Neural networks

Doc. RNDr. Iveta Mrázová, CSc.

Department of Theoretical Computer Science and Mathematical Logic

Faculty of Mathematics and Physics

Charles University in Prague

Neural networks

– Perceptron and linear separability –

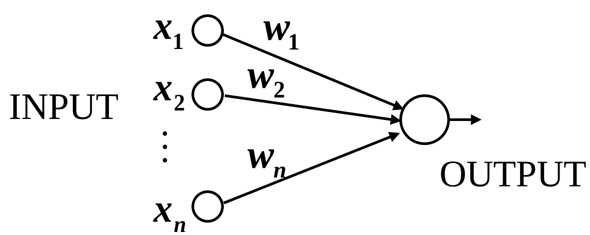
Doc. RNDr. Iveta Mrázová, CSc.

Department of Theoretical Computer Science
and Mathematical Logic

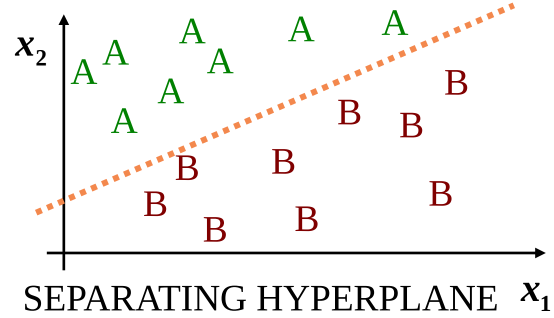
Faculty of Mathematics and Physics

Charles University in Prague

Formal neuron



$$y = f_h\left(\sum_{i=1}^{n} w_i\, x_i + \vartheta\right)$$



$$x_2 = -\frac{w_1}{w_2}x_1 - \frac{\vartheta}{w_2}$$

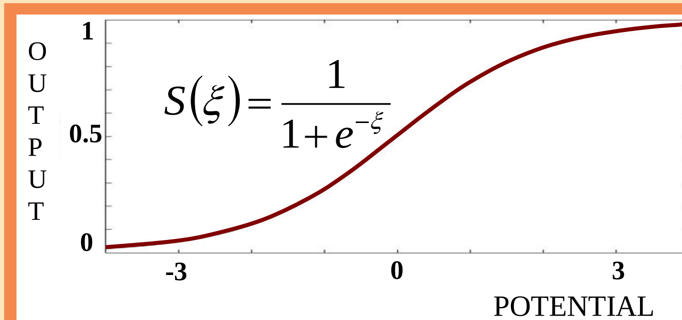
$$y = \begin{cases} 1 & \text{if } \sum_{i=1}^{n} w_i x_i + \vartheta \geq 0: \ \text{CLASS A} \\ 0 & \text{if } \sum_{i=1}^{n} w_i x_i + \vartheta < 0: \ \text{CLASS B} \end{cases}$$

Types of transfer functions

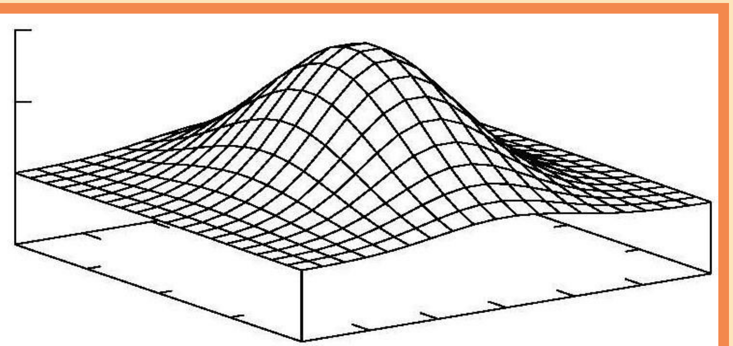
Hard-limiting

$$y = \begin{cases} \mathbf{1} & \boldsymbol{if} \sum_{i=1}^{n} w_i x_i + \vartheta \geq 0: \ \text{CLASS A} \\ \mathbf{0} & \boldsymbol{if} \sum_{i=1}^{n} w_i x_i + \vartheta < 0: \ \text{CLASS B} \end{cases}$$

