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

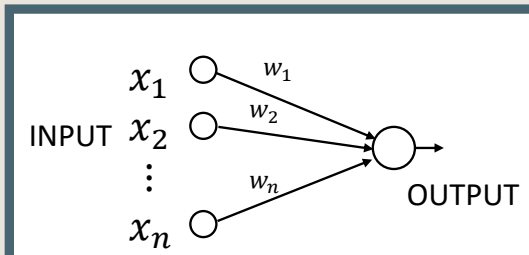
Neural Networks:

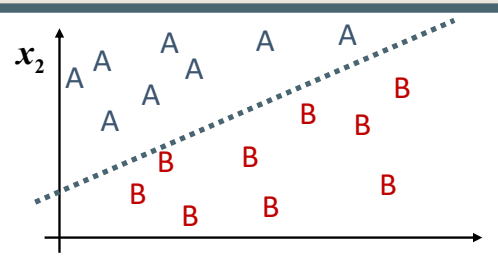
Contents:

- Introduction to the Field
- Perceptron and Linear Separability
- Multi-layered Neural Networks

Contents:

Formal neuron





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

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

Threshold ϑ (bias) is sometimes subtracted

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

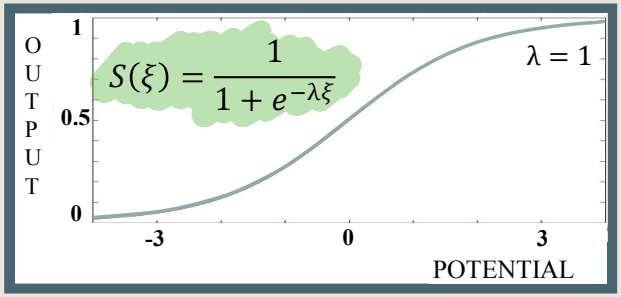
Types of transfer functions

$w^T x \geq 0$

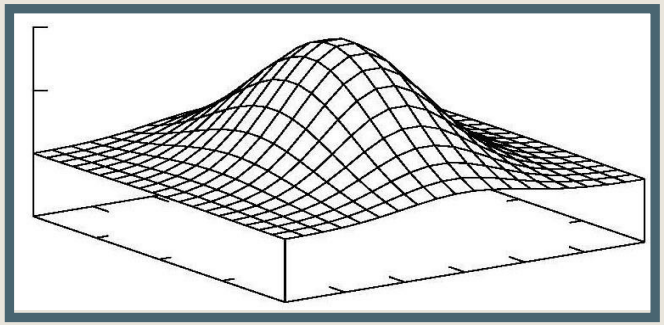
Hard-limiting

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

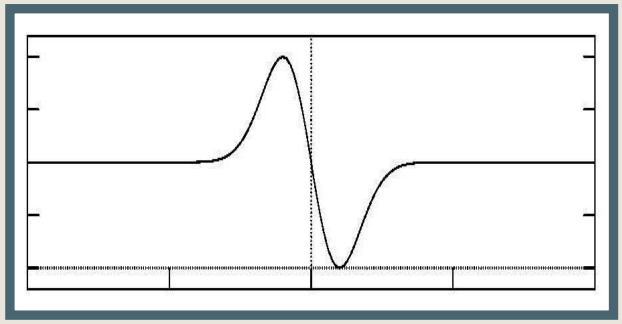
Sigmoidal



Radial basis (RBF)



