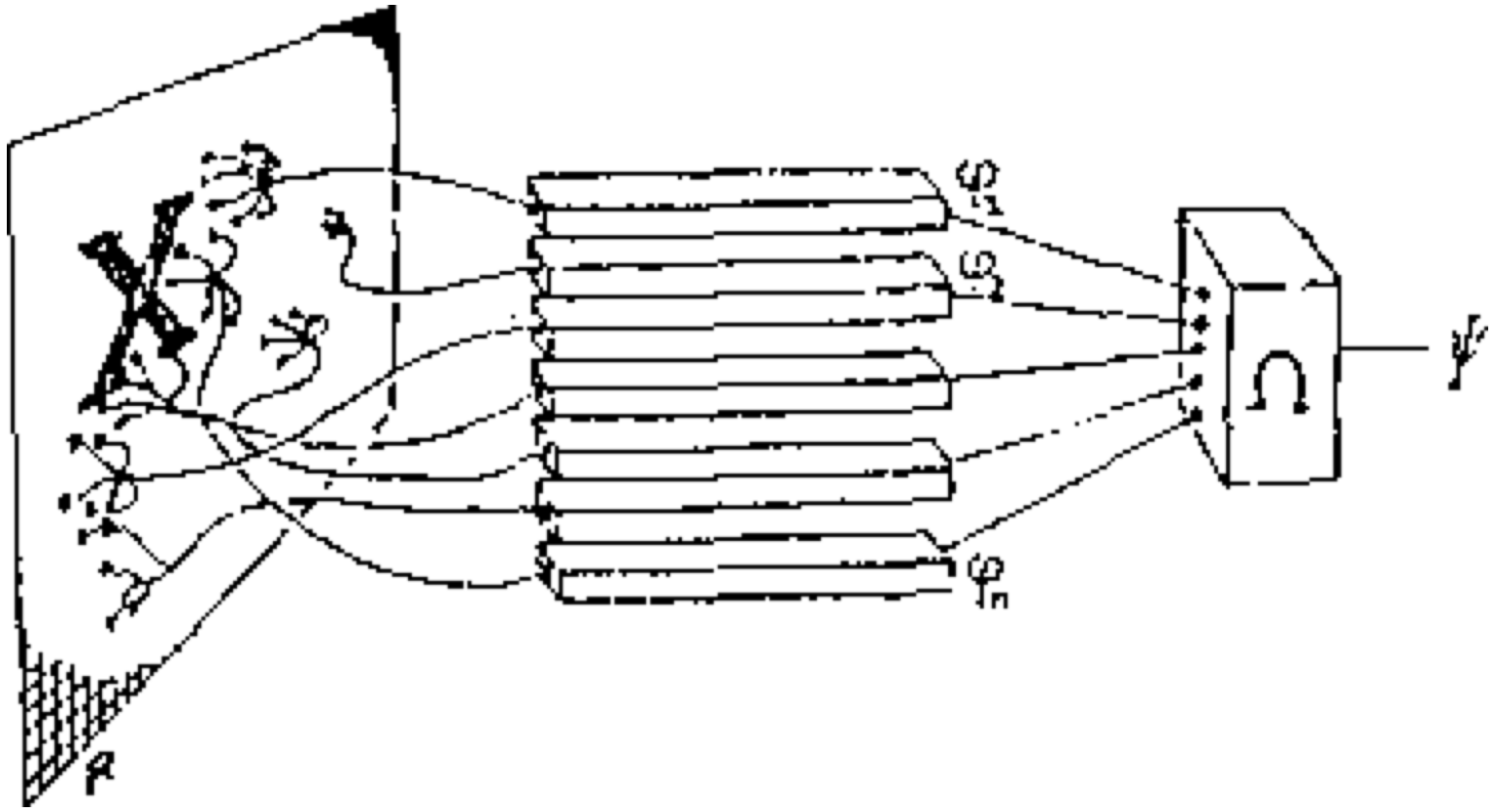
Initializxation:

$$w_0 = 0, w_1 = 0, w_2 = 0, \quad \eta = 1$$



The two classes 1 (indicated by a big point) and −1 (indicated a small point) are separated by the line $-2+0\cdot x_1+4\cdot x_2 = 0$ after the second epoch

