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:

Multi-layered Neural Networks: Analysis of Their Properties

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:

- **Multi-layered Neural Networks**
- **Multi-layered Neural Networks: Analysis of Their Properties**
- **Multi-layered Neural Networks: an Application Example**

Contents:

Kolmogorov´s Theorem – 1957

13. Hilbert problem

- Can every continuous function of n arguments be represented using a finite composition of functions of at most two arguments?
  - Example: $x \cdot y = \exp\,(\ln x + \ln y)$

Theorem (Kolmogorov, 1957): Let $f : [0,1]^n \rightarrow [0,1]$ be a continuous function.

There exist functions of one argument g and φ_q, for $q = 1, \dots, 2n+1$ and constants λ_p, for $p = 1, \dots, n$ such that

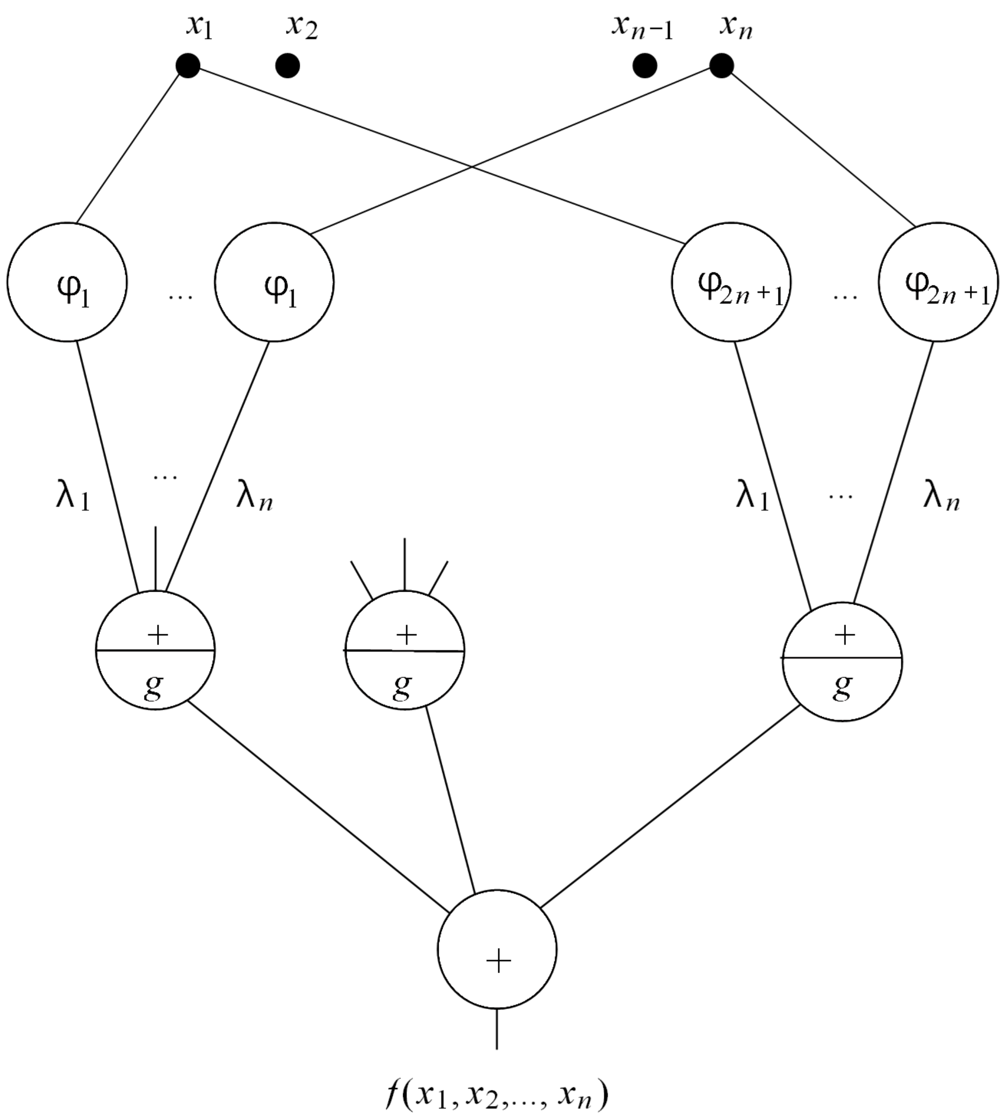
$$f(x_1, \dots, x_n) = \sum_{q=1}^{2n+1} g\left(\sum_{p=1}^{n} \lambda_p \varphi_q\left(x_p\right)\right)$$

Kolmogorov Networks

- to represent continuous functions of n variables

$$f(x_1, \dots, x_n) =$$

$$= \sum_{q=1}^{2n+1} g\left(\sum_{p=1}^{n} \lambda_p \varphi_q(x_p)\right)$$



Neural Networks:

Contents:

- **Multi-layered Neural Networks**
- **Multi-layered Neural Networks: Analysis of Their Properties**
- **Multi-layered Neural Networks: an Application Example**

Contents:

Function Approximation (1)

- Any continuous function can be reproduced exactly by a finite network of computing units, whereby the necessary primitive functions for each node exist

  × the choice of the right transfer function!

- The best possible approximation to a given function

  × the choice of the right number of computing units with the considered transfer function