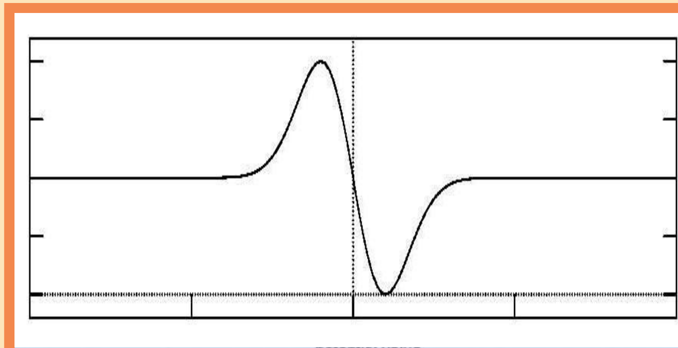
Sigmoidal



Radial basis (RBF)



Wavelet



Definition of a formal neuronu

A neuron with the weights $(w_1, \ldots, w_n) \in R^n$, the threshold $\vartheta \in R$ and the transfer function $f: R^n \times R^n \rightarrow R$ computes for any input $\mathbf{z} \in R^n$ its output $\mathbf{y} \in R$ as the value of the transfer function in $\mathbf{z}$, $f[\mathbf{w}, \vartheta](\mathbf{z})$.

Most often, the so-called sigmoidal transfer function is considered with the values bound by 0 and 1:

$$y = f[\mathbf{w}, \vartheta](\mathbf{z}) = f(\xi) = \frac{1}{1 + e^{-\lambda\xi}}$$

$\xi = \sum_{i=1}^{n} z_i w_i + \vartheta$ denotes the so-called neuron potential, R is the set of real numbers

Definition of neuron states

Let $f[\mathbf{w}, \vartheta](\mathbf{z})$ denotes the input of a neuron:

- when $f[\mathbf{w}, \vartheta](\mathbf{z}) = 1$, we say that the neuron is active;
- when $f[\mathbf{w}, \vartheta](\mathbf{z}) = ½$, we say that the neuron is silent;

  This fact indicates that the respective input is located on the separating hyperplace given by this neuron.
- when $f[\mathbf{w}, \vartheta](\mathbf{z}) = 0$, we say that the neuron is passive.

Training and recall

- **Training:**
  - **Supervised – training set** of the form [input / desired output]
  - **Self-organization** – no desired output

  ⇒ **Goal: setting (adaptation) of the synaptic weights**

  (e.g., through minimalization of the mean squared error)

  Objective function: e.g., $$\sum_p \sum_j \left(y_{p,j} - d_{p,j}\right)^2$$

  $\boldsymbol{y}$ is the actual and $\boldsymbol{d}$ is the desired output

- **Recall:**
  - **of newly presented input patterns**

  => **Goal: get the response (output) of the neural network**

Definition of training patterns

For a BP-network B with n input and m output neurons:

- An input pattern is an input vector $\mathbf{x} \in R^n$ being processed by B.
- An output pattern $\mathbf{d} = (d_1, \ldots, d_m)$ is formed by desired outputs of neurons lying in the output layer.
- An actual output of B is a vector $\mathbf{y} = (y_1, \ldots, y_m)$ formed by actual outputs of neurons lying in the outpt layer.

A training set T is a finite non-empty set of P ordered pairs of input / output patterns:

$$T = \{[x_1, d_1], \ldots, [x_{1P}, d_{1P}]\}.$$

Perceptron and linear separability (1)

D: **A simple perceptron** is a computing unit with the threshold ϑ which, when receiving the n real inputs $x_1, x_2, \ldots, x_n$ through edges with the associated weights $w_1, \ldots, w_n$ yields the output 1, if the following inequality holds:

$$\sum_{i=1}^{n} w_i \, x_i \geq \vartheta \quad \text{(i.e., if } \vec{w} \cdot \vec{x} \geq \vartheta \text{) and } 0 \text{ otherwise.}$$

Note: Similarly for the so-called **extended weight and input vector**:

$$\vec{w} = (w_1, w_2, \ldots, w_n, w_{n+1}) \quad ; \quad w_{n+1} = -\vartheta$$

$$\vec{x} = (x_1, x_2, \ldots, x_n, 1)$$

=> output 1, if $\vec{w} \cdot \vec{x} \geq 0$

Perceptron and linear separability (2)

Linear separability:

D: Two sets of points $\boldsymbol{A}$ and $\boldsymbol{B}$ are called **linearly separable** in an $\boldsymbol{n}$-dimensional space, if $\boldsymbol{n+1}$ real numbers $\boldsymbol{w_1, \ldots, w_n, \vartheta}$ exist, such that every point $(x_1, x_2, \ldots, x_n) \in A$ satisfies $\sum_{i=1}^{n} w_i x_i \geq \vartheta$ and every point $(x_1, x_2, \ldots, x_n) \in B$ satisfies $\sum_{i=1}^{n} w_i x_i < \vartheta$

Perceptron and linear separability (3)

Example:

- ***n=2*** => 14 out of 16 possible Boolean functions are „linearly separable“
- ***n=3*** => 104 z 256 - ´´ -
- ***n=4*** => 1882 z 65536 - ´´ -
- For a general case ***n***, there is still no known formula expressing the number of linearly separable functions

Perceptron and linear separability (4)

Absolute linear separability:

D: Two sets ***A*** and ***B*** are called **absolutely linearly separable** in an ***n***-dimensional space, if ***n+1*** real numbers $w_1, \ldots, w_n, \vartheta$ exist, such that every point $(x_1, x_2, \ldots, x_n) \in A$ satisfies $\sum_{i=1}^{n} w_i x_i > \vartheta$ and every point $(x_1, x_2, \ldots, x_n) \in B$ satisfies $\sum_{i=1}^{n} w_i x_i < \vartheta$

Perceptron and linear separability (5)

T: Two finite sets of points A and B, that are linearly separable in an *n*-dimensional space, are also absolutely linearly separable.

Proof: Since the two sets, A and B are linearly separable, real numbers $w_1, \dots, w_n, \vartheta$ exist, such that it holds

$$\sum_{i=1}^{n} w_i x_i \geq \vartheta \text{ for all points } (x_1, x_2, \dots, x_n) \in A$$

and $$\sum_{i=1}^{n} w_i x_i < \vartheta \text{ for all points } (x_1, x_2, \dots, x_n) \in B$$

Perceptron and linear separability (6)

Further let: $\varepsilon = \max\left\{\sum_{i=1}^{n} w_i b_i - \vartheta; (b_1, \dots, b_n) \in B\right\}$

clearly $\varepsilon < \varepsilon/2 < 0$

Let $\vartheta' = \vartheta + \frac{\varepsilon}{2}$ (therefore: $\vartheta = \vartheta' - \frac{\varepsilon}{2}$)

=> For all points in ***A*** it holds that $\sum_{i=1}^{n} w_i a_i - \left(\vartheta' - \frac{1}{2}\varepsilon\right) \geq 0$

This means that $\sum_{i=1}^{n} w_i a_i - \vartheta' \geq -\frac{1}{2}\varepsilon > 0$

$$\Rightarrow \quad \sum_{i=1}^{n} w_i a_i > \vartheta' \qquad (\forall (a_1, \dots, a_n) \in A) \qquad (\,*\,)$$

Perceptron and linear separability (6)

Similarly for all points in ***B***

$$\sum_{i=1}^{n} w_i b_i - \vartheta = \sum_{i=1}^{n} w_i b_i - \left(\vartheta' - \frac{1}{2}\varepsilon \right) \le \varepsilon$$

and therefore $$\sum_{i=1}^{n} w_i b_i - \vartheta' \le \frac{1}{2}\varepsilon < 0 \qquad (**)$$

From (*) and (**) it follows that the sets A and B are absolutely linearly separable.

QED

Separating hyperplane – for the extended weight and feature space (1)

D: The open (closed) positive half-space associated with the ***n*** – dimensional weight vextor $\vec{w}$ is the set of all points $\vec{x} \in R^n$ for which $\vec{w} \cdot \vec{x} > 0 \quad (\vec{w} \cdot \vec{x} \geq 0)$

The open (closed) negative half-space associated with is the set of all points $\vec{x} \in R^n$ for which $\vec{w} \cdot \vec{x} < 0 \quad (\vec{w} \cdot \vec{x} \leq 0)$

Separating hyperplane – for the extended weight and feature space (2)