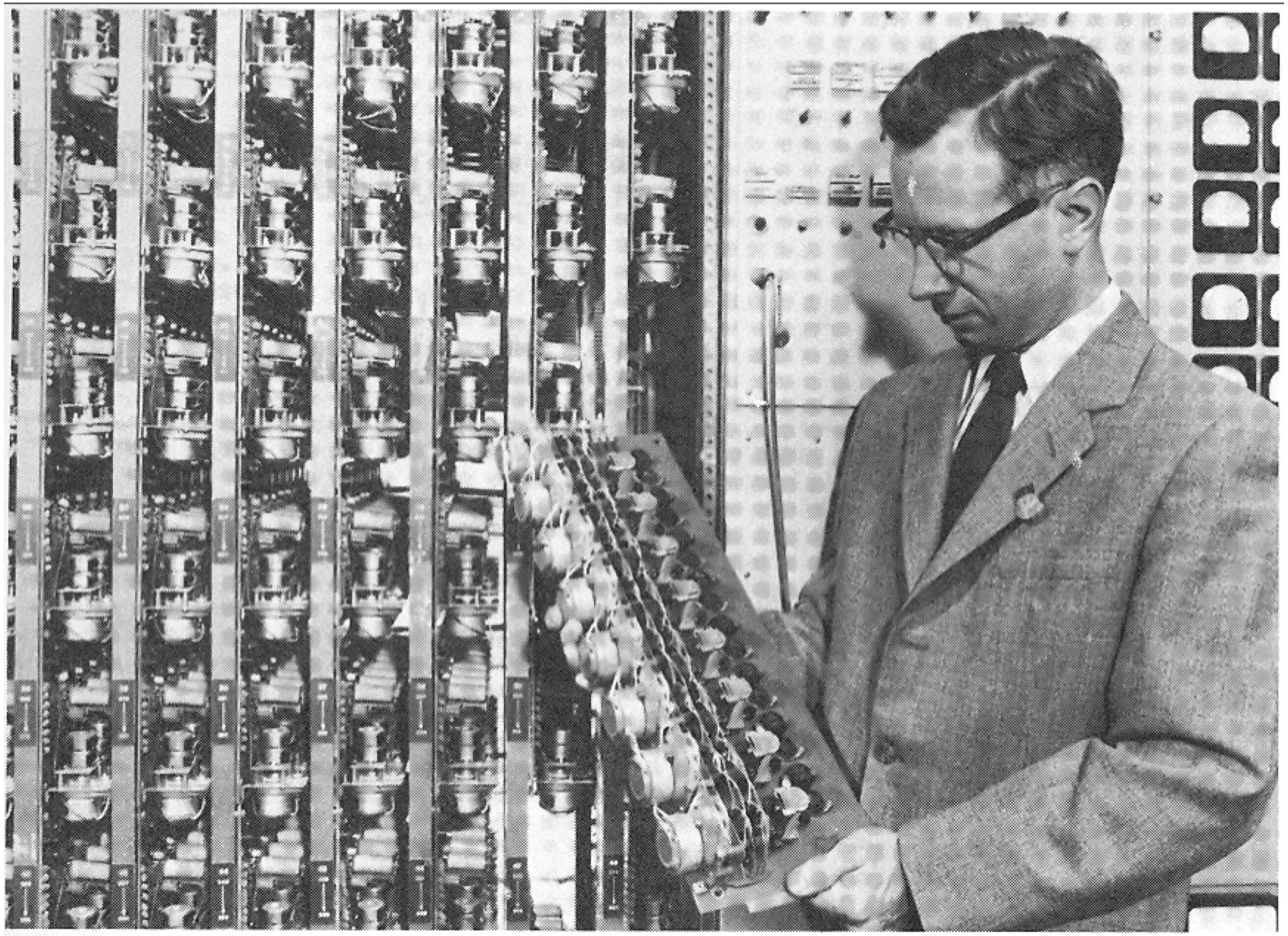
Constructions

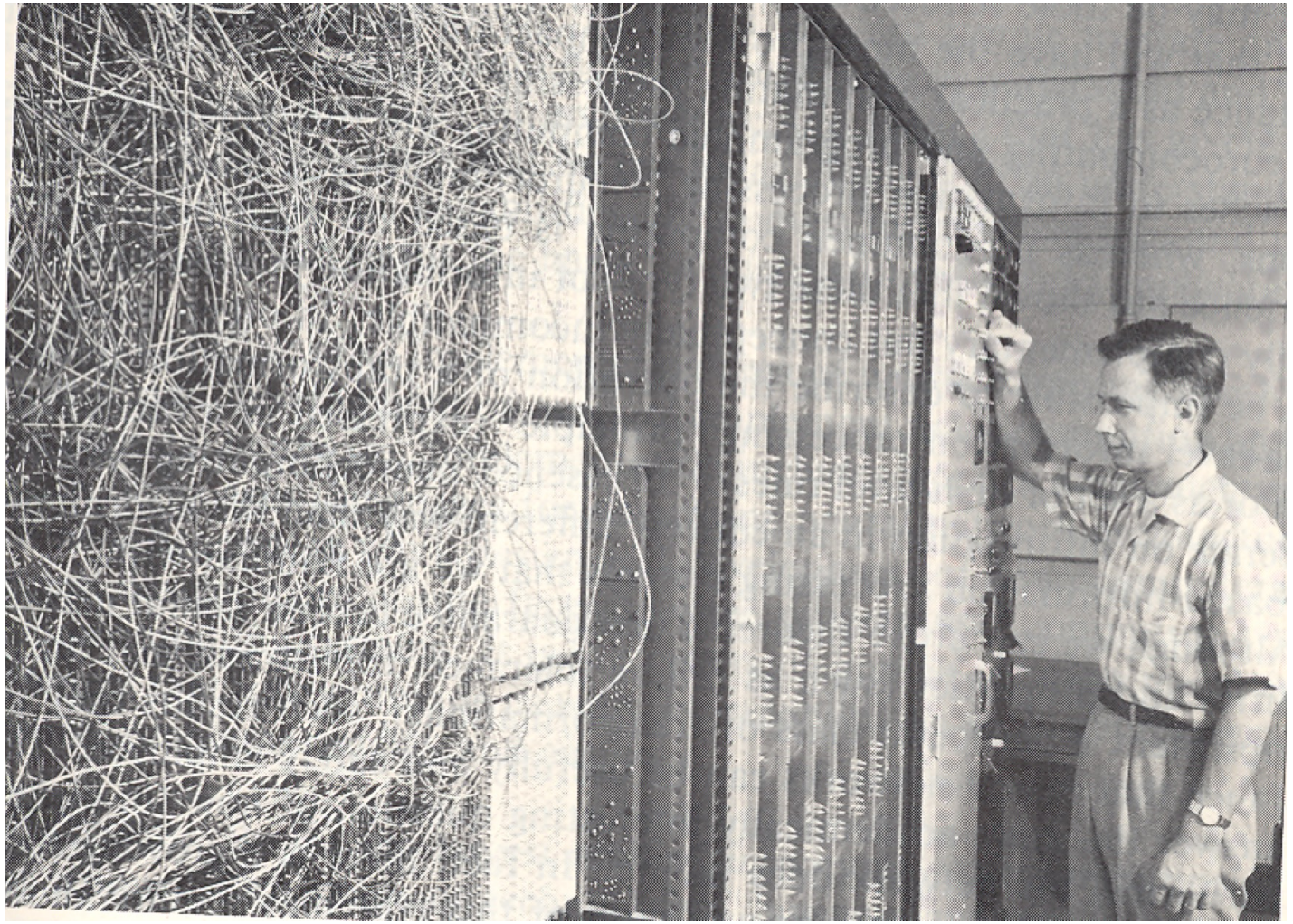


Frank Rosenblatt

Frank Rosenblatt (1928-1971) was an American psychologist notable for the invention of the perceptron. He is the father of machine learning, one cannot stress enough the importance of his model.