⇒ Instead of an exact representation, try to approximate

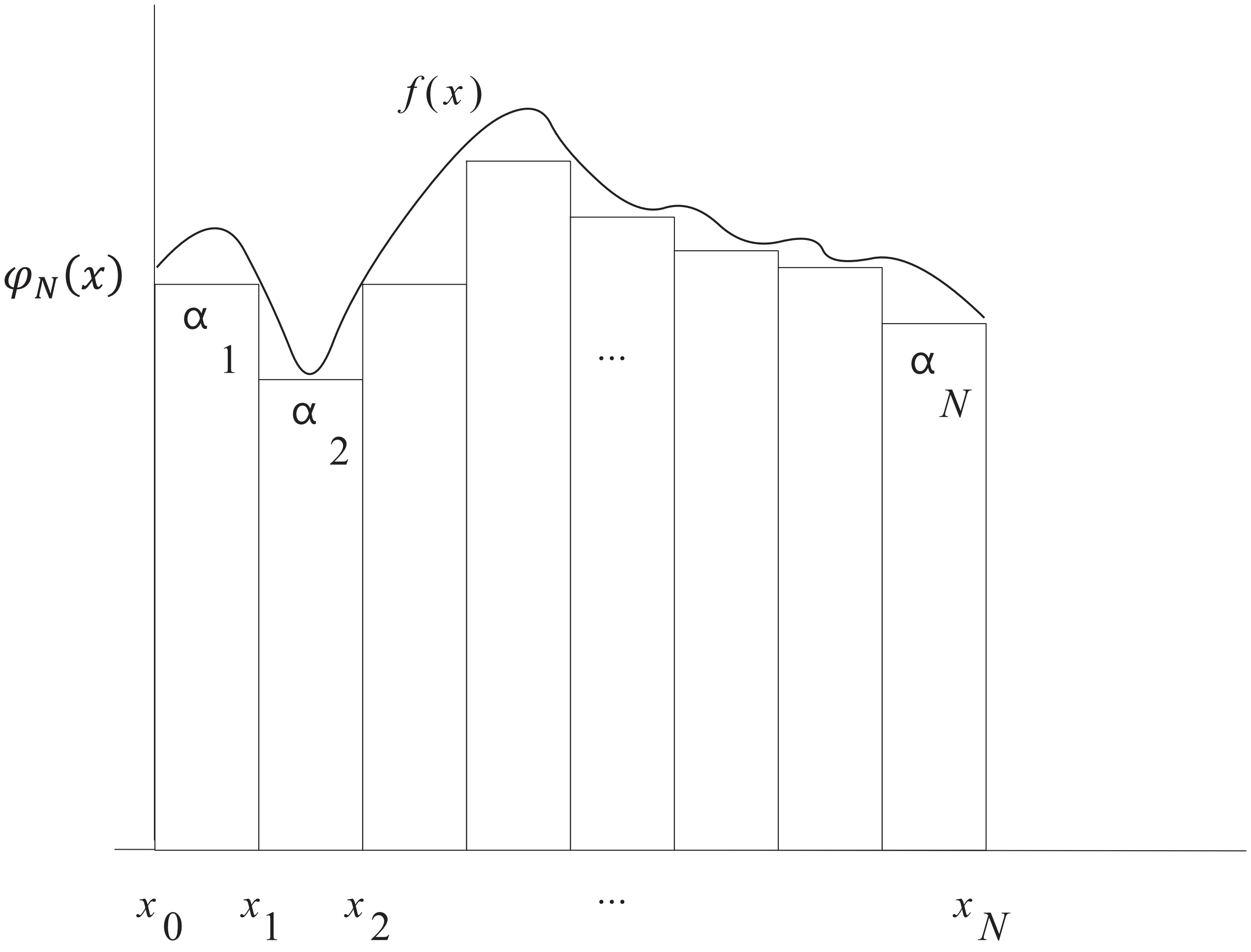
Function Approximation (2)

Theorem: Any continuous real function $f: [0, 1] \rightarrow [0, 1]$ can be approximated using a network of threshold elements in such a way that the total approximation error E is lower than any given real number $\varepsilon > 0$:

$$E = \int_0^1 \left| f(x) - \tilde{f}(x) \right| \, \mathrm{d}x < \varepsilon$$

where $\tilde{f}$ denotes the network function.

Function Approximation (3)



Proof - idea:

- approximation of f by means of φ_N

Inspirováno Riemannovským integrálem

Function Approximation (4)

Proof (continued):

- Divide the interval $[0, 1]$ into N equal segments selecting the points $x_0, x_1, \dots, x_N \in [0, 1];\ x_0 = 0,\ x_N = 1$
- Define a function φ_N as it follows:

$$\varphi_N(x) = \min\{\, f(x');\, x' \in [x_i\,, x_{i+1})\ \text{for}\ \ x_i \le x < x_{i+1}\}$$

- Further, consider φ_N an approximation of f so that the approximation error E_N is given by:

$$E_N \ = \int_0^1 |f(x) - \varphi_N(x)|\ \mathrm{d}x$$

Function Approximation (5)

Proof (continued):

- Since $f(x) \geq \varphi_N(x) \quad \forall x \in [0,1]$, E_N corresponds to

$$E_N = \int_0^1 f(x)\,\mathrm{d}x - \boxed{\int_0^1 \varphi_N(x)\,\mathrm{d}x}$$

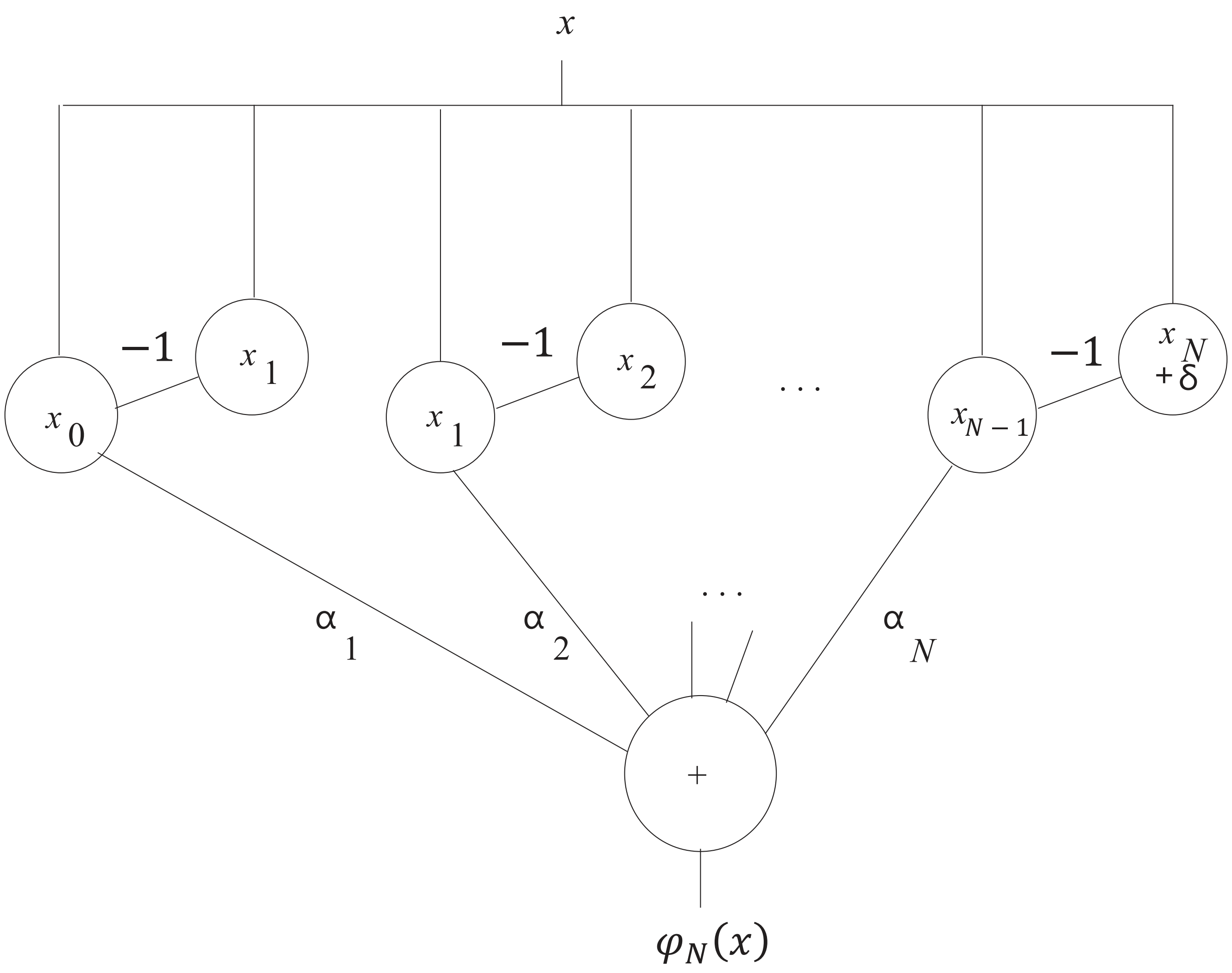
~ lower Riemann sum of the function f

- Since continuous functions are integrable → the lower sum of f converges in the limit $N \to \infty$ to the integral of f in the interval $[0,1]$
- Thus, it holds $E_N \to 0$ when $N \to \infty$, hence for any real number $\varepsilon > 0$ there exists an M such that $E_N < \varepsilon \ \forall N \geq M$
- The function φ_N is therefore the desired approximation of f.