Wavelet



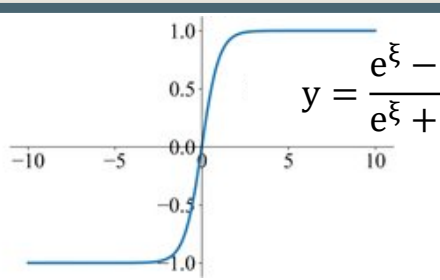
Types of transfer functions

Hard-limiting

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

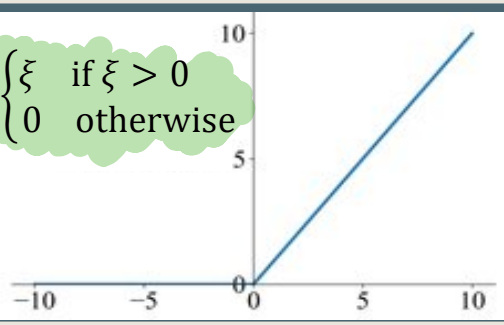
Tanh

$$y = \frac{e^{\xi} - e^{-\xi}}{e^{\xi} + e^{-\xi}}$$

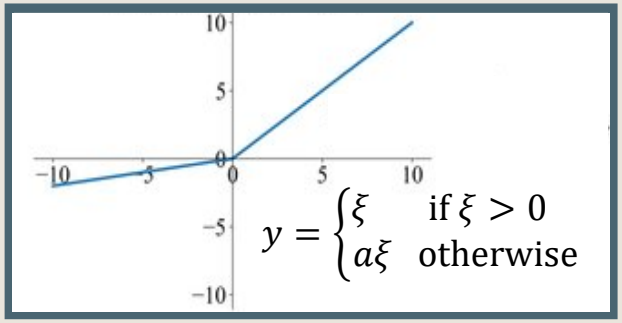


ReLU

$$y = \begin{cases} \xi & \text{if } \xi > 0 \\ 0 & \text{otherwise} \end{cases}$$



Leaky ReLU



Definition of a formal neuron

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

Most often, the so-called sigmoidal transfer function is considered with the values bound by 0 and 1: $y = f[\vec{w}, \vartheta](\vec{z}) = f(\xi) = \frac{1}{1+e^{-\xi}}$

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

– but we mostly ask whether $\sigma(x) \geq \frac{1}{2}$ or $\sigma(x) < \frac{1}{2}$. It is very unlikely to have it equal.

Specification of neuron states

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

- when $f[\vec{w}, \vartheta](\vec{z}) = 1$, we say that the neuron is **active**;
- when $f[\vec{w}, \vartheta](\vec{z}) = \frac{1}{2}$, we say that the neuron is **silent**;

  This fact indicates that the respective input is located on the separating hyperplane given by this neuron.

- when $f[\vec{w}, \vartheta](\vec{z}) = 0$, 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., by minimizing the mean squared error)
  - **Objective function:** e.g., $\sum_p \sum_j \left(y_{p,j} - d_{p,j}\right)^2$, -> ideální chci $\sqrt{\sum_i (y_i - t_i)^2}$

    $\vec{y}$ is the actual and $\vec{d}$ is the desired output
- **Recall** of newly presented input patterns:

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

Specification of training patterns

For a (neural) network B with n input and m output neurons:

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

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

$$T = \left\{ \left[\vec{x}_1, \vec{d}_1 \right], \dots, \left[\vec{x}_P, \vec{d}_P \right] \right\}.$$

Neural Networks:

Contents:

- Introduction to the Field
- Perceptron and Linear Separability
- Multi-layered Neural Networks

Contents:

requires linearly separable data!

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:

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

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

bias

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

fixed, not updated

$$\vec{x} = (x_1, x_2, \ldots, x_n, 1) \quad \Rightarrow \text{output 1, if } \vec{w} \cdot \vec{x} \geq 0$$

– then we can compare to 0 instead of ϑ.

Perceptron and linear separability (2)

Linear separability:

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

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

Perceptron and linear separability (3)

Absolute linear separability: ... nie nezůstane na dělicí čáře

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

Perceptron and linear separability (4)

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

– Vezmu si nejbližší bod od hranice u množiny B (ta bude prvky neležící na hranici) a o polovinu posunu tuto hranici směrem k nejbližšímu prvku v B. Pak budou mít obě nerovnosti ostré.

Perceptron and linear separability (5)

Further let: $\varepsilon = \max_{(b_1,\dots,b_n)\in B}\left(\sum_{i=1}^{n} w_i b_i - \vartheta\right)$, then clearly $\varepsilon < \frac{\varepsilon}{2} < 0$.

vybírám nejbližší bod

→ tento zápis nechápu smysl

Let $\vartheta' = \vartheta + \frac{\varepsilon}{2}$ (and 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 $\sum_{i=1}^{n} w_i a_i - \vartheta' \geq -\frac{1}{2}\varepsilon > 0$.

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

Perceptron and linear separability (6)

Similarly for all points in B

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

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

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

QED

Neural Networks:

Contents:

- Introduction to the Field
- Perceptron and Linear Separability
- Multi-layered Neural Networks

Contents:

- Introduction to the Field
  - Motivation and a Brief History
  - Biological Background
  - Adaptation and Learning
  - Feature Selection and Ordering
  - Probability and Hypotheses Testing (Review)
- Perceptron and Linear Separability
  - A Formal Neuron
  - Perceptron and Linear Separability
  - Perceptron Learning Algorithm
  - Convergence of Perceptron Learning
  - The Pocket Algorithm

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

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

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

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

D The separating hyperplane associated with the n–dimensional weight vector $\vec{w}$ is the set of all points $\vec{x} \in R^n$ for which $\vec{w} \cdot \vec{x} = 0$