C





Psychological Review
Vol. 65, No. 6, 1958

THE PERCEPTRON: A PROBABILISTIC MODEL FOR INFORMATION STORAGE AND ORGANIZATION IN THE BRAIN[1]

F. ROSENBLATT

Cornell Aeronautical Laboratory

If we are eventually to understand the capability of higher organisms for perceptual recognition, generalization, recall, and thinking, we must first have answers to three fundamental questions:

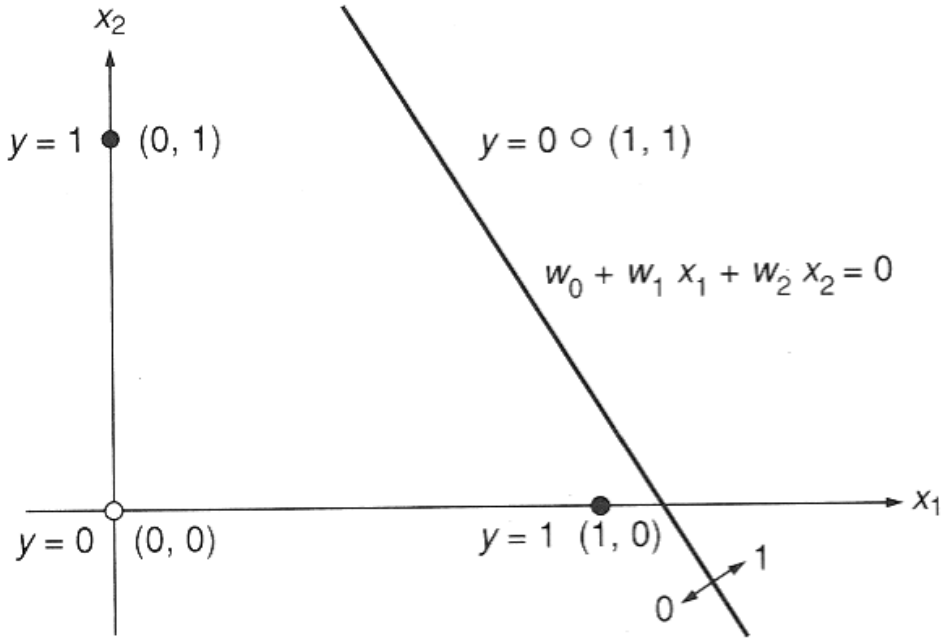
1. How is information about the physical world sensed, or detected, by the biological system?
2. In what form is information stored, or remembered?
3. How does information contained in storage, or in memory, influence recognition and behavior?

and the stored pattern. According to this hypothesis, if one understood the code or "wiring diagram" of the nervous system, one should, in principle, be able to discover exactly what an organism remembers by reconstructing the original sensory patterns from the "memory traces" which they have left, much as we might develop a photographic negative, or translate the pattern of electrical charges in the "memory" of a digital computer. This hypothesis is appealing in its simplicity and ready intelligibility, and a large family of theoretical brain

- Rosenblatt's bitter rival and professional nemesis was Marvin Minsky of Carnegie Mellon University
- Minsky despised Rosenblatt, hated the concept of the perceptron, and wrote several polemics against him
- For years Minsky crusaded against Rosenblatt on a very nasty and personal level, including contacting every group who funded Rosenblatt's research to denounce him as a charlatan, hoping to ruin Rosenblatt professionally and to cut off all funding for his research in neural nets

XOR problem and Perceptron

- By Minsky and Papert in mid 1960



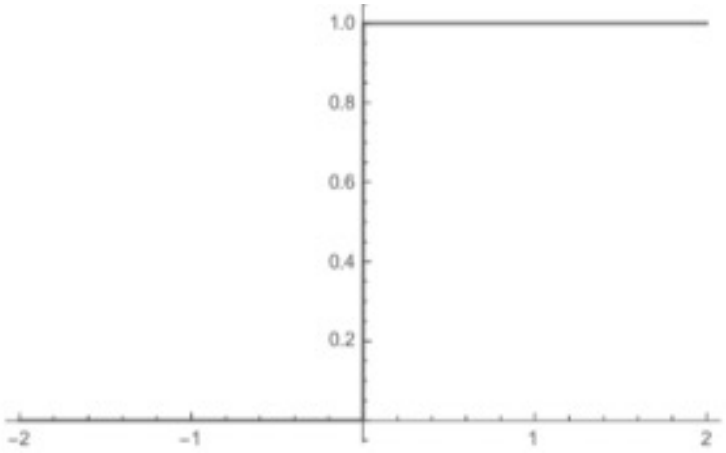
- The *sgn* activation function can be scaled to the two values 0 or 1 for not firing and firing,

$$\phi(net) := sgn_0(net) = \begin{cases} 1 & \text{if } net \geq 0 \\ 0 & \text{if } net < 0 \end{cases}$$

Discriminant Functions

$$f(\mathbf{x}) = \begin{cases} 1 & \text{if} \quad \langle \mathbf{x} | \mathbf{w} \rangle + b \geq 0 \\ 0 & \text{otherwise,} \end{cases}$$

$$b := w_0$$



$\delta_k = (t_k - o_k)$ plays the role of the error signal with being either zero, ***1*** **or** ***–1***

$$\delta_k = \begin{cases} 2 & \text{if } t_k = 1 \text{ and } o_k = -1 \\ -2 & \text{if } t_k = -1 \text{ and } o_k = 1 \end{cases}$$

Linear Regression and Linear Artificial Neuron

- Imagine we have no activation function

$$y = y(\mathbf{x}, \mathbf{w}) = w_0 + \sum_{j=1}^{D} w_j \cdot x_j = w_0 + \langle \mathbf{w} | \mathbf{x} \rangle$$