Function Approximation (6)

Proof (continued):

- The function $\varphi_N(x)$ can be computed by a network of threshold units (~ a neural network)
  - $\varphi_N(x)$ is a step-wise function
  - in each of the N segments of the interval $[0, 1]$: $[x_0, x_1), [x_1, x_2), \ldots, [x_{N-1}, x_N]$, $\varphi_N(x)$ has the respective value $\alpha_1, \ldots, \alpha_N$



Udělám si dostatečně jemnou síť, abych na intervalu [0,1] pokryl všechny významné body (změny y-směru), ve kterých budu mít threshold a budu na něm porovnávat

Function Approximation (7)

Proof (continued):

This network can compute the step-wise function $\varphi_N(x)$:

- The single input to the network is x (from the interval $[0,1]$)
- Each pair of units with the thresholds x_i and x_{i+1} guarantees that the unit with threshold x_i will be active when $x_i \leq x < x_{i+1}$.
- The (linear) output unit adds all outputs of the previous layer of units and produces their (weighted) sum as a result
- The unit with the threshold $x_N + \delta$, where δ is a small positive number, is used to recognize the case $x_{N-1} \leq x \leq x_N$.

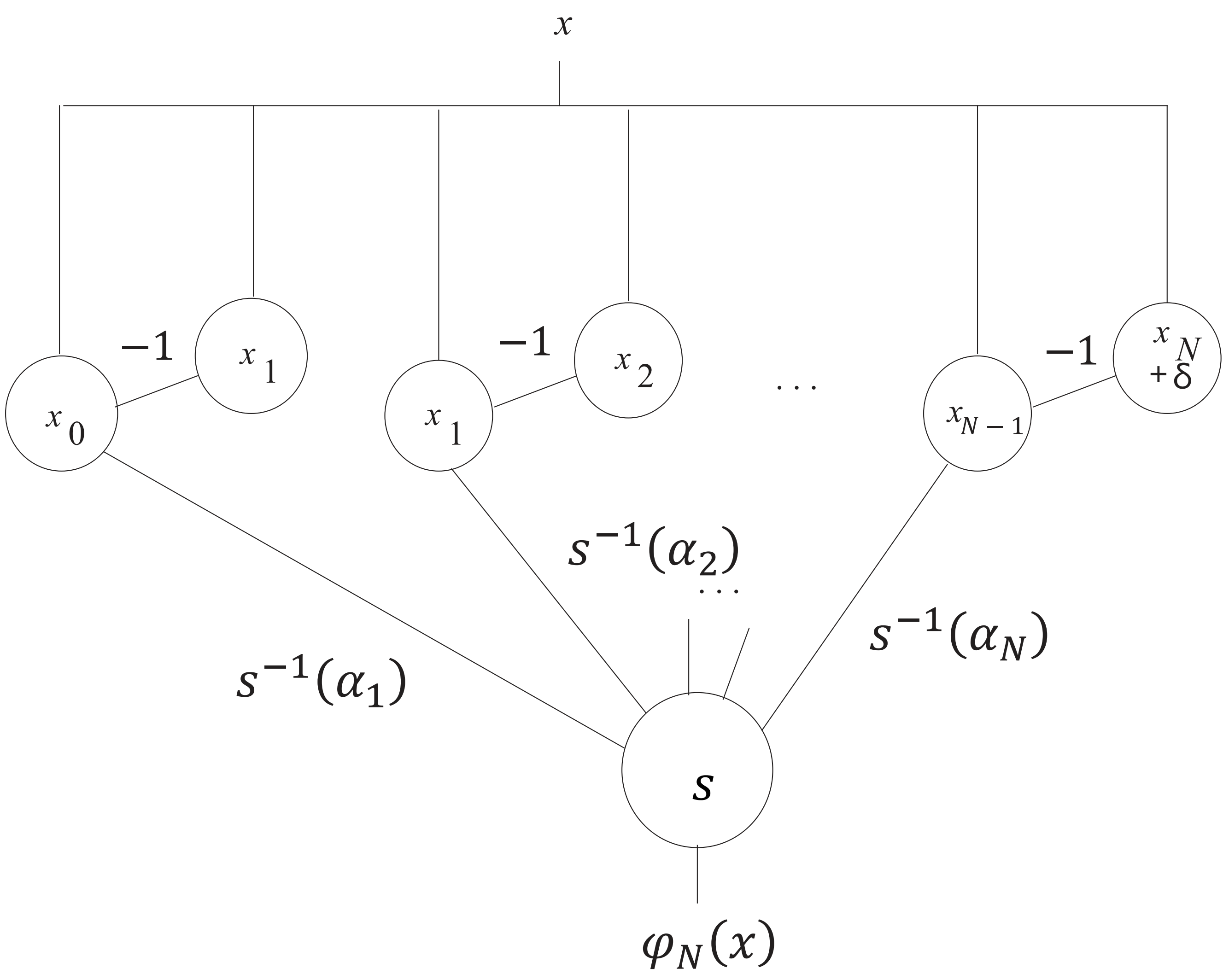
This network computes the function $\varphi_N(x)$, that approximates the function f with the desired maximum error.

QED

Function Approximation (8)

Corollary:

The theorem is valid also for any function $f: [0, 1] \to (0, 1)$ and networks with the sigmoidal transfer function.