D: The separating hyperplane associated with the

$\boldsymbol{n}$ – dimensional weight vextor $\vec{w}$ is the set

of all points $\vec{x} \in R^n$ for which $\vec{w} \cdot \vec{x} = 0$

Problem: Find such weights and threshold capable of absolutely separating two sets

=> e.g., **PERCEPTRON LEARNING ALGORITHM**

Assumption:

- $\boldsymbol{A}$ … a set of input vectors in $\boldsymbol{n}$ –dimensional space
- $\boldsymbol{B}$ … a set of input vectors in $\boldsymbol{n}$ –dimensional space

Separating hyperplane – for the extended weight and feature space (3)

SEPARATION of A and B:

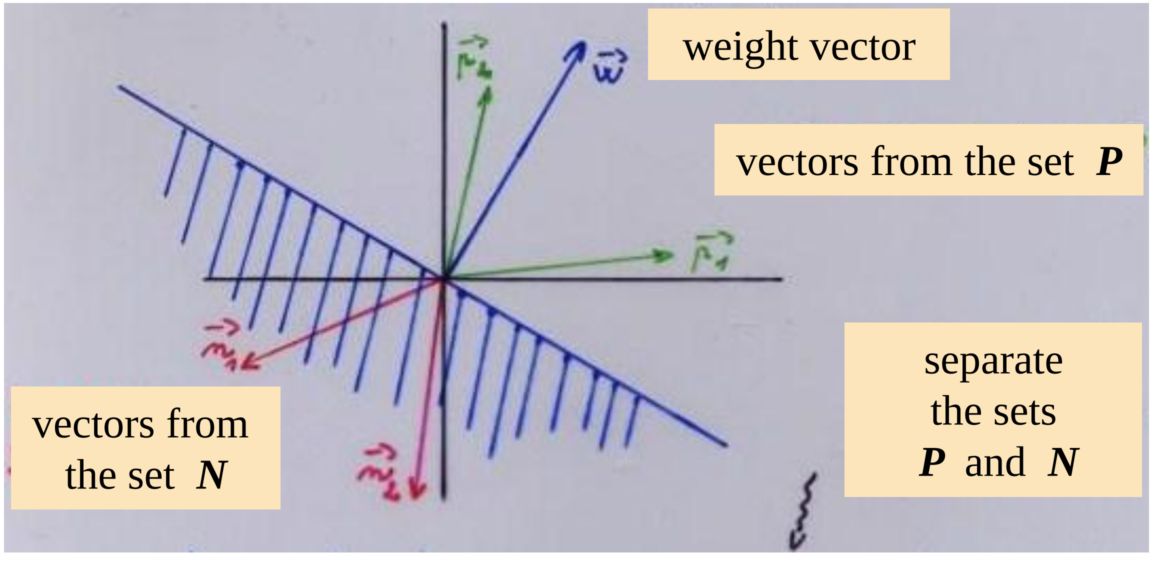
$\Rightarrow$ Perceptron should realize a binary function $f_{\vec{w}}$ such that $f_{\vec{w}}(\vec{x}) = 1 \quad \forall \vec{x} \in A$ and $f_{\vec{w}}(\vec{x}) = 0 \quad \forall \vec{x} \in B$

($f_{\vec{w}}$ depends on the weights and threshold, resp.)

- The error corresponds to the number of incorrectly classified points: $E(\vec{w}) = \sum_{\vec{x} \in A} \left(1 - f_{\vec{w}}(\vec{x})\right) + \sum_{\vec{x} \in B} f_{\vec{w}}(\vec{x})$

The goal of learning: minimize $E(\vec{w})$ in the weight space ($E(\vec{w}) = 0$)

Perceptron learning algorithm (1)



We are looking for a weight vector $\vec{w}$ with a positive scalar product with all the extended vectors represented by the points in ***P*** and with a negative product with the extended vectors represented by the points in ***N***

Perceptron learning algorithm (2)

⇒ **IN GENERAL:** assume that ***P*** and ***N*** are sets of ***n*** – dimensional vectors and a weight vector $\vec{w}$ must be found, such that:

$$\vec{w} \cdot \vec{x} > 0 \quad \forall \vec{x} \in P$$
$$\vec{w} \cdot \vec{x} < 0 \quad \forall \vec{x} \in N$$