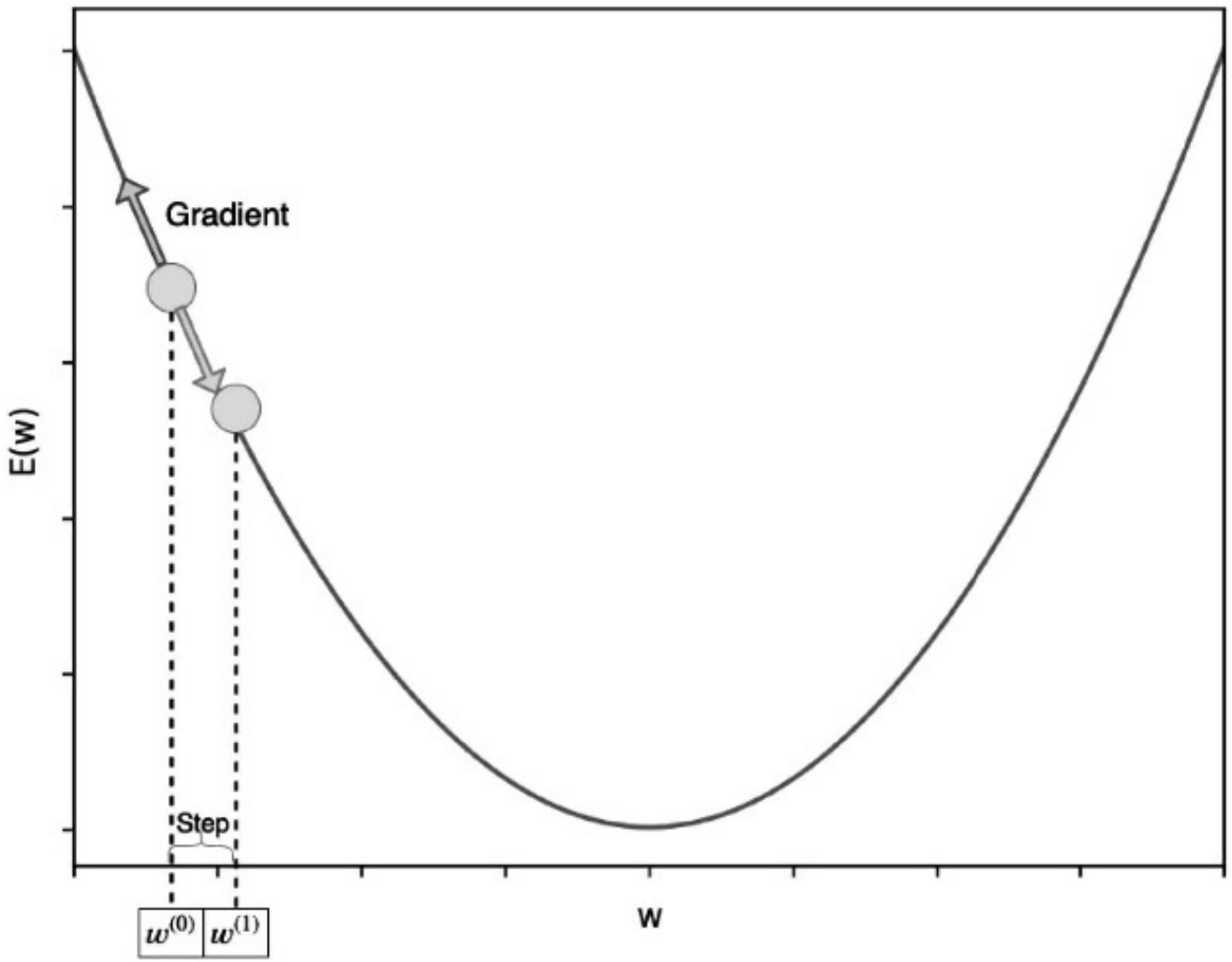
$$o_k = net_k = bias + \sum_{j=1}^{D} w_j \cdot x_{k,j} = w_0 + \sum_{j=1}^{D} w_j \cdot x_{k,j}$$

The training error for a training data set of N elements with inputs $\mathbf{x}_1, \cdots, \mathbf{x}_k, \cdots, \mathbf{x}_N$ and targets $t_1, \cdots, t_k, \cdots, t_N$ is given by

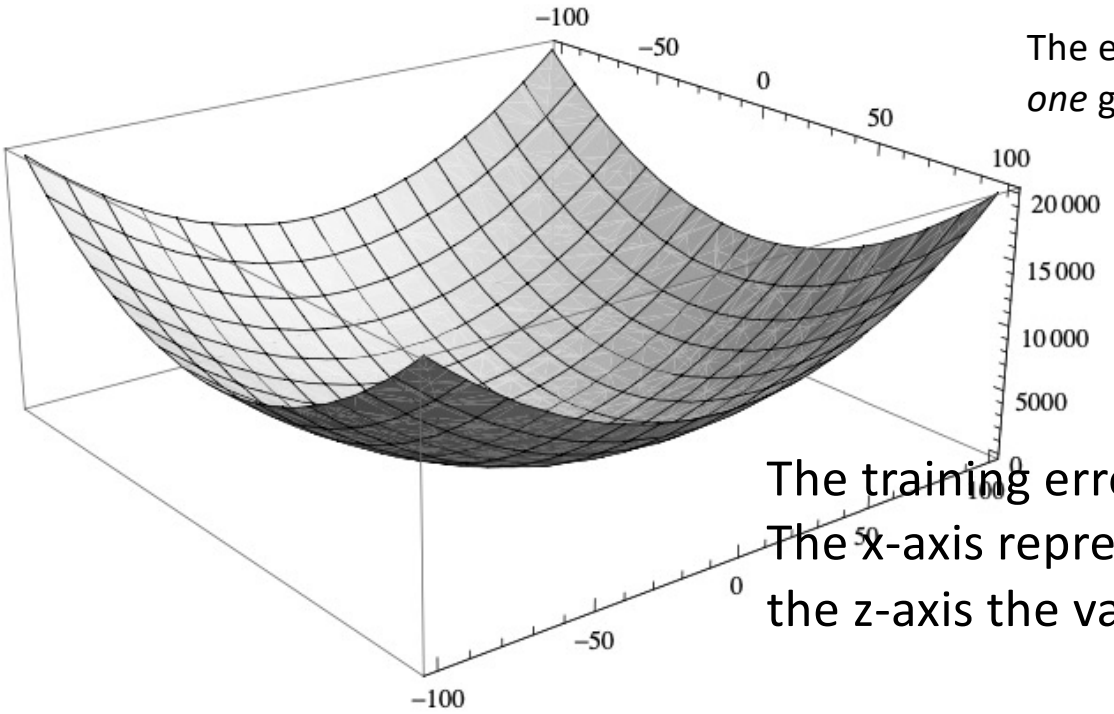
$$E(\mathbf{w}) = \frac{1}{2} \cdot \sum_{k=1}^{N} (t_k - o_k)^2 = \frac{1}{2} \cdot \sum_{k=1}^{N} \left(t_k - \sum_{j=0}^{D} w_j \cdot x_{k,j} \right)^2$$