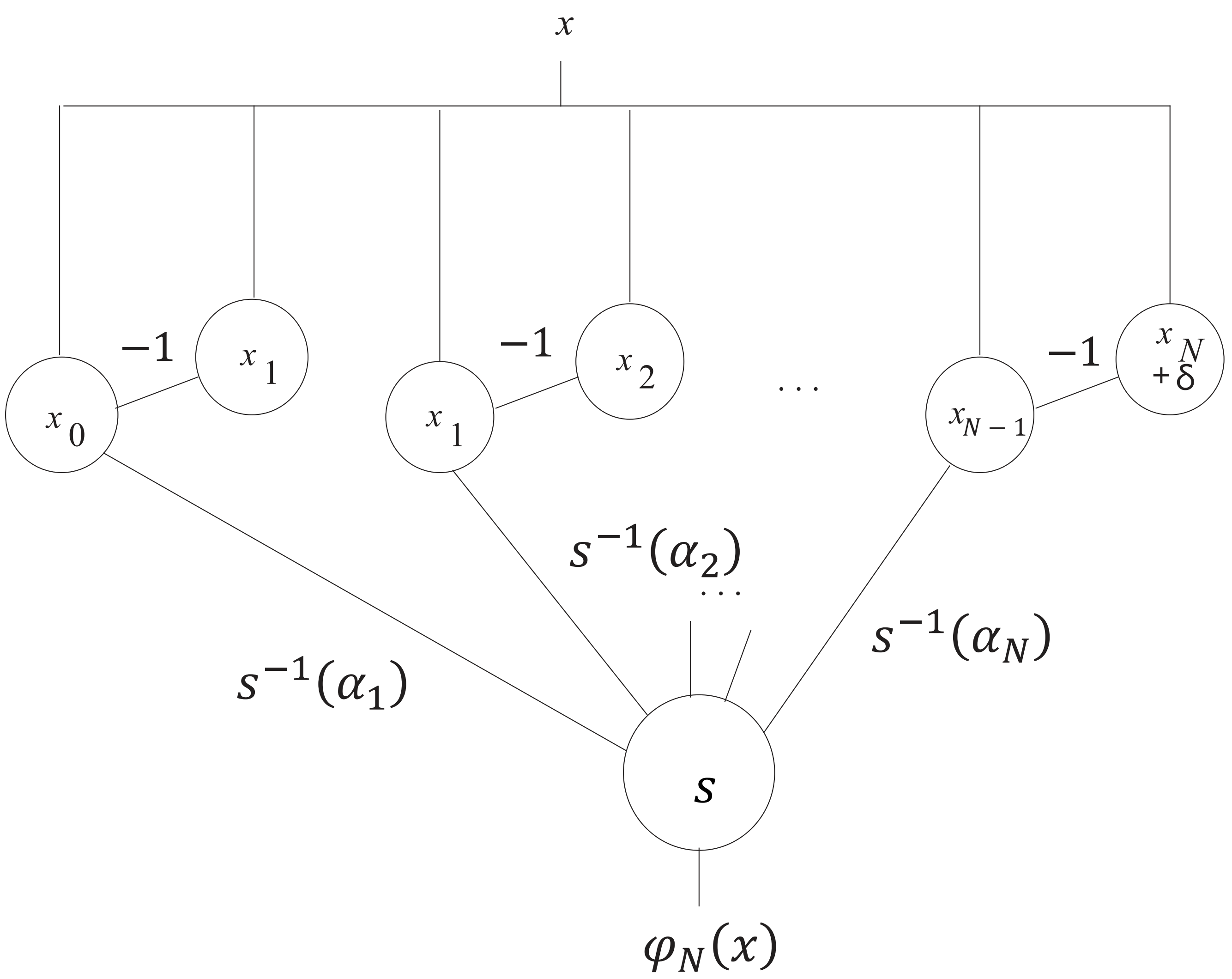


Function Approximation (9)

Proof:

- The image of the function f has been limited to the interval $(0, 1)$ in order to simplify the proof
- The function f can be approximated using the sketched network
- The transfer function of the units with the threshold x_i is given by $s_c(x - x_i)$, where c controls the slope of the function

$$s_c(x - x_i) = \frac{1}{1 + e^{-c(x - x_i)}}$$



Function Approximation (10)

Proof (continued):

- The network can approximate the function $\varphi_N(x)$ with an approximation error lower than any desired bound (> 0)

  ~ threshold functions can be approximated with any desired precision by a sigmoidal function parametrized with c)

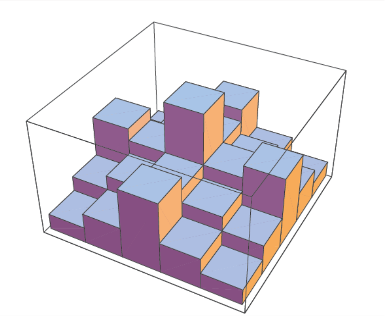
- The weights connecting the first layer of units to the output unit have been set in such a way that the sigmoid produces the desired values α_i as a result

- Further it should be guaranteed that every input x produces a single 1 from the first layer to the output unit

  → the first layer just finds out to which of the N segments of the interval $[0, 1]$ the input x belongs

QED

Function Approximation (11)

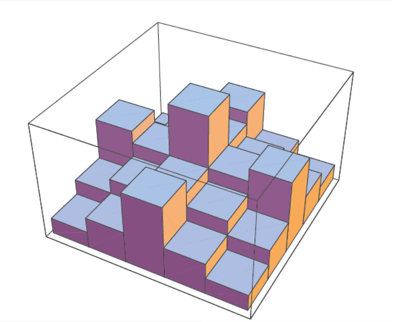


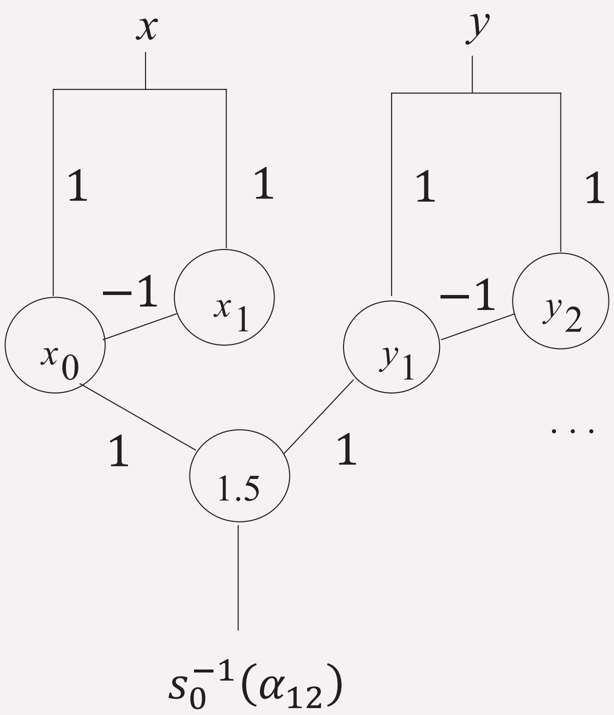
The multidimensional case:

The network capable of approximating the function $f: [0,1]^n \to (0,1)$ can be constructed using the same general idea as before in the one-dimensional case

- **Extensions necessary for the two-dimensional case**
  - Recognition of intervals in the x and y domains
    - 2 units left are used to test $x_0 \leq x < x_1$
    - 2 units right are used to test $y_1 \leq y < y_2$
  - The unit with the threshold 1.5 recognizes the conjunction of both conditions (for x and y)

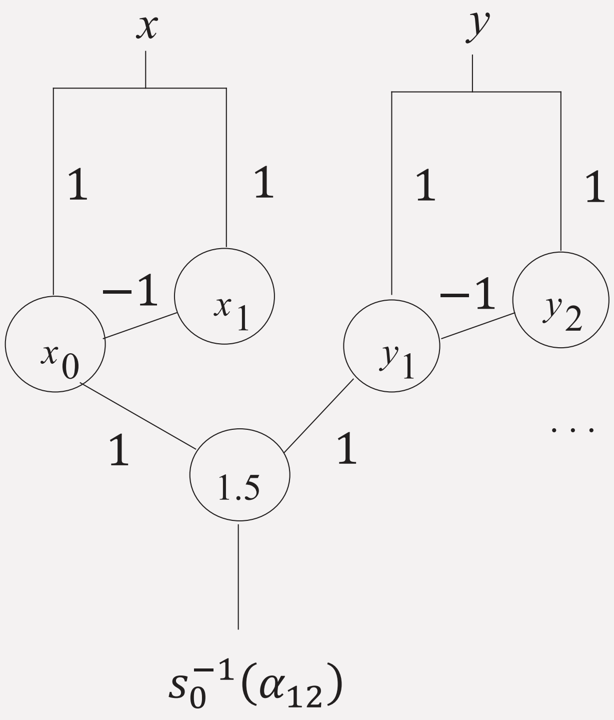
Function Approximation (12)