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

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

Assumption:

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

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

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

SEPARATION of A and B:

=> Perceptron should realize a binary function $f_{\vec{w}}$ such that

$f_{\vec{w}}(\vec{x}) = 1 \quad \forall\, \vec{\boldsymbol{x}} \in \boldsymbol{A}$ and $f_{\vec{w}}(\vec{x}) = 0 \quad \forall\, \vec{x} \in B$

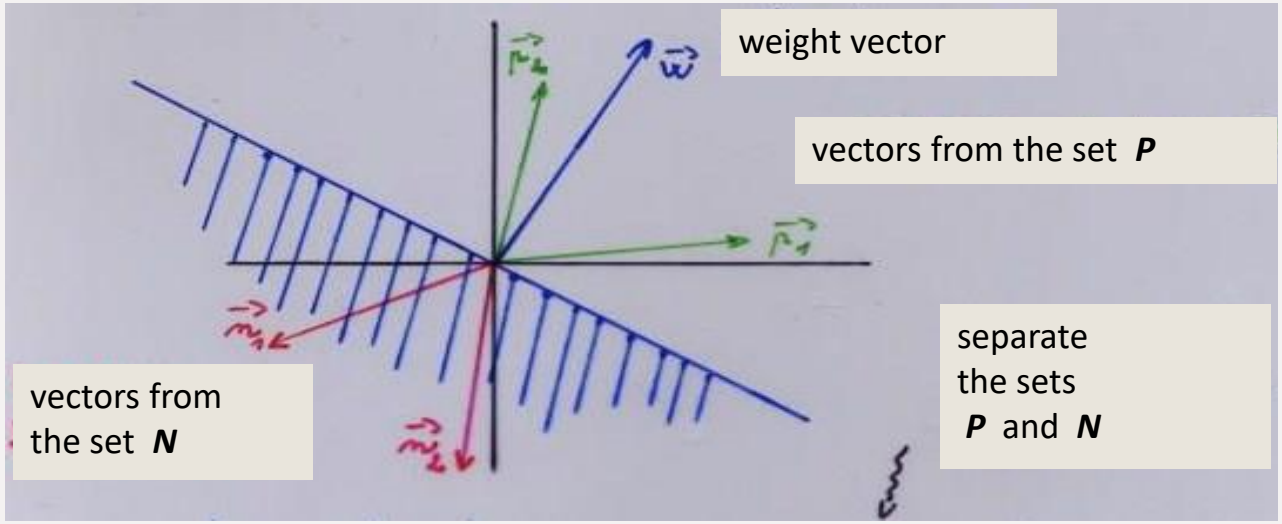
($f_{\vec{w}}$ depends on the weights and threshold, resp.).

=> The error corresponds to the number of incorrectly classified points:

$$E(\vec{w}) = \sum_{\vec{\boldsymbol{x}} \in \boldsymbol{A}} \left(1 - f_{\vec{w}}(\vec{x})\right) + \sum_{\vec{x} \in B} f_{\vec{w}}(\vec{x})$$

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

Perceptron learning algorithm (1)



$\vec{w}$

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

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 $\vec{w}$).

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

Perceptron learning algorithm (3)

If a vector $\forall \vec{x} \in N$ 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 P thus rotate the weight vector in one direction, while the vectors from 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 $w_i(0)$ ($w_i(0)$ is the weight of input $\boldsymbol{i}$ at time 0 ; $\boldsymbol{1 \leq i \leq n+1}$).

Step 2: Present a randomly selected training pattern in the form of the input pattern $(x_1(t), \ldots, x_{n+1}(t))$ and the desired output pattern $d(t)$ (for the presented input).

Step 3: Compute the actual response (network output)

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

Step 4: Adjust the weights according to:

$w_i(t+) = w_i(t)$ if the actual output is correct

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

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

$y = x^T w$

if $t_i y < 0$

$w \mathrel{+}= t_i x$

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 $\alpha\ (0 \leq \alpha \leq 1)$

(adaptivity level of the weights ~ network plasticity)

- Weight adjustment according to:

$w_i(t+) = w_i(t)$ if the actual output is correct

$w_i(t+) = w_i(t) + \alpha\, x_i(t)$ if the actual output is 0 but should be 1

$w_i(t+) = w_i(t) - \alpha\, x_i(t)$ if the actual output is 1 but should be 0

Neural Networks:

Contents:

- Introduction to the Field
- Perceptron and Linear Separability
- Multi-layered Neural Networks

Contents:

Only if data are linearly separable...

Convergence of perceptron learning
(Rosenblatt, 1959)

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

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