We have seen that a closed-form solution that minimizes E(w) for w exists with

$$\nabla E(\mathbf{w}) = 0$$


Gradient
E(w)
Step
$w^{(0)}$
$w^{(1)}$
w

- We can also get a **non closed-form solution** using gradient descent.



The error surface as defined by E(w) contains only *one* global minimum since it is convex

The training error is a function of w, with w = (w_0,w_1). The x-axis represents w_0, the y-axis represents w_1 and the z-axis the value E(w) = E((w_0, w_1))

- To find a local minimum of a function *E(**w**)* using gradient descent, one takes steps proportional to the negative of the gradient

$$E(\mathbf{w}) = \frac{1}{2} \cdot \sum_{k=1}^{N} (t_k - o_k)^2 = \frac{1}{2} \cdot \sum_{k=1}^{N} \left(t_k - \sum_{j=0}^{D} w_j \cdot x_{k,j} \right)^2$$

$$-\nabla E(\mathbf{w}) = -\left(\frac{\partial E}{\partial w_0}, \frac{\partial E}{\partial w_1}, \cdots, \frac{\partial E}{\partial w_n} \right)^T$$

$$\mathbf{w}^{new} = \mathbf{w}^{old} + \Delta\mathbf{w}$$

$$w_j^{new} = w_j^{old} + \Delta w_j$$

$$\Delta\mathbf{w} = \eta \cdot (-\nabla E(\mathbf{w}))$$

$$\Delta w_j = -\eta \cdot \frac{\partial E}{\partial w_j}.$$

$$\frac{\partial E}{\partial w_j} = \frac{\partial}{\partial w_j}\frac{1}{2}\cdot\sum_{k=1}^{N}(t_k - o_k)^2 = \frac{1}{2}\cdot\sum_{k=1}^{N}\frac{\partial}{\partial w_j}(t_k - o_k)^2$$

$$\frac{\partial E}{\partial w_j} = \frac{1}{2}\cdot\sum_{k=1}^{N} 2\cdot(t_k - o_k)\cdot\frac{\partial}{\partial w_j}(t_k - o_k)$$

$$\frac{\partial E}{\partial w_j} = \sum_{k=1}^{N}(t_k - o_k)\cdot\frac{\partial}{\partial w_j}\left(t_k - \sum_{j=0}^{D} w_j\cdot x_{k,j}\right)$$

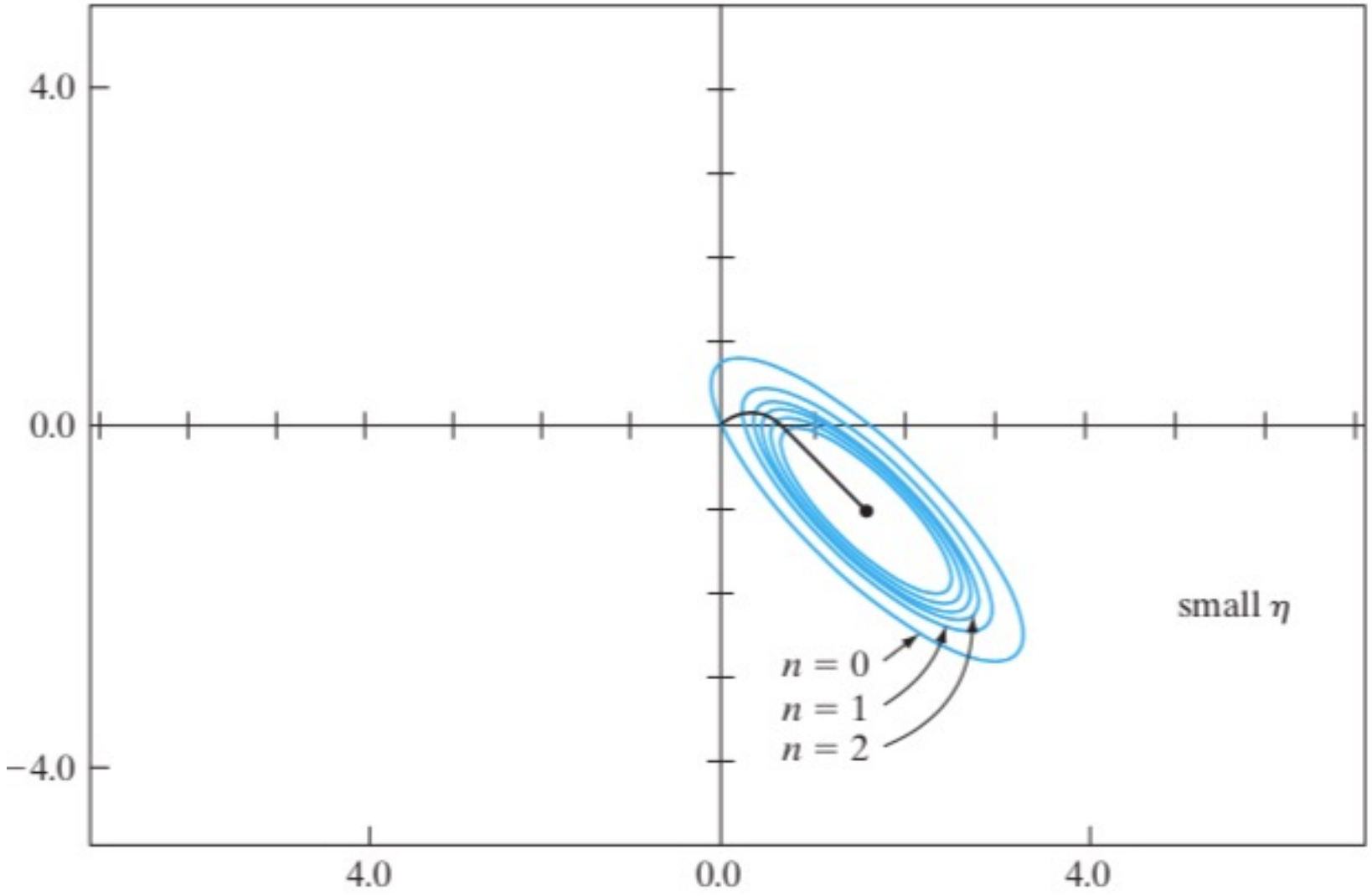
$$\frac{\partial E}{\partial w_j} = \sum_{k=1}^{N}(t_k - o_k)\cdot(-x_{k,j})$$