Extensions necessary for the 2D-case

- Recognition of intervals in the x and y domains
  - 2 units left are used to test $x_0 \leq x < x_1$
  - 2 units right are used to test $y_1 \leq y < y_2$
- The unit with the threshold 1.5 recognizes the conjunction of both conditions (for x and y)
- The „output" has the weight $s_0^{-1}(\alpha_{12})$, so the sigmoidal transfer function yields α_{12}
  → this number corresponds to the desired approximation of the function f on $[x_0, x_1) \times [y_1, y_2)$



Extensions necessary for the 2D-case

- Recognition of intervals in the x and y domains
  - 2 units left are used to test $x_0 \leq x < x_1$
  - 2 units right are used to test $y_1 \leq y < y_2$
- The unit with the threshold 1.5 recognizes the conjunction of both conditions (for x and y)
- The „output“ has the weight $s_0^{-1}(\alpha_{12})$, so the sigmoidal transfer function yields α_{12}
  → this number corresponds to the desired approximation of the function f on $[x_0, x_1) \times [y_1, y_2)$

Neural Networks:

Contents:

- **Multi-layered Neural Networks**
- **Multi-layered Neural Networks: Analysis of Their Properties**
- **Multi-layered Neural Networks: an Application Example**

Contents:

The Complexity of Learning

The satisfiability problem

D: Let V be a set of n logical variables, and let F be a logical expression in conjunctive normal form (conjunction of disjunctions of literals) which contains only variables from V.

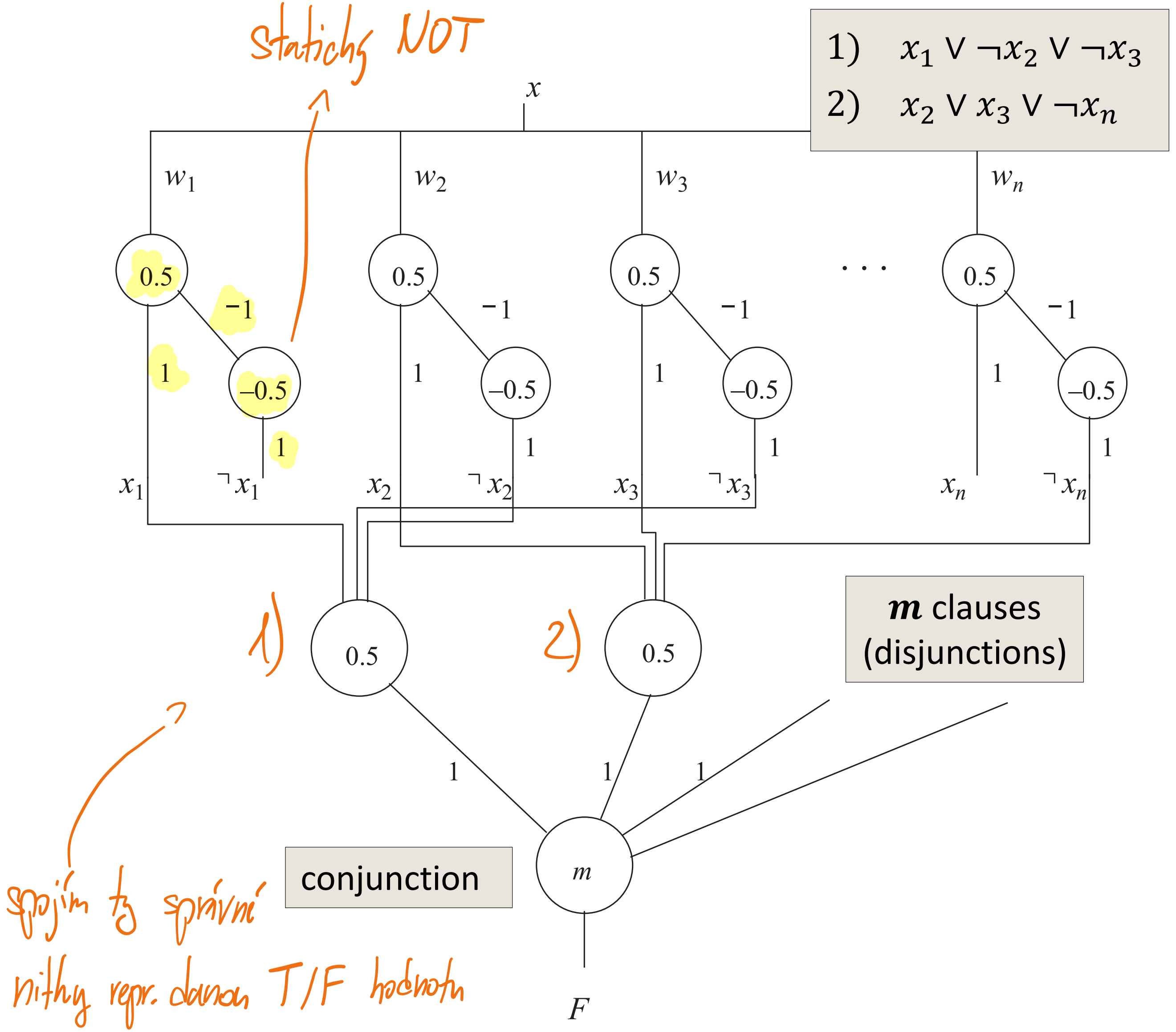
The satisfiability problem consists in assigning truth values to the variables in V in such a way that the expression F becomes true.

THEOREM: The general learning problem for networks of threshold functions is NP-complete.

The Complexity of Learning (2)

Proof - idea:

- a network equivalent to 3SAT



The Complexity of Learning (3)

Proof (continue):

1. 3SAT can be reduced to an instance of a learning problem for neural networks in polynomial time

   A logical expression F in conjunctive normal form, which contains n variables can be transformed in polynomial time in the description of a network of the above type:

   - For each variable x_i a weight w_i is defined
   - The connections to the third layer are fixed according to the conjunctive normal form we are dealing with

The Complexity of Learning (4)

Proof (continue):

- This can be done (using a suitable coding) in polynomial time, because it holds for the number m of different possible disjunctions in a 3SAT formula that $m \leq (2n)^3$
- If an instantiation A with logical values of the variables x_i exists, such that F becomes true, then there exist weights $w_1, w_2, \ldots, w_n$, that solve the learning problem

The Complexity of Learning (5)

Proof (continue):

- It is sufficient to set the weights $w_i = 1$, if $x_i = 1$;
  and $w_i = 0$, if $x_i = 0$.
  (in both cases, we thus choose $w_i = x_i$)