- The perceptron learning algorithm starts with a randomly chosen vector $\vec{w}_0$
- If a vector $\vec{x} \in P$ is found such that $\vec{w} \cdot \vec{x} < 0$, this means that the angle between the two vectors is greater than $90°$
  → The weight vector must be rotated in the direction of $\vec{x}$
  (to bring this vector into the „positive“ half-space defined by $\vec{w}$)

Perceptron learning algorithm (3)

→ Rotation in the direction of $\vec{x}$ can be done by adding $\vec{w}$ and $\vec{x}$

- If a vector $\vec{x} \in N$ is found such that $\vec{w} \cdot \vec{x} > 0$, this means that the angle between the two vectors is smaller than $90°$
  - → The weight vector must be rotated away from $\vec{x}$
    (to bring this vector into the „negative“ half-space defined by $\vec{w}$)
  - → Rotation away from $\vec{x}$ can be done by subtracting $\vec{x}$ from $\vec{w}$
- the vectors from $\boldsymbol{P}$ thus rotate the weight vector in one direction, while the vectors from $\boldsymbol{N}$ do it in the opposite way
- If a solution exists, it can be found in a finite number of steps

Perceptron learning algorithm (4)

Step 1: Initialize the weights with small random values $w_i(0)$

$w_i(0)$ … the weight of the input i in time 0 ; $(1 \leq i \leq n+1)$

Step 2: Present a randomly selected training pattern in the form of

$(x_1, \ldots, x_{n+1})$ … input pattern and

$d(t)$ ………….... desired output pattern (for the presented input)

Step 3: Compute the actual response (network output)

$$y(t) = \operatorname{sgn}\left(\sum_{i=1}^{n+1} w_i(t)\, x_i(t) \right)$$

Step 4: Adjust the weights according to:

$w_i(t+1) = w_i(t)$ actual output is correct

$w_i(t+1) = w_i(t) + x_i(t)$ actual output is 0 but should be 1

$w_i(t+1) = w_i(t) - x_i(t)$ actual output is 1 but should be 0

Step 5: If the time t is smaller than the pre-set value, go to Step 2

Perceptron learning algorithm (5)

- **Heuristics for weight initialization:**
  Start with the averaged „positive“ input vector minus the averaged „negative“ vector

- **Modification** **learning rates** $\eta \ (0 \leq \eta \leq 1)$
  (adaptivity level of the weights ~ network plasticity)

  - Weight adjustment according to:

$w_i(t+1) = w_i(t)$	actual output is correct
$w_i(t+1) = w_i(t) + \eta \, x_i(t)$	actual output is 0 but should be 1
$w_i(t+1) = w_i(t) - \eta \, x_i(t)$	actual output is 1 but should be 0

Convergence of perceptron learning (Rosenblatt, 1959)

T: If the sets $\boldsymbol{P}$ and $\boldsymbol{N}$ are finite and linearly separable, the perceptron learning algorithm updates the weight vector $\vec{w}_t$ **a finite number of times**.

(If the vectors in $\boldsymbol{P}$ and $\boldsymbol{N}$ are tested cyclically one after the other, a weight vector $\vec{w}_t$ is found after a finite number of steps $\boldsymbol{t}$ which can separate the two sets $\boldsymbol{P}$ a $\boldsymbol{N}$.)

Proof: We will show that the perceptron learning works by bringing the initial vector $\vec{w}_0$ sufficiently close to the „solution vector“ $\vec{w}^*$.

Convergence of perceptron learning (2)

<u>Three simplifications</u> – without loosing generality:

a) Instead of $\boldsymbol{P}$ and $\boldsymbol{N}$ we form a single set $P' = P \cup N^-$ (N^- consists of „negated“ elements of $\boldsymbol{N}$)

b) The vectors in $\boldsymbol{P}'$ will be normalized
(If a weight vector $\vec{w}$ is found so that $\vec{w} \cdot \vec{x} > 0$, then this is also valid for any other vector $\eta \vec{x}$; $\eta > 0$.)

c) The weight vector will be also normalized
(The assumed normalized solution of the linear separation problem will be denoted as $\vec{w}^*$.)

Convergence of perceptron learning (3)

- Assume that after ***t + 1*** steps the weight vector $\vec{w}_{t+1}$ has been computed

  $\rightarrow$ this means, that at time ***t*** a vector $\vec{p}_i \in P'$ was incorrectly classified (by the weight vector $\vec{w}_t$), and hence $\vec{w}_{t+1} = \vec{w}_t + \vec{p}_i$

- The cosine of the angle $\boldsymbol{\rho}$ between $\vec{w}_{t+1}$ and $\vec{w}^*$ is:

$$\cos\rho = \frac{\vec{w}^* \cdot \vec{w}_{t+1}}{\left\|\vec{w}_{t+1}\right\|} \qquad (*)$$

Convergence of perceptron learning (4)

- For the numerator (*) we know that:

$$\vec{w}^* \cdot \vec{w}_{t+1} = \vec{w}^* \cdot \left(\vec{w}_t + \vec{p}_i\right) = \vec{w}^* \cdot \vec{w}_t + \vec{w}^* \cdot \vec{p}_i \geq \vec{w}^* \cdot \vec{w}_t + \delta$$

where $\delta = \min \left\{ \vec{w}^* \cdot \vec{p} \; ; \; \forall \vec{p} \in P' \right\}$

  - Since the weight vector $\vec{w}^*$ defines an absolute linear separation of $\boldsymbol{P}$ and $\boldsymbol{N}$, $\delta > 0$

    $\rightarrow$ by induction we obtain:

$$\vec{w}^* \cdot \vec{w}_{t+1} \geq \vec{w}^* \cdot \vec{w}_0 + (t+1)\,\delta \qquad (**)$$

Convergence of perceptron learning (5)

- At the same time, it holds for the denominator (*):

$$\|\vec{w}_{t+1}\|^2 = (\vec{w}_t + \vec{p}_i)\cdot(\vec{w}_t + \vec{p}_i) = \|\vec{w}_t\|^2 + 2\vec{w}_t \cdot \vec{p}_i + \|\vec{p}_i\|^2$$

  - Since $\vec{w}_t \cdot \vec{p}_i \leq 0$

    (Otherwise we would have not corrected $\vec{w}_t$ using $\vec{p}_i$.)

  - It holds: $\|\vec{w}_{t+1}\|^2 \leq \|\vec{w}_t\|^2 + \|\vec{p}_i\|^2 \leq \|\vec{w}_t\|^2 + 1$

    (As all vectors in P′ have been normalized.)

    $\rightarrow$ induction then gives us:

$$\|\vec{w}_{t+1}\|^2 \leq \|\vec{w}_0\|^2 + (t+1) \qquad (**)$$

Convergence of perceptron learning (6)

- From (**) and (***) we get in comparison with (*) the inequality:

$$\cos \rho \quad \geq \quad \frac{\vec{w}^* \cdot \vec{w}_0 + (t+1)\,\delta}{\sqrt{\|\vec{w}_0\|^2 + (t+1)}}$$

$\rightarrow$ the right term grows proportionally to $\sqrt{t}$ and, since $\delta > 0$, it can become arbitrarily large

$\times$ since $\cos \rho \leq 1$, t must be bounded by a maximum value and number weight adjustments must be finite.

QED

The pocket algorithm (1)
Gallant, 1990

(approximation of the „ideal linear separation“)

IDEA:

- Store the best weight vector found so far by perceptron learning in a „pocket“
- At the same time, continue updating the weight vector itself
- If a better weight vector is found, it supersedes the one currently stored and the algorithm continues to run

The pocket algorithm (2)

START:

- Initialize the weight vector $\vec{w}$ randomly and store the weight vector in the pocket: $\vec{w}_s = \vec{w}$
- Set the history of the stored weight vector to zero: $\boldsymbol{h_s = 0}$

ITERATION:

- Adjust $\vec{w}$ using a single iteration of the perceptron learning algorithm
- Update the number $\boldsymbol{h}$ of consecutively successfully tested vectors
- If $\boldsymbol{h > h_s}$, substitute $\vec{w}_s$ with $\vec{w}$ and $\boldsymbol{h_s}$ with $\boldsymbol{h}$
- Continue iterating

The pocket algorithm (3)

- Since only information from the last run of selected examples is considered, the algorithm can occasionally exchange a good stored weight vector for an inferior one – the probability of this event, however, decreases with the growing number of iterations
- If the training set is finite and the weights and vectors are rational, it can be shown that this algorithm converges to an optimal solution with probability 1