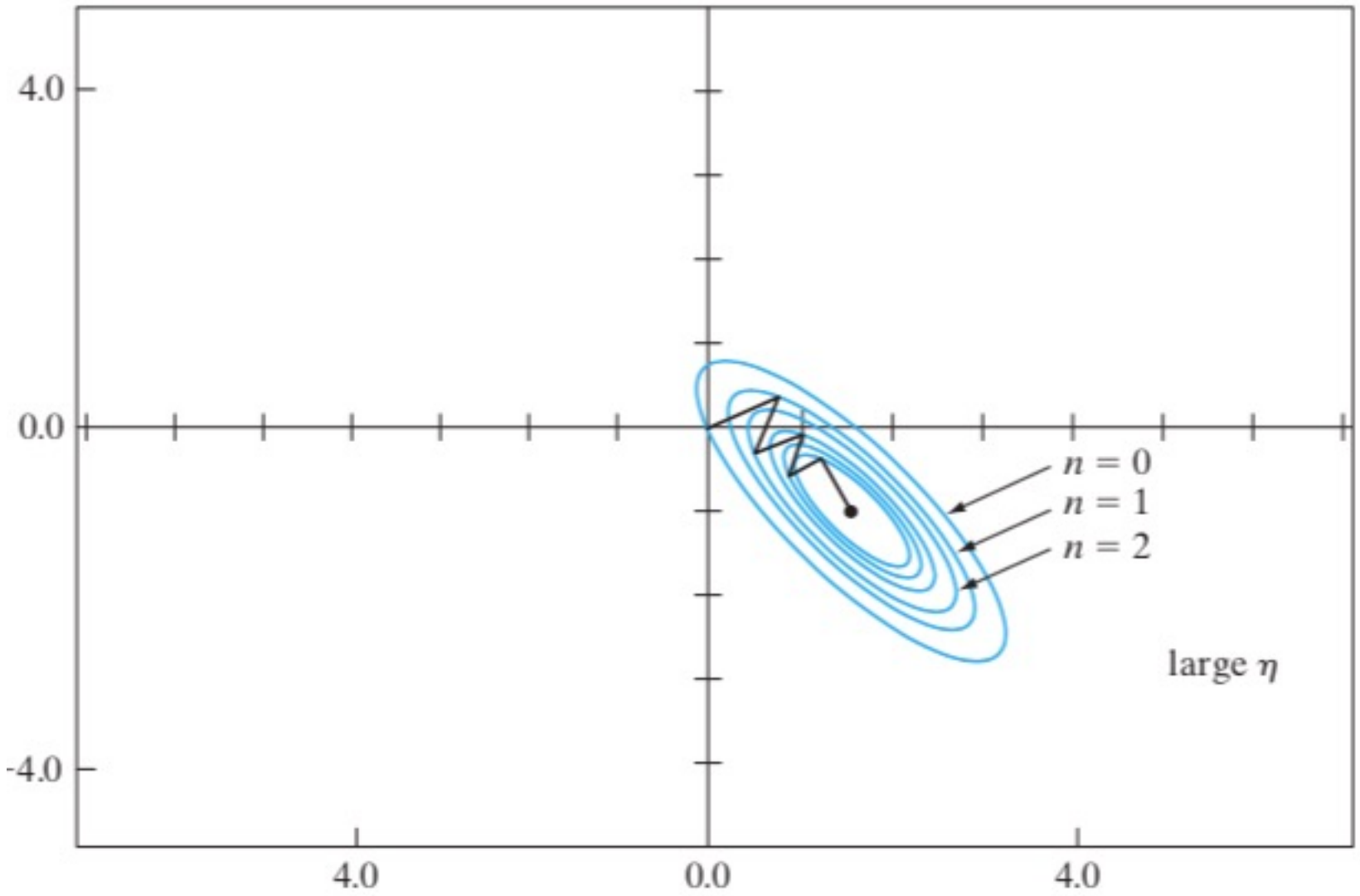
$$\frac{\partial E}{\partial w_j} = -\sum_{k=1}^{N}(t_k - o_k)\cdot x_{k,j}.$$