- Similarly in the opposite way:
  if there exist weights $w_1, w_2, \dots, w_n$, that solve the learning problem, then the instantiation $x_i = 1$ for $w_i \geq 0.5$ and $x_i = 0$ otherwise, is a valid instantiation that makes F true

The Complexity of Learning (6)

Proof (continue):

2. Further, we must show that the learning problem belongs to the class NP (its solution can be checked in polynomial time)

   - If the weights $w_1, w_2, \dots, w_n$ are given**,** then a single run of the network can be used to check if the output F is equal to 1
   - The number of computation steps is directly proportional to the number n of variables and to the number m of disjunctive clauses (which is bounded by the polynomial $(2n)^3$)

The Complexity of Learning (7)

Proof (continue):

- The time required to check an instantiation is therefore bounded by a polynomial in n
- The given learning problem thus belongs to the class NP

QED

Remark:

For some special types of simple neural networks, the learning problem can be solved in polynomial time (by means of linear programming algorithms)

Neural Networks:

Contents:

- **Multi-layered Neural Networks**
- **Multi-layered Neural Networks: Analysis of Their Properties**
- **Multi-layered Neural Networks: an Application Example**

Contents:

- Multi-layered Neural Networks
  - Back-Propagation Training Algorithm
  - Strategies to Speed-up the Training Process
- Multi-layered Neural Networks: Analysis of Their Properties
  - Kolmogorov´s Theorem
  - Function Approximation
  - The Complexity of Learning
  - The Number of Regions in the Feature Space
  - Vapnik-Chervonenkis Dimension
- Multi-layered Neural Networks: an Application Example
  - Internal Knowledge Representation and Pruning
  - Sensitivity Analysis and Feature Selection
  - Analysis of the World Bank Data

The XOR Problem (1)

16 Boolean functions of two variables

AND OR

x_1	x_2	f_0	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}	f_{15}
0	0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
0	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	0	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

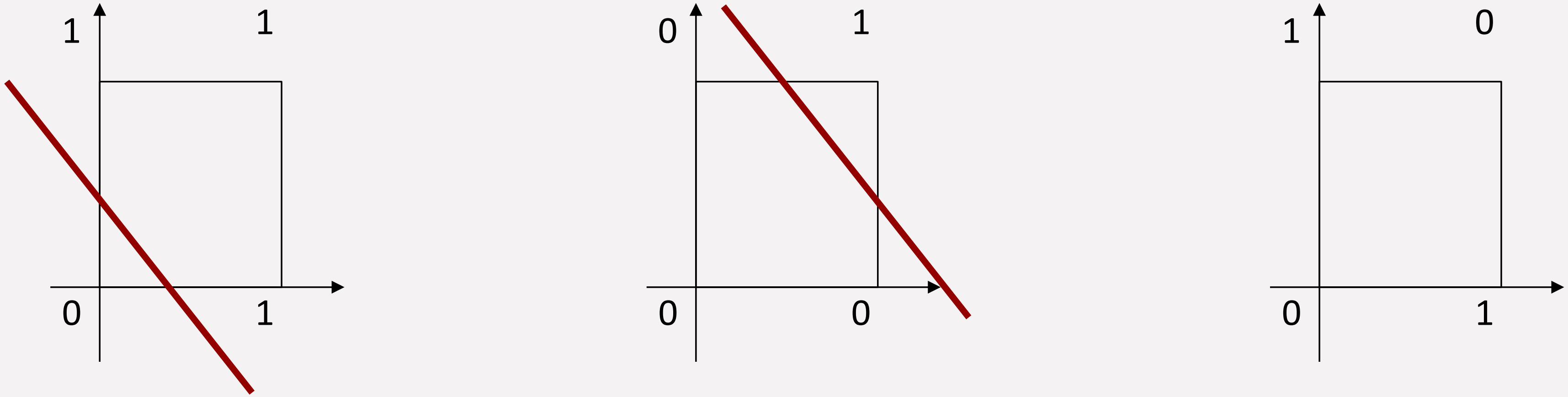
XOR

- Input vector $\vec{x} = (x_1, x_2)$ is a point in the two-dimensional space; f_j defines the labeling for all points (x_1, x_2), where $x_i \in \{0,1\}$
- The perceptron can compute functions for which a hyperplane can separate the points labeled by 0 from the points labeled by 1

The XOR Problem (2)

16 Boolean functions of two variables

OR AND XOR



The Number of Regions in the Feature Space (1)

- The capacity of a neuron depends on the dimension of the weight space and the number of cuts with separating hyperplanes

→ Question:

How many regions are defined by m cutting hyper-planes of the dimension $n-1$ in an n-dimensional space?

- we consider only hyperplanes going through the origin

→ The intersection of l hyperplanes; $l \leq n$ is of dimension $n-l$

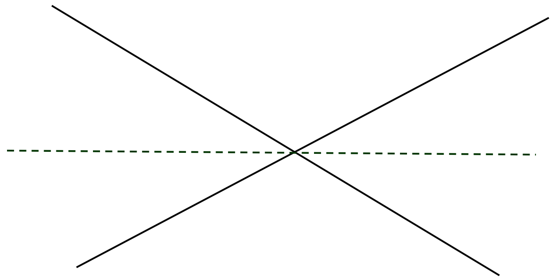
The Number of Regions in the Feature Space (2)

- <u>2-dimensional case:</u>

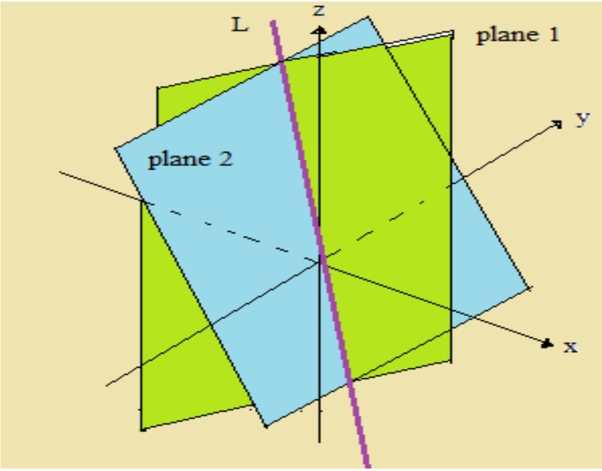
  m lines going through the origin define at most $2 \cdot m$ different regions
- <u>3 – dimensional case:</u>

  each new cut increases the number of regions two times
- <u>in general:</u>

  n cuts with $(n-1)$-dimensional hyperplanes in n-dimensional space define at most 2^n different regions



+ 2 new regions



The Number of Regions in the Feature Space (3)

Theorem:

Let $R(m, n)$ denote the number of different regions defined by m separating hyperplanes of the dimension $n-1$ in an n-dimensional space.

We set $R(1, n) = 2$ for $n \geq 1$ and $R(m, 0) = 0 \ \forall\, m \geq 1$.