- The update rule for gradient decent is given by

$$\Delta w_j = \eta \cdot \sum_{k=1}^{N} (t_k - o_k) \cdot x_{k,j}.$$

$$w_j^{new} = w_j^{old} + \Delta w_j.$$

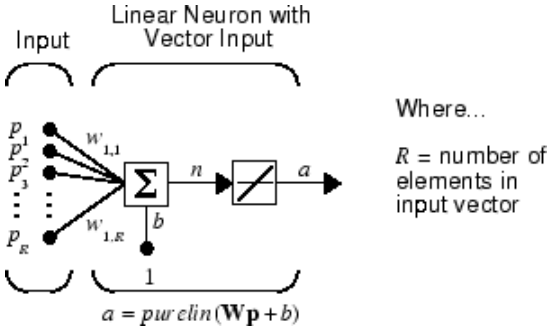

4.0
0.0
−4.0
4.0
0.0
4.0
small η
n = 0
n = 1
n = 2


4.0
0.0
-4.0
4.0
0.0
4.0
n = 0
n = 1
n = 2
large η

Stochastic gradient descent

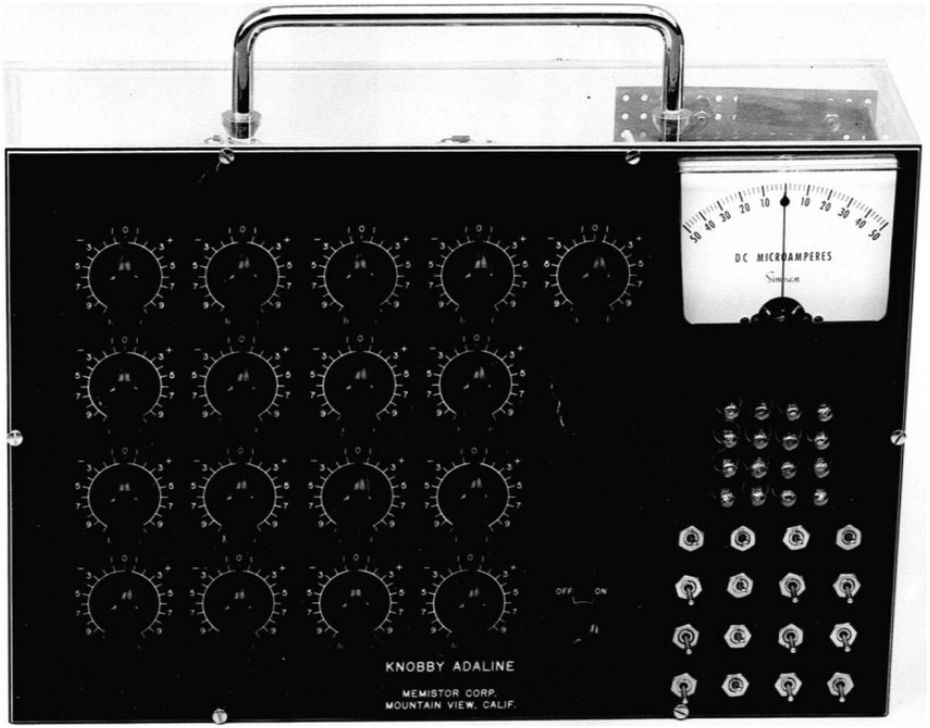
- The gradient decent training rule updates summing over all N training examples.
- Stochastic gradient descent (SGD) approximates gradient decent by updating weights incrementally and calculating the error for each example according to the update rule by

$$\Delta w_j = \eta \cdot (t_k - o_k) \cdot x_{k,j}$$



$$\Delta w_j = \eta \cdot (t_k - o_k) \cdot x_{k,j}$$

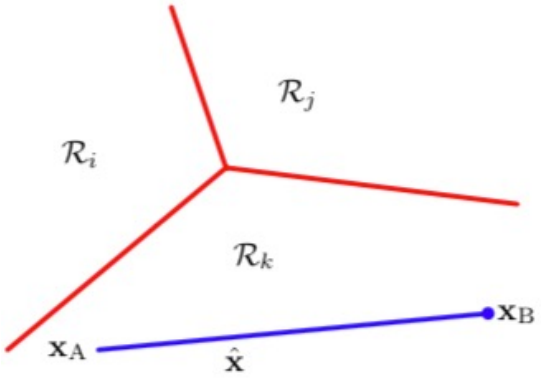
- This rule is known as delta-rule or LMS (last mean-square) weight update.
- It is used in Adaline (Adaptive Linear Element) for adaptive filters and was developed by Widroff and Hoff



- ADALINE. An adaptive linear neuron. Manually adapted synapses. Designed and built by Ted Hoff in 1960. Demonstrates the least mean square LMS learning algorithm. (Courtesy of Professor Bernard Widrow.)

Multiclass linear discriminant

- For K artificial neurons an index k is used with $k \in \{1,2,\cdots,K\}$
- Identify the weight vector w_k and the output o_k of the neuron.



with the prediction

$$o_k = net_k = \sum_{j=1}^{D} w_{k,j} \cdot x_j$$

$$\arg\max_{\kappa} \left(\mathbf{w}_\kappa^T \cdot \mathbf{x}\right)$$

REGULARISATION

With l_2 regularisation we get (see linear regression, for simplicity without 1/2)

$$\frac{\partial E}{\partial w_j} = -\sum_{k=1}^{N}(t_k - o_k) \cdot x_{k,j} + \lambda \cdot w_j.$$

and the stochastic gradient descent (SGD) rule is

$$\Delta w_j = \eta \cdot (t_k - o_k) \cdot x_{k,j} - \eta \cdot \lambda \cdot w_j$$

With l_1 regularisation we get (see linear regression), there is no derivative for $|w_j|$, we use a subderivative with $sign$ function

$$\frac{\partial E}{\partial w_j} = -\sum_{k=1}^{N}(t_k - o_k) \cdot x_{k,j} + \lambda \cdot sign(w_j)$$

and the stochastic gradient descent (SGD) rule is

$$\Delta w_j = \eta \cdot (t_k - o_k) \cdot x_{k,j} - \eta \cdot \lambda \cdot sign(w_j)$$

sign

- $|w_j|$ is convex
  - The subdifferential at the origin is the interval *[−1, 1].*
  - The subdifferential at any point $x_0<0$ is the singleton set *{−1}*
  - The subdifferential at any point $x_0>0$ is the singleton set *{1}*.
  - This is similar to the sign function, but is not a single-valued function at *0*
    - instead including all possible subderivatives.

$$sign(w_j) = \begin{cases} -1 & if \quad w_j < 0 \\ 0 & if \quad w_j = 0 \\ 1 & if \quad w_j > 0 \end{cases}$$