Then for $n \geq 1$ and $m > 1$:

$$R(m, n) = R(m-1, n) + R(m-1, n-1)$$

The Number of Regions in the Feature Space (4)

Proof (by induction on m):

1. $m = 2$ and $n = 1$: The formula is valid, because
$$R(2,1) = R(1,1) + R(1,0) = 2 + 0 = 2$$
2. $m = 2$ and $n \geq 2$: $R(2,n) = 4$ => valid, because
$$R(2,n) = R(1,n) + R(1,n-1) = 2 + 2 = 4$$
3. $m + 1$ hyperplanes of dimension $n - 1$ are given in the n-dimensional space and in general position $(n \geq 2)$:
   - The first m hyperplanes define $R(m,n)$ regions in the n-dimensional space

The Number of Regions in the Feature Space (5)

Proof (continue):

- $(\boldsymbol{m}+\mathbf{1})$-st hyperplane intersects the first $\boldsymbol{m}$ hyperplanes in $\boldsymbol{m}$ hyperplanes of dimension $\boldsymbol{n}-\mathbf{2}$
- These $\boldsymbol{m}$ hyperplanes (of dimension $\boldsymbol{n}-\mathbf{2}$) divide the $(\boldsymbol{n}-\mathbf{1})$-dimensional space into $\boldsymbol{R}(\boldsymbol{m},\boldsymbol{n}-\mathbf{1})$ regions
- After the cut with the hyperplane $(\boldsymbol{m}+\mathbf{1})$, exactly $\boldsymbol{R}(\boldsymbol{m},\boldsymbol{n}-\mathbf{1})$ new regions have been created
- → The new number of regions is therefore:
- $\boldsymbol{R}(\boldsymbol{m}+\mathbf{1},\boldsymbol{n}) = \boldsymbol{R}(\boldsymbol{m},\boldsymbol{n}) + \boldsymbol{R}(\boldsymbol{m},\boldsymbol{n}-\mathbf{1})$

QED

Number of Regions in the Feature Space (6)

- A useful alternative for $R(m,n)$:

$$R(m,n) = 2 \sum_{i=0}^{n-1} \binom{m-1}{i}$$

- x With growing n, the number of Boolean functions grows significantly quicker than the number of regions formed by hyperplanes in a general position
  - this number can be in general larger than the number of threshold functions over binary inputs

Number of Regions in the Feature Space (7)

Example:

n	# Boolean functions	# threshold functions	# regions
1	4	2	2
2	16	14	14
3	256	104	128
4	65536	1882	3882
5	$4.3\ \times 10^9$	94572	412736

Number of Regions in the Feature Space (8)

Consequences:

- **Learnability problems** ~ if the number of input vectors is too high, the network might be not able to form enough regions with the given number of hidden neurons
- **Generalization**
  ~ expected number of correctly classified examples
- **Over-fitting**
  ~ erroneous interpolation of patterns outside of the training set
- **Vapnik-Chervonenkis dimension (VC-dimension)**
  ~ finite VC-dimension → „the class of concepts“ is learnable

Neural Networks:

Contents:

- Multi-layered Neural Networks
- Multi-layered Neural Networks: Analysis of Their Properties
- Multi-layered Neural Networks: an Application Example

Contents:

Vapnik-Chervonenkis Dimension (VC-dimension) (1)

Definition:

Let $C = \{f_i\}$ be a set of functions (concept class).
The set of m training patterns $\{t_k\}_{k=1,\dots,m}$ *can be shattered* by means of C, if for each of the 2^m possible labelings of these patterns with $1/0$, there exists at least one function, that satisfies this labeling.

Definition:

Kolik nejvíc těmi funkcemi můžu rozlišit.

The *VC-dimension* V *of a set of functions* C is defined as the biggest m, for which a set of m training patterns exists that can be shattered.

Vapnik-Chervonenkis Dimension (VC-dimension) (2)

- If there exists for any m a set of m training patterns, that can be shattered by means of C, the VC-dimension of C is infinite

  → Such a problem is not „ L E A R N A B L E “

- In general, the VC-dimension of a set of functions does not depend on the number of parameters
- VC-dimension impacts adequate generalization
  - The network can have many parameters, but it should have a small VC-dimension → better generalization
  - High VC-dimension correlates with worse generalization

Vapnik-Chervonenkis Dimension (VC-dimension) (3)

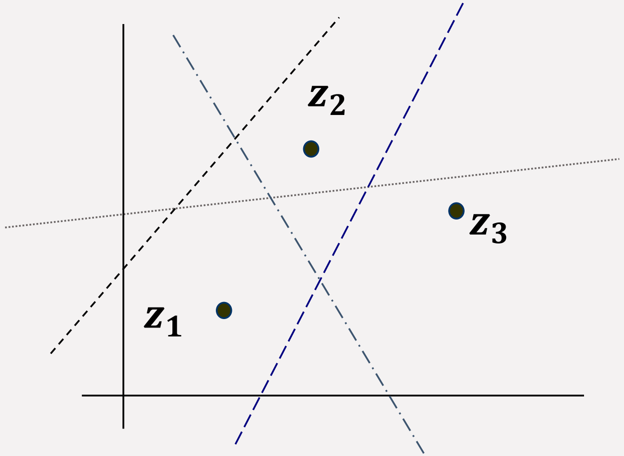
Example:

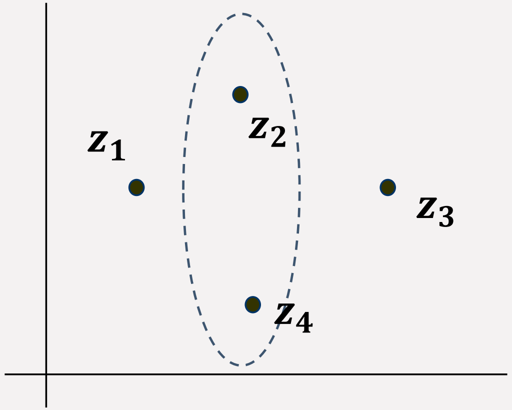
1. The VC-dimension of a set of linear indicator functions

   $Q(\vec{z}, \alpha) = \Theta\left\{\sum_{p=1}^{n} \alpha_p z_p + \alpha_0\right\}$ in the n-dimensional space is $n + 1$,

   (i.e., it can shatter at most $n + 1$ patterns)

$n = 2$





Vapnik-Chervonenkis Dimension (VC-dimension) (4)

$$\theta(x) = \begin{cases} 1 & for\ x \geq 0 \\ 0 & for\ x < 0 \end{cases}$$

2. VC-dimension of the set of functions $f_\alpha(z) = \theta(cos\ \alpha z),\ \alpha \in R$ is infinite
   - The points $z_1 = 10^{-1}, \dots, z_m = 10^{-m}$ can be shattered by means of the functions from this set
   - To shatter these patterns into two classes $(+1/-1)$ given by the sequence $\delta_1, \dots, \delta_m;\ \delta_i \in \{0, 1\}$ it is sufficient to choose the value of the parameter as

$$\alpha = \left(1 + \sum_{i=1}^{m} (1 - \delta_i)\ 10^i\right) \cdot \pi$$

Vapnik-Chervonenkis Dimension (VC-dimension) (5)