Continuous activation functions

For continuous activation function $\phi()$

$$o_k = \phi\left(\sum_{j=0}^{D} w_j \cdot x_{k,j}\right).$$

we can define as well the update rule for gradient decent with the differential

$$\frac{\partial E}{\partial w_j} = \sum_{k=1}^{N}(t_k - o_k) \cdot \frac{\partial}{\partial w_j}\left(t_k - \phi\left(\sum_{j=0}^{D} w_j \cdot x_{k,j}\right)\right)$$

$$\frac{\partial E}{\partial w_j} = \sum_{k=1}^{N}(t_k - o_k) \cdot \left(-\phi'\left(\sum_{j=0}^{D} w_j \cdot x_{k,j}\right) \cdot x_{k,j}\right)$$

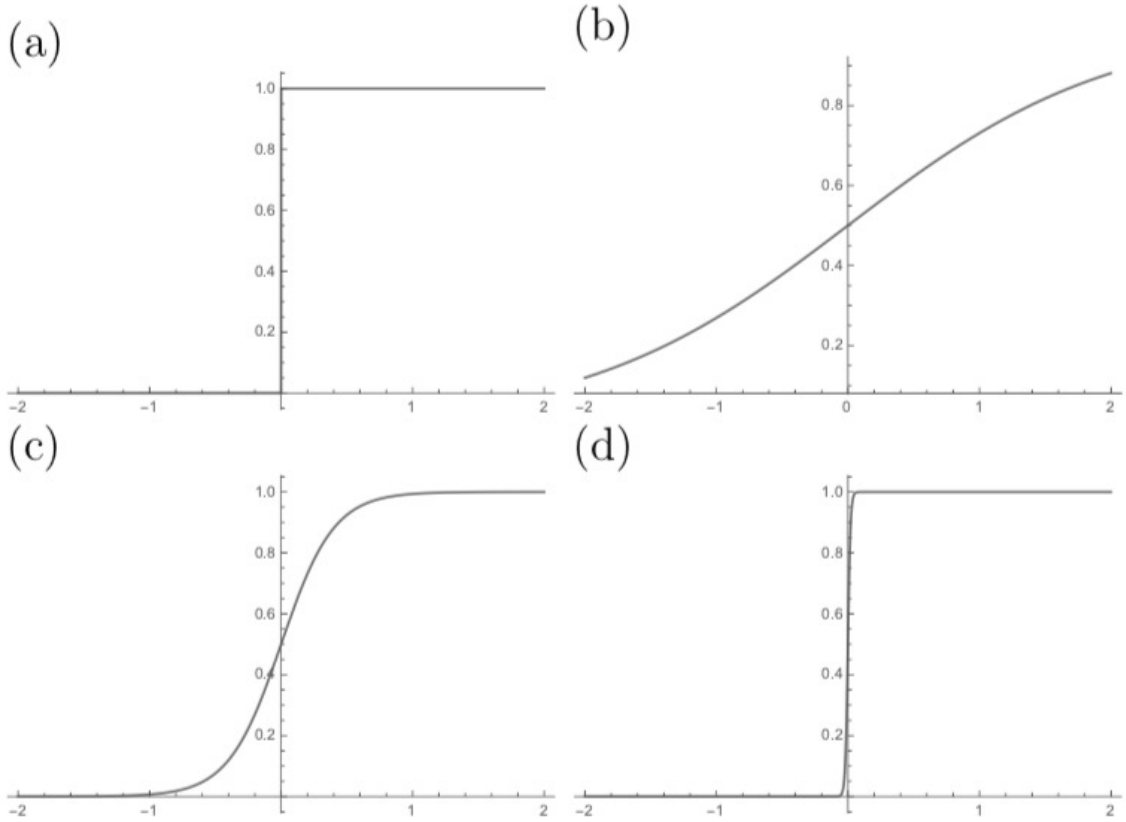
$$\frac{\partial E}{\partial w_j} = -\sum_{k=1}^{N}(t_k - o_k) \cdot \phi'\left(\sum_{j=0}^{D} w_j \cdot x_{k,j}\right) x_{k,j}.$$

- The *sgn* activation function can be scaled to the two values 0 or 1 for not firing and firing,

$$\phi(net) := sgn_0(net) = \begin{cases} 1 & \text{if } net \geq 0 \\ 0 & \text{if } net < 0 \end{cases}$$

- Both non linear function *sgn(net)* and *sgn0(net)* are non continuous. The activation function $sgn_0(net)$ can be approximated by the non linear continuous function $\sigma(net)$

$$\phi(net) := \sigma(net) = \frac{1}{1 + e^{(-\alpha \cdot net)}}$$



(a) The activation function *sgn0(net).* (b) The function σ(net) with *α = 1*. (c) The function σ*(net) with α = 5*. (d) The function σ(net) with *α = 10* is very similar to *sgn0(net),* bigger *α* make it even more similar

Sigmoid Activation

For the non linear continuous activation function $\sigma()$

$$o_k = \sigma\left(\sum_{j=0}^{N} w_j \cdot x_{k,j}\right)$$

we can define as well the update rule for gradient decent with the differential

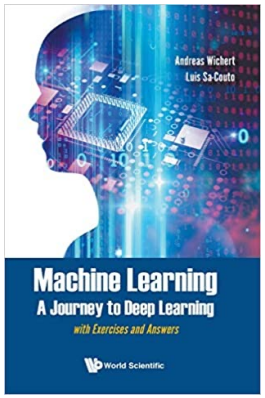
$$\sigma(x)' = \alpha \cdot \sigma(x) \cdot (1 - \sigma(x))$$

we get

$$\frac{\partial E}{\partial w_j} = -\alpha \cdot \sum_{k=1}^{N} (t_k - o_k) \cdot \sigma\left(net_{k,j}\right) \cdot \left(1 - \sigma\left(net_{k,j}\right)\right) \cdot x_{k,j}.$$

what does this mean?

Literature



- Machine Learning - A Journey to Deep Learning, A. Wichert, Luis Sa-Couto, World Scientific, 2021
  - Chapter 1 and Chapter 5