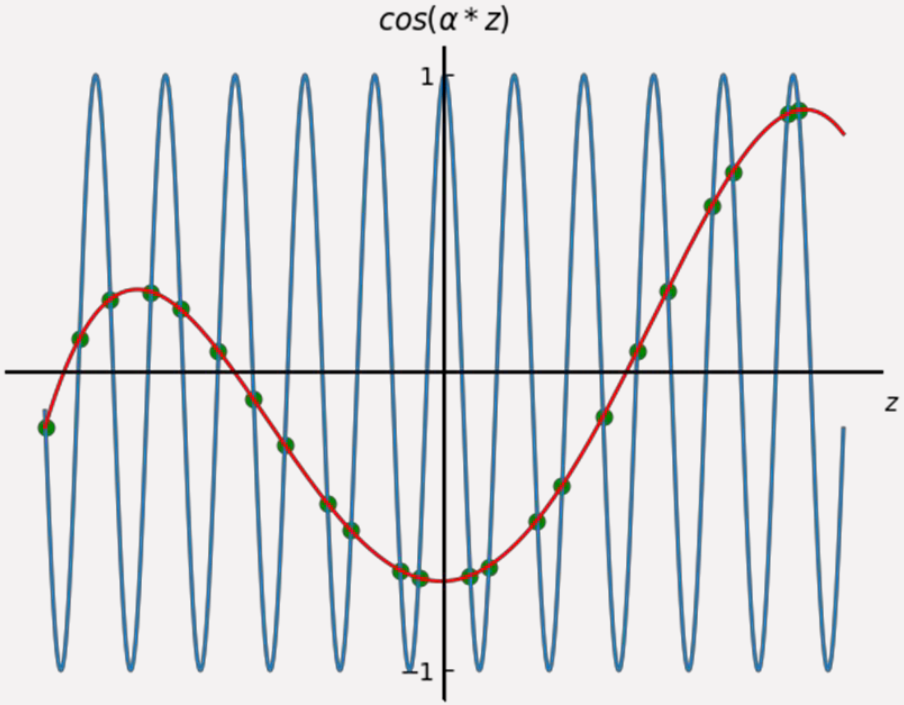
For the set of functions $f_\alpha(z) = \theta(cos\ \alpha z),\ \ \alpha \in R$

- The points $z_1 = 10^{-1}, \dots, z_m = 10^{-m}$ can be shattered by this set of functions
- To shatter these patterns into two classes $(+1/-1)$ given by the sequence $\delta_1, \dots, \delta_m;\ \ \delta_i \in \{0, 1\}$ it is sufficient to choose the value of the parameter

$$\alpha = \left(1 + \sum_{i=1}^{m} (1 - \delta_i)\ 10^i\right) \cdot \pi$$

- e.g., for $\delta_1 = 1, \delta_2 = 0, \delta_3 = 1,\ \alpha = \pi(1 + 0 \cdot 10^1 + 1 \cdot 10^2 + 0 \cdot 10^3) = 101 \cdot \pi$
- $\cos \alpha z_1 = \cos 10.1\, \pi \approx 0.9511 > 0$
- $\cos \alpha z_2 = \cos 1.01\, \pi \approx -0.9995 < 0$
- $\cos \alpha z_3 = \cos 0.101\, \pi \approx 0.9501 > 0$

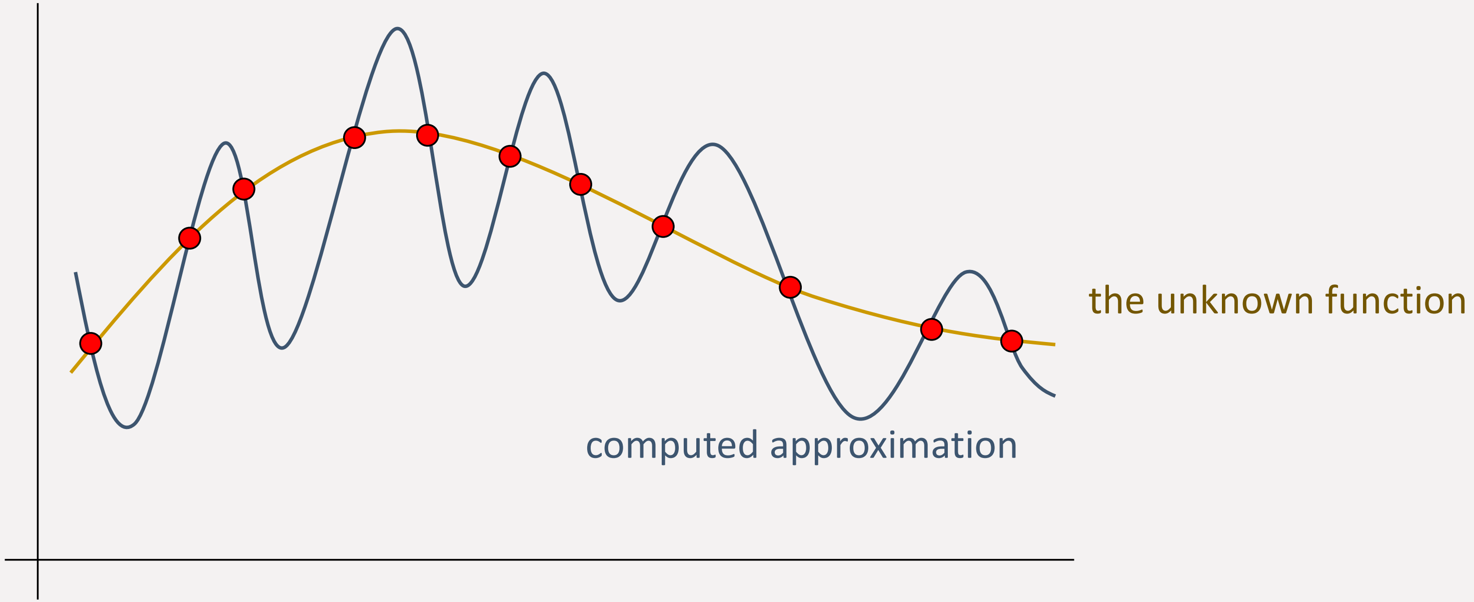
Vapnik-Chervonenkis Dimension (VC-dimension) (6)



when choosing a suitable coefficient α , it is possible to approximate any function bounded in the interval $\langle +1; -1 \rangle$ for any number m of selected points by $cos\, \alpha z$

Vapnik-Chervonenkis Dimension (VC-dimension) (7)

The problem of „overfitting“ ~ the network learns also the noise



Vapnik-Chervonenkis Dimension (VC-dimension) (8)

- For the network with W weights and N neurons and with the required limit for the generalization error ε, the number P of training patterns necessary for good generalization is:

$$P \geq \left(\frac{W}{\varepsilon}\right) log_2 \left(\frac{N}{\varepsilon}\right)$$

- A multi-layered network with 1 hidden layer cannot generalize well, if there were less than W/ε randomly chosen training patterns, i.e., $P \geq W/\varepsilon$
  - To achieve the accuracy of at least *90 %*, it is necessary to provide at least $10 \cdot W$ patterns

Neural Networks:

Contents:

- **Multi-layered Neural Networks**
- **Multi-layered Neural Networks: Analysis of Their Properties**
- **Multi-layered Neural Networks: an Application Example**

Contents: