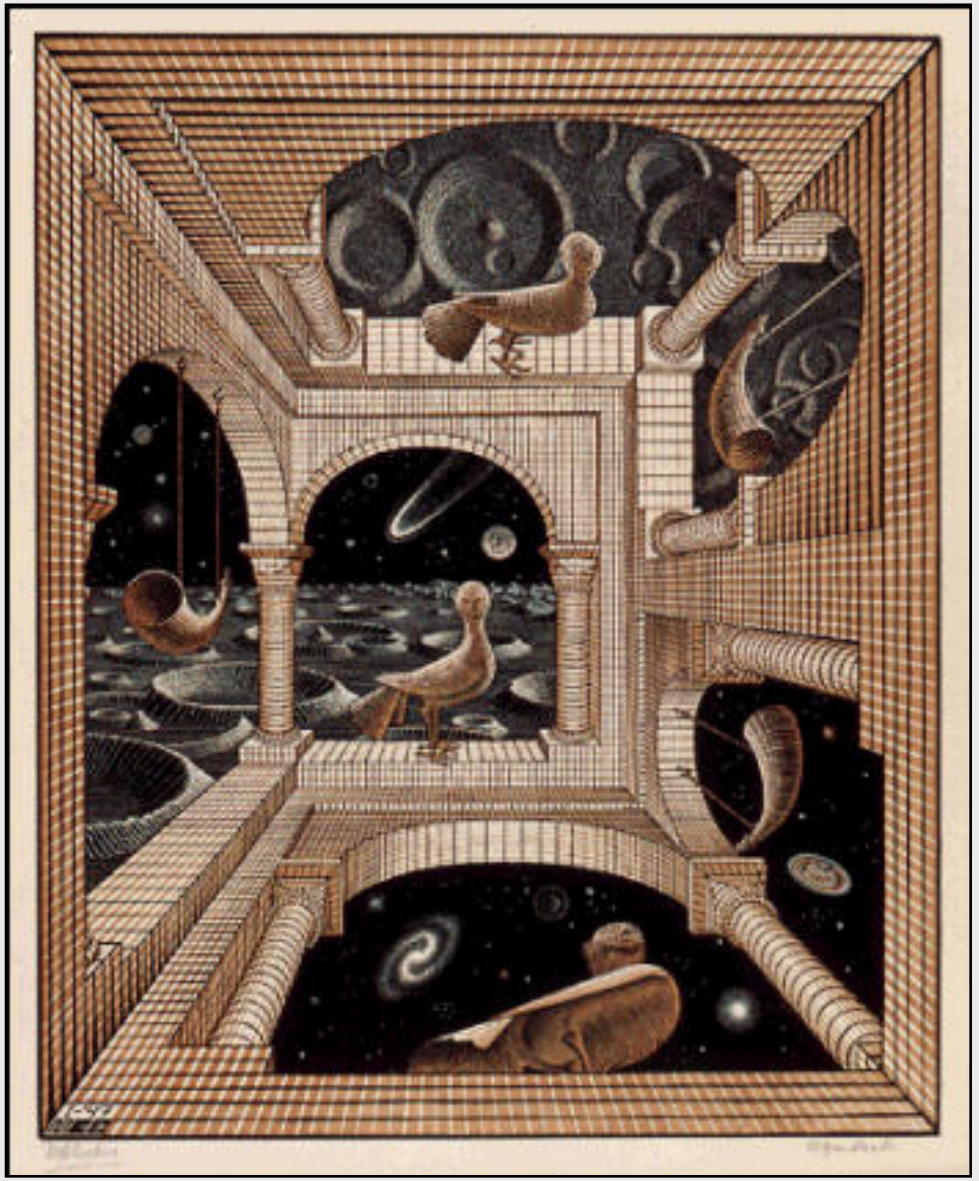
h_da

HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES



Computational Intelligence

Kapitel 3: Neuronale Netze

Teil B: Grundprinzipien

Dr. Norbert Waleschkowski

h_da Fachbereich Informatik

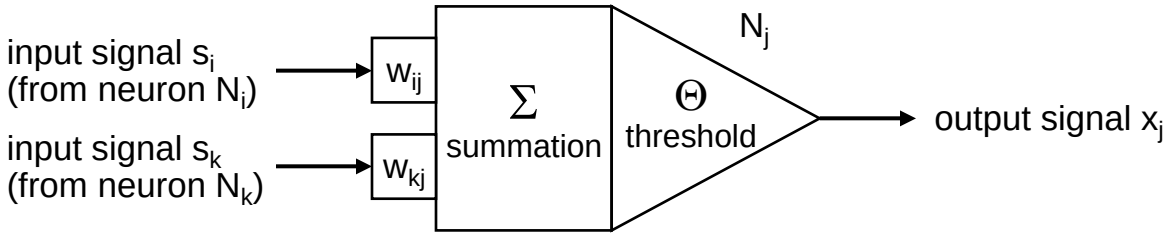
Sommersemester 2011
Master-Studiengang

Intro

- We will discuss artificial neural networks (ANN) from now on. For the sake of simplicity we will simply call them neural networks (NN).
- Neural Networks are very different with respect to their structure and behavior.
- There are many features and criteria to distinguish one network from another one.
- The most important features are:
  i. Artificial neurons
  ii. Activity functions
  iii. Output Functions
  iv. Network topologies
  v. Learning strategies
  vi. Energy functions
  vii. Local and distributed representation

Artificial Neurons (1)

- (Artificial) Neurons are the smallest units of a neural network. They can be connected with a great many of other neurons.
- A connection between two neurons N_i and N_j is called weight w_{ij}.
- In 1943 the American physicists McCulloch and Pitts introduced the so-called McCulloch-Pitts neurons. This was the first formal model of a neuron.



A McCulloch-Pitts neuron with two inputs

- A McCP-neuron has 2 input signals s_i and s_k and 2 corresponding weights w_{ij} resp. w_{kj}, which represent the synapses.
- All weights will have constant values $\in$ {-1, +1}. I.e. McCP-neurons are unable to learn.
- McCP neurons deliver output values $\in$ { 0, 1}, i.e. the neuron fires a signal of 1 or not (it “fires” a 0 resp.).

Artificial Neurons (2)

Remark:
We consider the following terminology for vectors as equivalent: **a** and $\vec{a}$

- The incoming signals are weighted and added up. The result is compared with a threshold Θ. If it exceeds the threshold Θ, it fires a 1.
- The input signals can be written as input vector **s** $=(s_1,s_2,\ldots, s_n)$. The corresponding weights can be written as a weight vector **w** $=(w_{1j},w_{2j},.., w_{nj})$
- Then the net input x (activity) can be expressed as the dot product of vectors **s** and **w** :

$$x = (\vec{s}^T, \vec{w}) = \sum_{i=1}^{n} s_i w_{ij}$$

- McCulloch and Pitts could prove that each logical function can be represented by a network of such neurons. In other words: Each Turing machine may be represented as a finite network of McCP neurons.

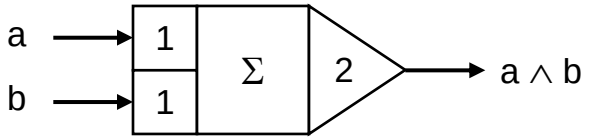
AND

a b	a ∧ b
0 0	0
0 1	0
1 0	0
1 1	1

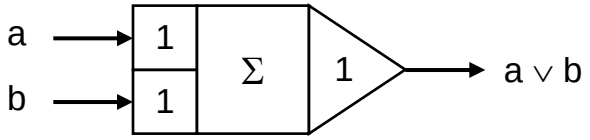
OR

a b	a ∨ b
0 0	0
0 1	1
1 0	1
1 1	1

- The logical AND function:

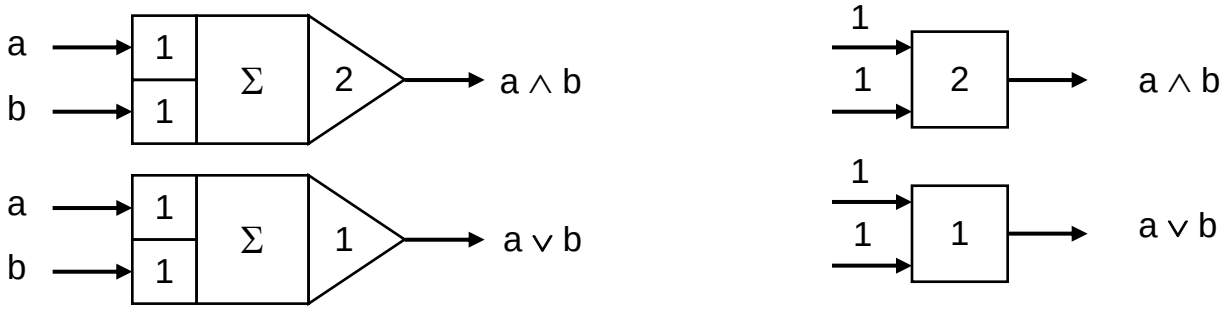


- The logical OR function:



Artificial Neurons (3)

- For the sake of simplicity we will represent neurons without loss of information in a simpler form:



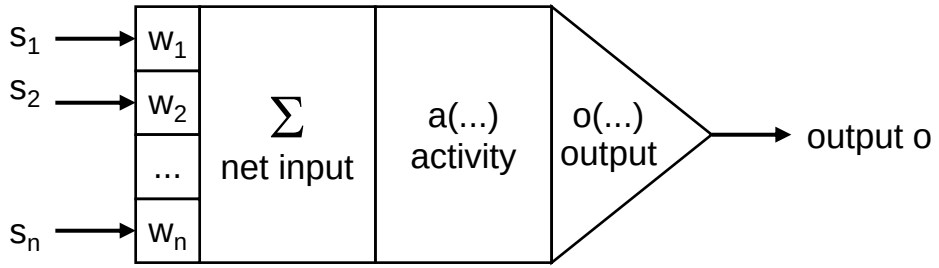
- The weights will be placed above the edges, the summation box is omitted.
- There are 16 logical resp. Boolean functions with 2 variables. Each of it can be represented by a McCP network.

a b	a ∧ b	a ∨ b	XOR	...	always false	always true
0 0	0	0	0		0	1
0 1	0	1	1		0	1
1 0	0	1	1		0	1
1 1	1	1	0		0	1

- The same applies for logical and Boolean functions, resp., with n variables.

Artificial Neurons (4)

- Disadvantage of McCP neurons: They are unable to learn, because their weights are fixed.
- Today adaptable neurons with variable weights are used. Normally the weights are real numbers from the interval [-2, +2]. Sometimes it may be useful to use fixed weights.
- An artificial neuron has the following structure:



- After adding up the weighted input signals, an activity function determines the activity of the neuron. This function is a monotonous function. Normally it has the net input as its argument.
- The activity is a measure for the excitation of a neuron. An output function determines the value the neuron fires at last.

Artificial Neurons (5)

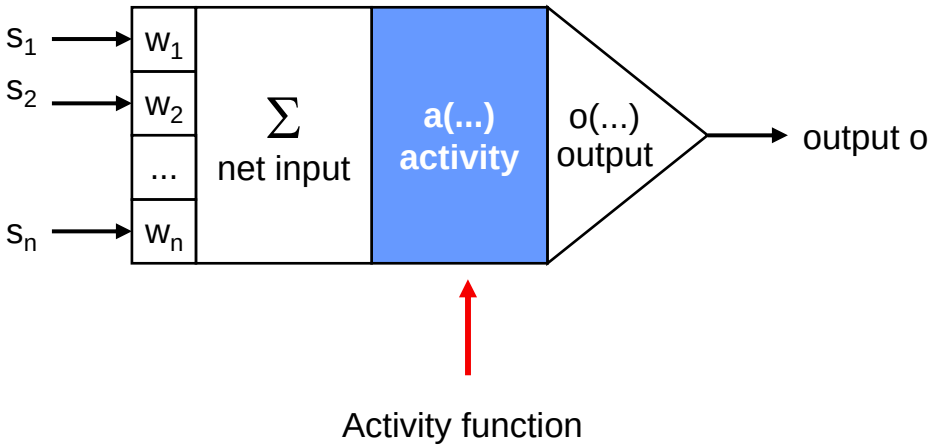
- All neurons share the property to first carry out some kind of a pattern matching operation. For a neuron N_j the following applies:

$$(\vec{s}^T, \vec{w}_j) = \sum_{i=1}^{n} s_i w_{ij} = \left\|\vec{s}\right\| \cdot \left\|\vec{w}_j\right\| \cos(\vec{s}, \vec{w}_j)$$

- This expression is nothing more than the scalar (dot) product of input and weight vectors. The result is called net-input net_j of neuron N_j.
- The higher the „similarity“ of input and weight vector, the higher the degree of excitation. Orthogonal vectors have a scalar product of 0. In this case there is no „similarity“.

Activity Functions (1)

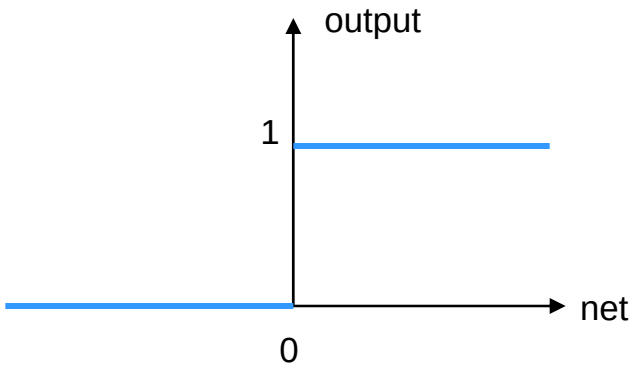
- Consider a neuron N_j.
- An activity function a(…) determines the degree of the activity of a neuron.
- The argument of the activity function is the net input: $a = a(net_j)$
- The simplest activity function is the identity. In this case the activity of the neuron is the same as the net input net_j.
- Activity functions are always monotonous functions. A function is called a monotonous function if $x < y \Rightarrow f(x) \leq f(y)$.
- A function is called a strongly monotonous function if $x < y \Rightarrow f(x) < f(y)$.



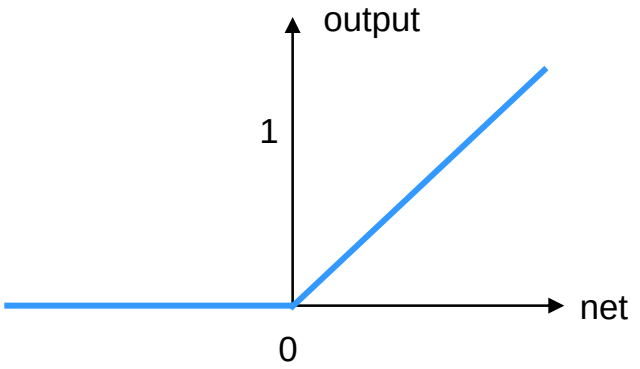
Activity Functions (2)

Examples of activity functions:

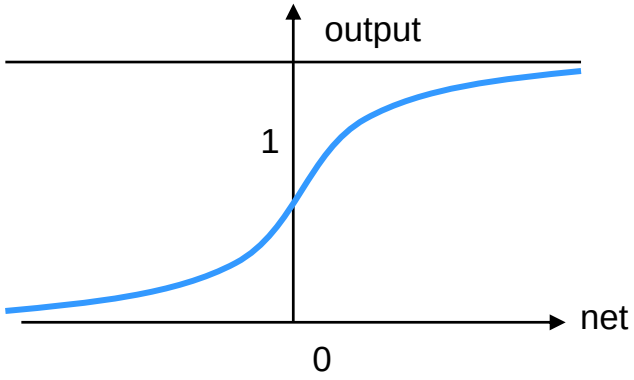
- Hard delimiter



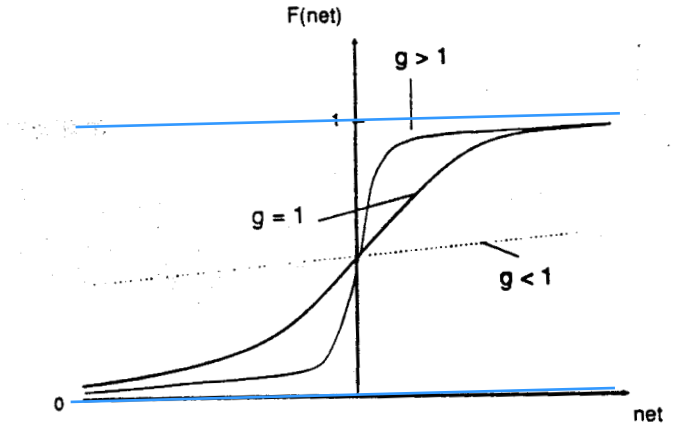
- Threshold function



- Sigmoid function



Sigmoid Functions (1)



Properties of sigmoid functions:

$$s(x) > 0$$

$$0 \le s(x) \le 1$$

$$x \to \pm\infty \Rightarrow s(x) \to 1 \; bzw. \; 0$$

$$s(x) \; is \; \infty \; times \; differentiable$$

The Fermi function as an example for sigmoid functions:

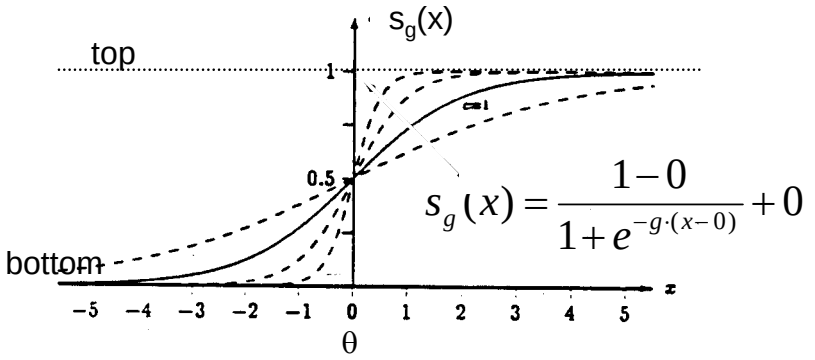
$$s(x) = \frac{1}{1+e^{-gx}}$$

The gain parameter g determines the steepness of the function. For $g \to \infty$ this function converges against the hard delimiter function.

The general formula is

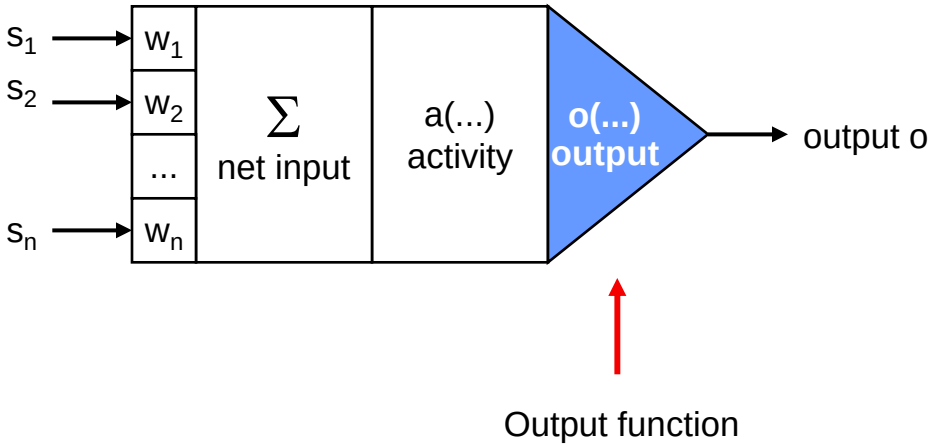
$$s_g(x) = \frac{top - bottom}{1+e^{-g\cdot(x-\theta)}} + bottom$$

where
„top“ is the upper asymptotis (here =1)
„bottom“ is the lower asymptotis (here =0)
θ is the position of the reversal point (here θ = 0).



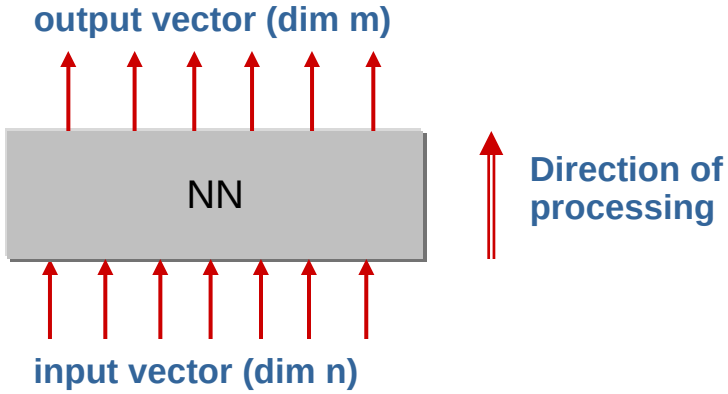
Output Functions

- The output function o(…) determines the value which will be transferred to all connected neurons.
- The argument of the output function is (inter alia) the activity a.
- Calculating and setting the output value to > 0 means “to fire”.
- The simplest output function is the identity. In this case the activity of the neuron is the same as the net input net_j.
- Often the output function is a hard delimiter function.
- The sending neuron is often called pre-synaptic neuron. The accepting neuron is often called post-synaptic neuron.

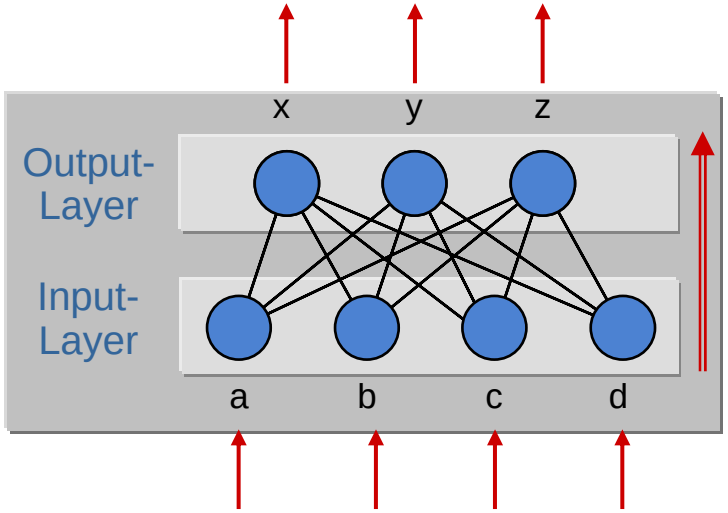


Network Topologies (1)

- There is a great variety of network topologies. The type of connectivity determines to a high degree the knowledge and behaviour of a NN.
- A NN can be considered as a black box. It has n > 1 inputs und m ≥ 1 outputs. Internally it performs some non-linear vector transformation.

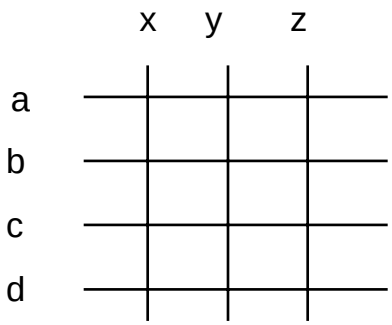


- Often neurons are grouped to a layer. Within one layer the neurons are not interconnected. Neigh-bouring layers are completely interconnected.
- See a 2-layered NN at the right hand side.



Network Topologies (2)

- The arrow at the right hand side of the NN indicates the direction of processing. The incoming information is processed and forwarded to the next layer. This is called feed-forward propagation of information.
- As an alternative the 2-layered NN could also represented as a matrix.

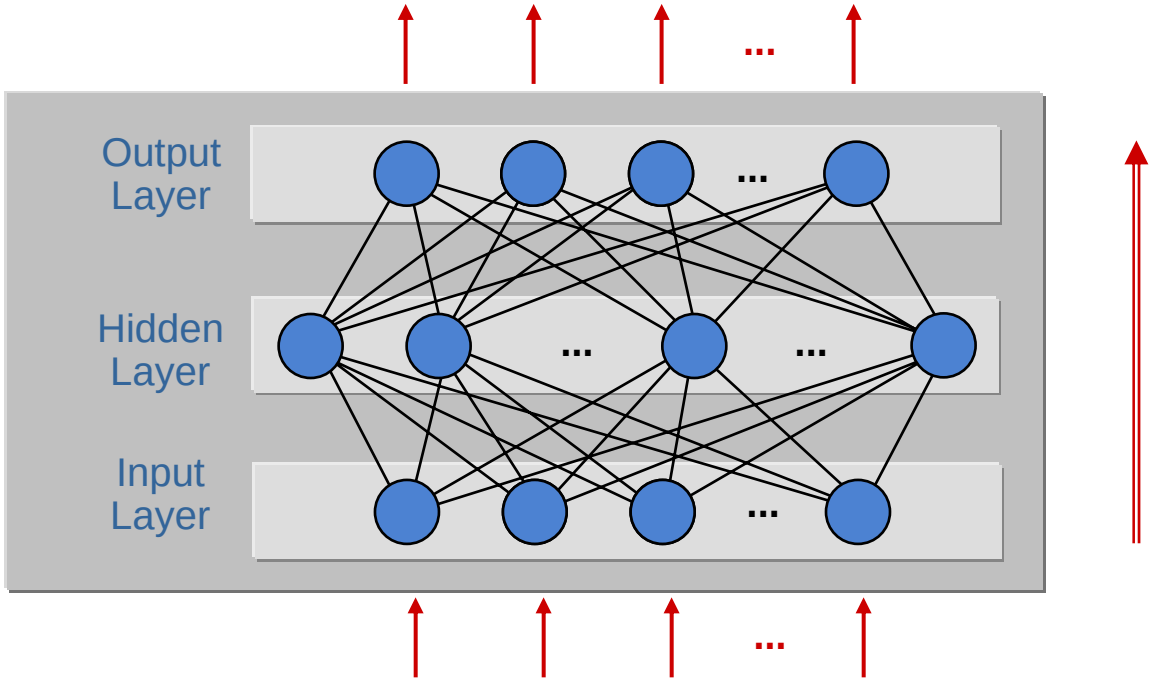


$$\begin{pmatrix} ax & bx & cx & dx \\ ay & by & cy & dy \\ az & bz & cz & dz \end{pmatrix}$$

- The behaviour of a 2-layered NN is often called an association.

Network Topologies (3)

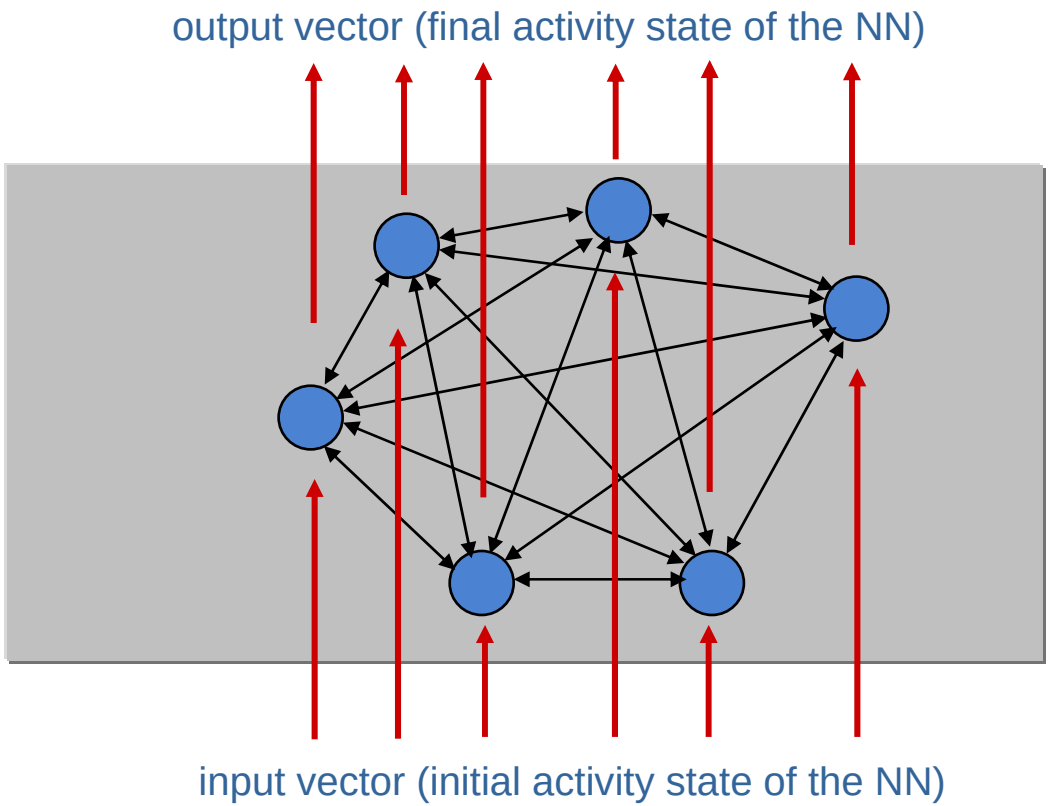
- Network topologies can be much more complex.
- Here is a multi-layered feed-forward NN consisting of 3 layers.



- All layers between input and output layer are called hidden layers, because they are not visible to a user.

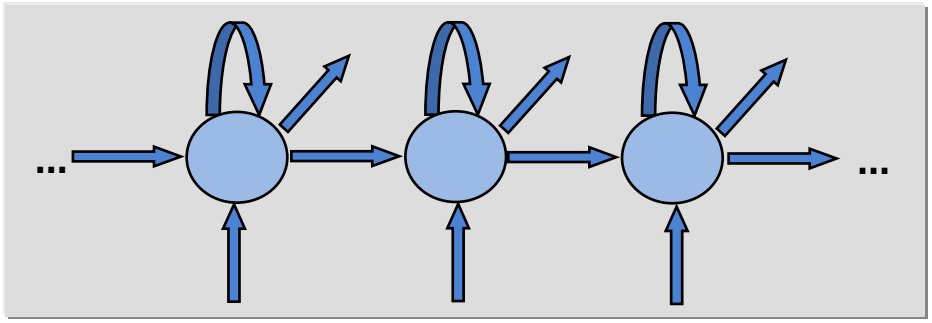
Network Topologies (4)

- Network topologies can be much more complex. Here are some examples.
- A NN with neurons that are completely interconnected:



Network Topologies (5)

- In this example the neurons are coupled laterally.



- Definition of the Activity State of a NN: The activity state of a NN ist the activity vector of the activities of all neurons.

$$\vec{a}(t) = (a_1(t), a_2(t), ..., a_n(t)), \quad a_i \in \Re$$

Learning Strategies (1)

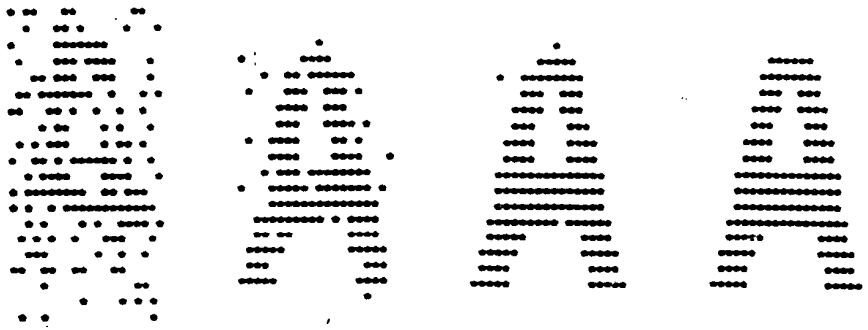
- Biological neural systems are not born pre-programmed with all the knowledge and abilities that they eventually have. A learning process that takes place over a period of time somehow modifies the network to incorporate new information.
- But how does this learning process work? How can we make a NN learn? In other words: How do we program a NN?
- The complete knowledge of a NN is contained in its topology, or to be more precise in the weights of a NN. The complete weight set deter-mines the complete behaviour of a NN.
- Weights could be set manually or automatically. Setting weights manually is a very difficult and time-consuming task. In case of distributed representation of knowledge it is almost impossible.
- Setting weights automatically seems to be much more appropriate.
- Before starting the discussion of learning , let's have a look at the phases of a NN. Typically a NN has a learning phase and a recall phase. During the learning phase the NN learns from a training collective, i.e. from a set of training vectors. After learning has been completed the NN can go to work in the recall and the working phase, resp.

Phase 1:
Learning resp.
Training Phase

Phase 2:
Recall resp.
Working Phase

Learning Strategies (2)

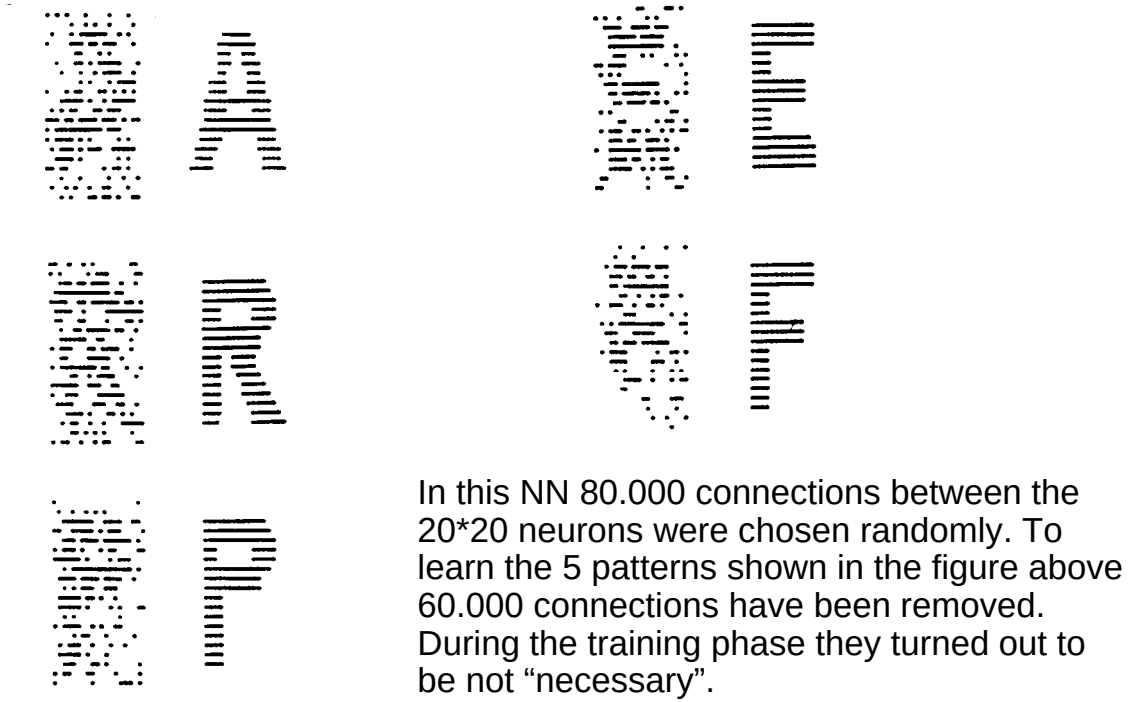
- Neural learning means to modify the weights of the neurons involved.
- Normally there is no need to create new connections in an ANN. A no-connection is normally modelled by a connection with weight zero. Changing the zero-weight to a non-zero value has the same effect as creating a new connection.
- Examples of learning:



- This neural network works with 20 * 20 neurons placed regularly within a rectangle. A star represents an active neuron. (Within this NN valid activity values are “0” and “1”.) Non-active neurons as well as the connections of the network are not visible here. This NN has learned 30 pattern of letters. Here it recognizes the polluted “A” as an “A” within only 3 steps.

Learning Strategies (3)

- Another example:



In this NN 80.000 connections between the 20*20 neurons were chosen randomly. To learn the 5 patterns shown in the figure above 60.000 connections have been removed. During the training phase they turned out to be not “necessary”.

Learning Strategies (4)

- There are many different learning strategies. A systematical overview is difficult. One criteria is the existence of a teacher instance.

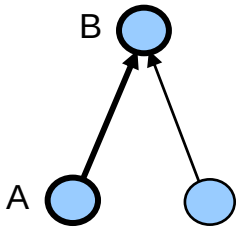
  Learning with or without a teacher (i.e. a teaching instance)

- Supervised learning
  The presentation of a training vector leads to a specific result o. In parallel a teacher signal z is provided, which represents the required result. The teacher signal specifies the required output value per neuron within the output layer in detail. The difference d = z – o is called error.

- Reinforcement learning
  The teacher instance provides the NN with a scalar as a measure for the quality of the result, but not with the required output value in detail.

- Unsupervised learning
  Here there is no teacher instance and very often also no learning phase. Weight modifications are due to the correlation of pre- and postsynaptic excitement.

Learning Strategies (5)



Hebbian Learning

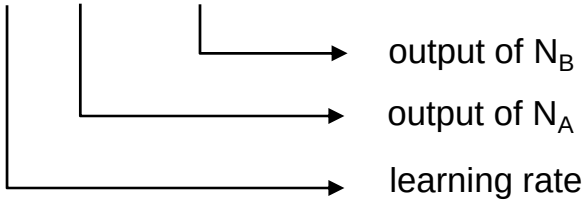
- The basic principles of neural learning have been detected by Donald Hebb (1949):

  "When an axon of cell A is near enough to excite a cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that A's efficiency , as one of the cells firing B, is increased." (Donald D. Hebb. The Organization of Behaviour. Wiley, New York 1949, p. 50)

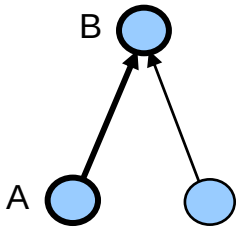
- This has lead to a learning theorem which is known as Hebb's Learning rule:

$$w_{AB}(t+1) = w_{AB}(t) + \Delta w_{AB}$$

$$\text{where} \quad \Delta w_{AB} = c \cdot o_A(t) \cdot o_B(t)$$



Learning Strategies (6)

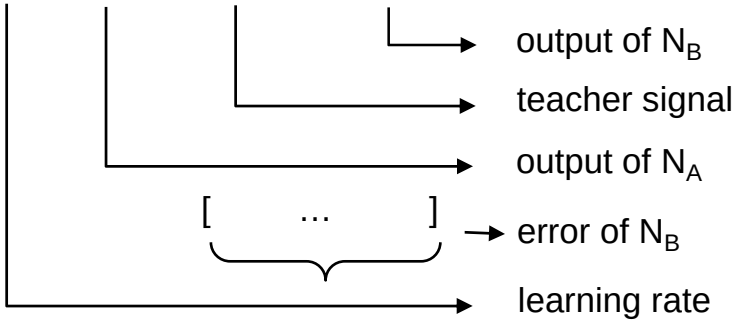


Delta Learning

- The so-called delta rule is a variation of Hebb's rule.
- The delta rule leads to a convergence of the weights.

$$w_{AB}(t+1) = w_{AB}(t) + \Delta w_{AB}$$

$$\text{where } \Delta w_{AB} = c \cdot o_A(t) \cdot [\vec{z}(t) - o_B(t)]$$

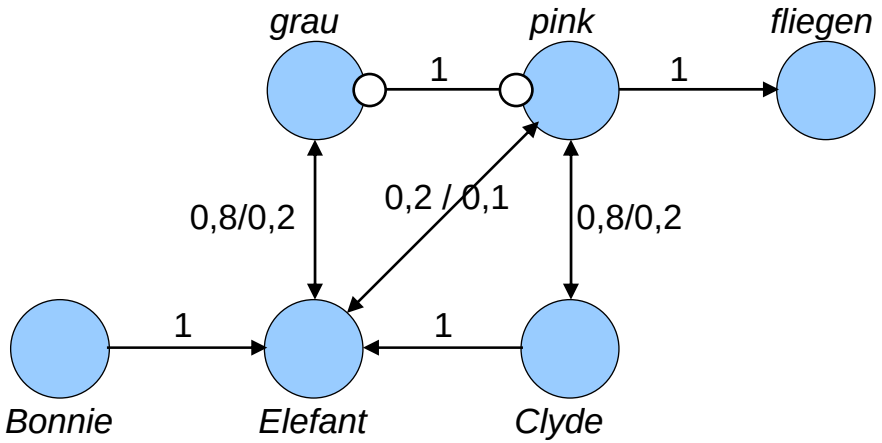


Energy Functions

- There are many network types which are primarily motivated by physical systems.
- In physical systems the energy plays a central role. The term “energy” can also be transferred to neural networks.
- The energy function is a function which describes the global state of the excitement of a NN and can be used to describe its dynamics.
- Examples for such network types are
  - Hopfield networks (motivated by so-called spin glasses)
  - Boltzmann machines (motivated by thermodynamics)
- For these network types the term “energy” has an essential meaning to describe its overall behaviour.

Local and Distributed Representation (1)

- To assess whether a NN works properly, it is important to “understand” the input and output signals .
- Using conventional computer systems this is done via symbols, strings and numbers.
- Using NNs there are only groups of neurons that have activities. How to interpret such an activity pattern?
- There are 2 types of representation:
  - local representation
  - distributed representation



- Local representation:
  Each concept to be represented is associated with exactly one neuron.

Local and Distributed Representation (2)

- Example:
  - Bonnie is an elephant
  - Clyde is probably a pink elephant.
  - Elephants are mostly gray, sometimes pink.
  - Everything that is pink is able to fly.
- Let's assume that the activity function is the identity function and the output function is a hard delimiter function.
- If the neuron which represents Bonnie is activated and the network would run for a few cycles, the result could be interpreted as follows.

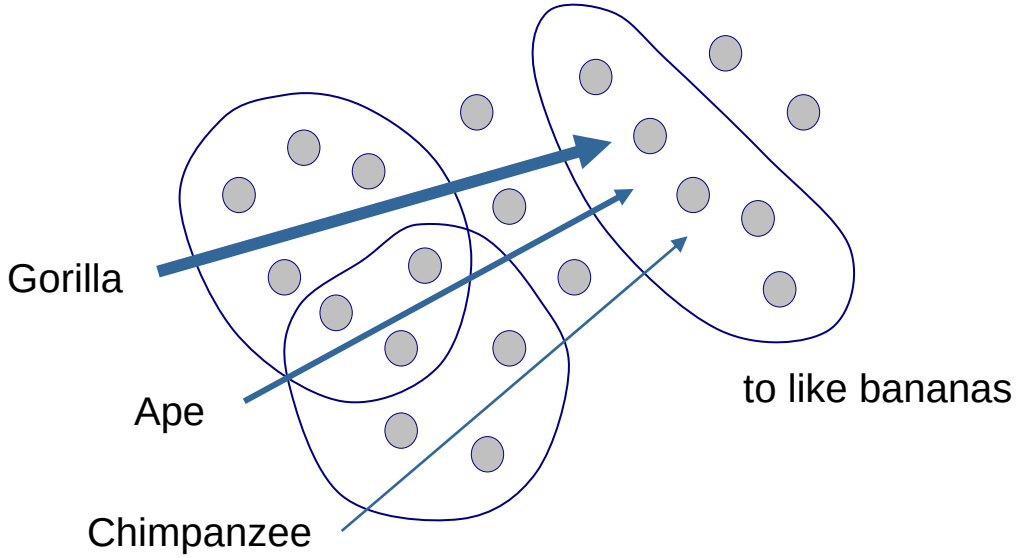
Bonnie	1
Clyde	0
Elephant	1
Gray	0,82
Flies	0,11

} i.e., Bonnie is probably a gray elephant and is probably not able to fly.

- If the Clyde neuron would be activiated it would lead to a pattern that describes the characteristics of Clyde.
- Result: A local representation of concepts leads to a pattern with a strong symbolic characteristics.

Local and Distributed Representation (3)

- If multiple neurons contribute to the representation of a concept we have a distributed representation. I.e., a single neuron does not represent a concept on its own.
- Advantages:
  - Higher capacity of a NN
  - Spontaneous generalization
  - Recognition of similarities
- In the NN given below four concepts are represented: gorilla, ape, chimpanzee, to-like-bananas



Local and Distributed Representation (4)

- The concepts “gorilla” and “chimpanzee” overlap.
- The concept “ape” could be interpreted as a superior concept.
- If the connections between “gorilla” and “to-like-bananes” are strengthened the connections between “ape” and “to-like-bananas” as well as between “chimpanzee” and “to-like-bananas” are strengthened, too.
- Further properties of a distributed representation:
  - More robust against perturbations
  - A single neuron of a concept may be interpreted as a micro-feature of that concept
  - Normally there is no need to introduce new neurons if a new concept has to be represented
- Balancing reasons of pros and cons is often difficult. A decision has to be made depending on the problem to be solved.
- Sometimes a mixture of both types of representation is useful.

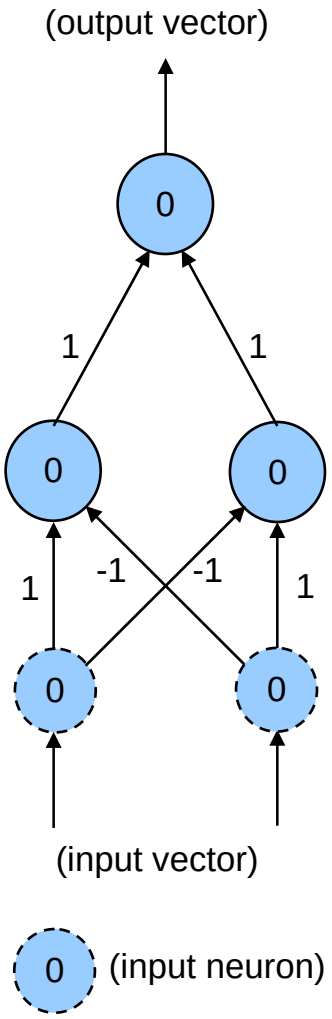
Example: The XOR-Problem (1)

- The logical XOR function represents the “exclusive OR”.

XOR-Problem (exclusive OR)

X	Y	XOR
0	0	0
0	1	1
1	0	1
1	1	0

- How to solve the XOR problem with a NN?
- There are different solutions:
- The solution at the right hand side has 5 neurons organized in 3 layers.
- A neuron fires if the threshold is exceeded, i.e. if activity is > 0.
- A neuron fires a 1 (resp. a 0 signal).



Example: The XOR-Problem (2)

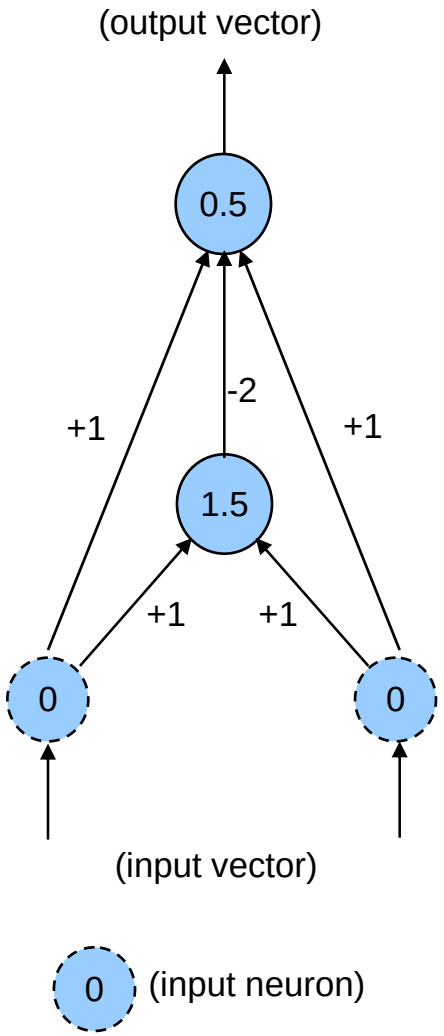
- This solution has only 4 neurons. The neurons are not organized within layers. The topology is more complex than in the layered solution.
- Each neuron fires if its individual threshold is exceeded:

$$o_i(t) = \begin{cases} 1 \Leftrightarrow \sum_j w_{ij} x_j > \Theta \\ 0 \quad otherwise \end{cases}$$

- The activity function is the identity function:

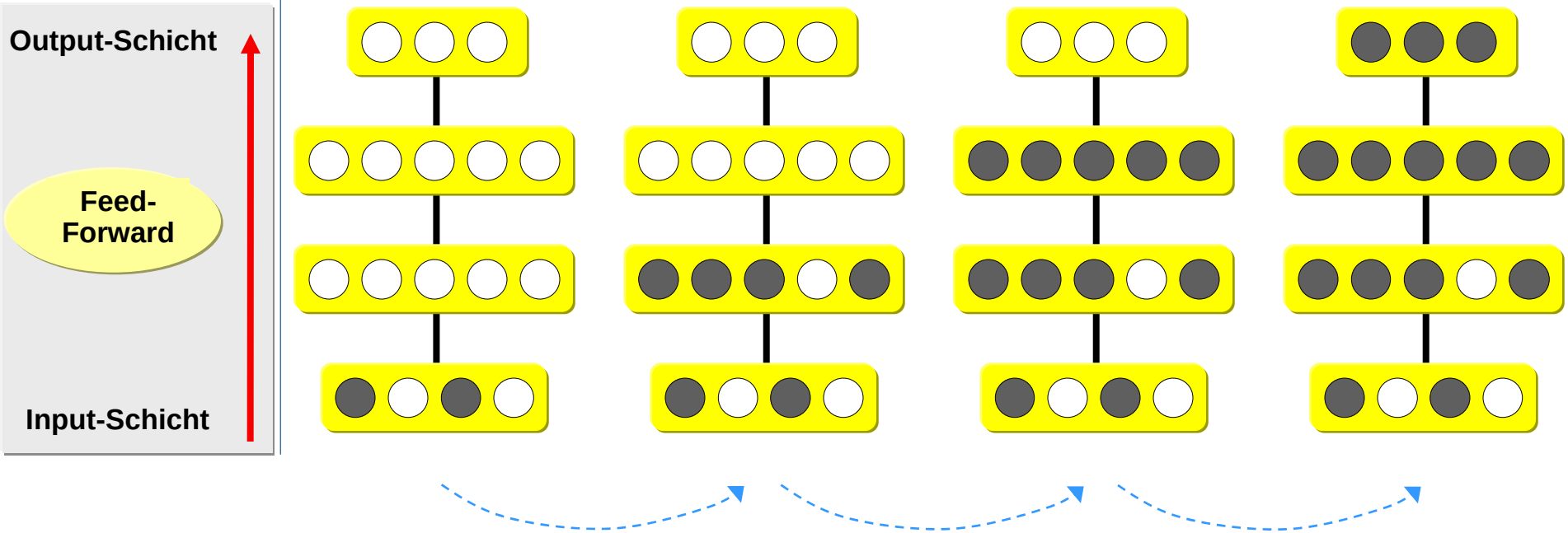
$$a_i(net_i) = id(net_i) = net_i$$

- There is no chance to solve the XOR problem with 3 neurons only !



Das Multi-Layer-Perzeptron & Backpropagation-Algorithmus (1)

- Mit dem BP-Algorithmus können auch die verdeckten Schichten eines Multi Layer-Perzeptrons lernen.
- Dabei wird nicht nur der Output-Schicht, sondern auch den verdeckten Schichten ein Fehler zugeordnet.
- Schritt 1: Das zu trainierende Muster wird angelegt.
- Die Information wird nach dem Feed-Forward-Prinzip durch die Netzschichten propagiert. Es entsteht der Ist-Output.



Das Multi-Layer-Perzeptron & Backpropagation-Algorithmus (2)

Fehler am Output-Layer:

$$\delta_{li} = (z_{li} - a_{li}) \cdot \sigma'(net_{li})$$

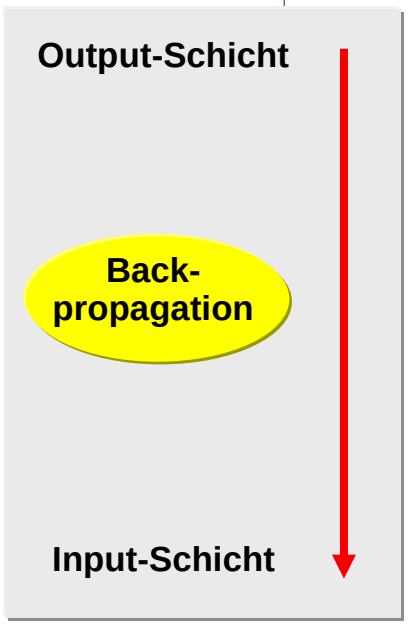
Fehler Hidden Layer:

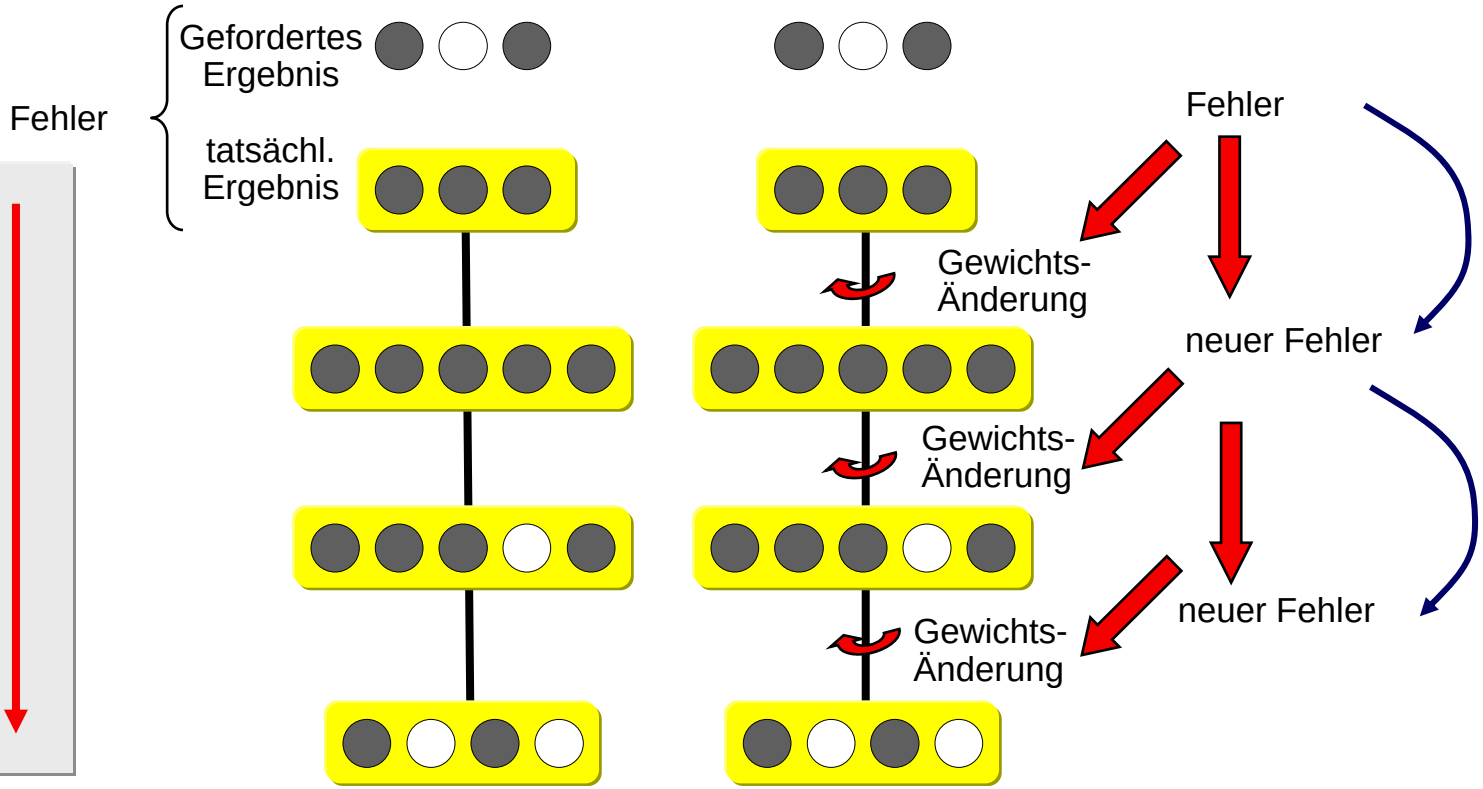
$$\delta_{li} = \sigma'(net_{li}) \cdot \sum_k \delta_{lk} w_{ki}$$

Gewichtskorrektur per Lernmuster l:

$$\Delta_l w_{ij} = c \cdot \delta_{li} \cdot a_{lj}$$

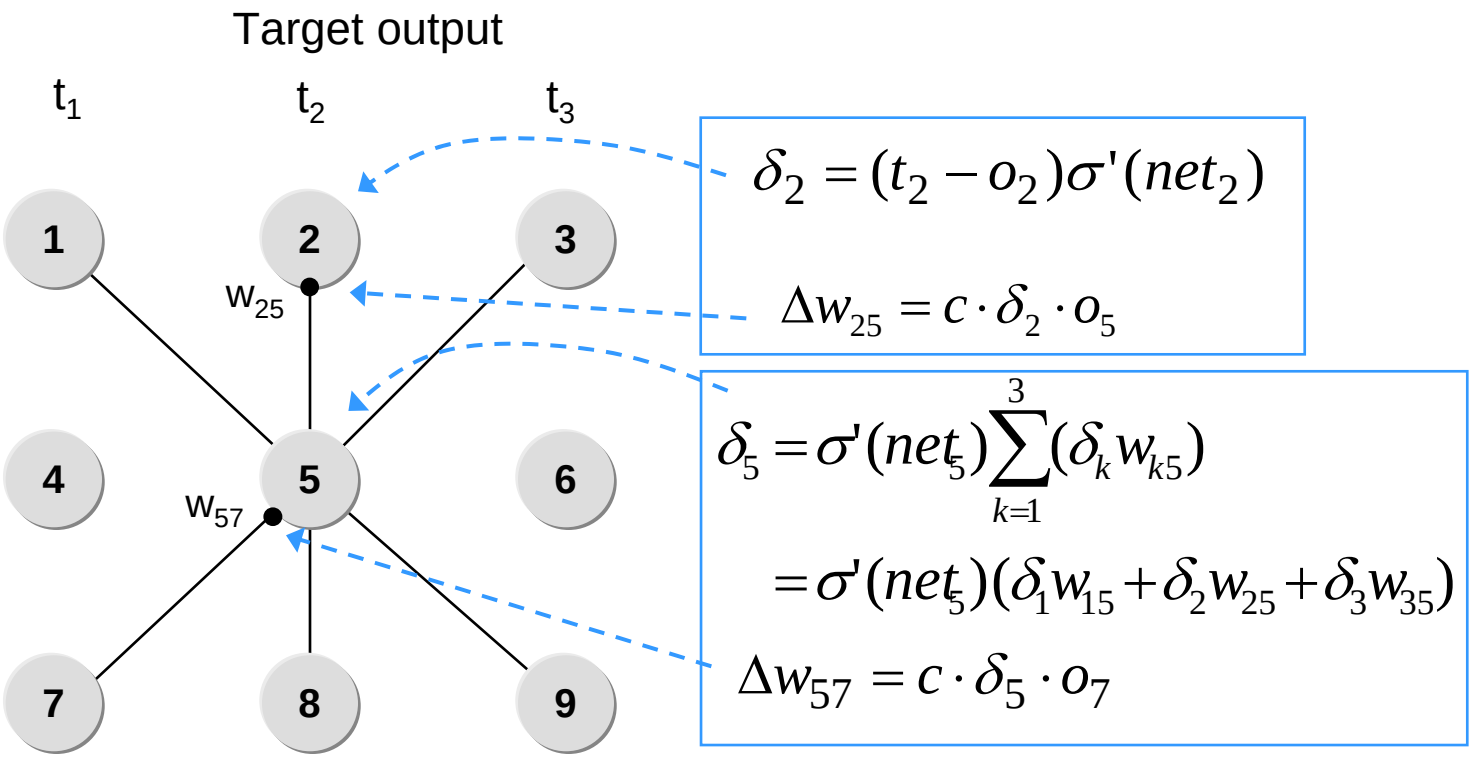
- Schritt 2: Die Größe des Fehlers fließt in die Gewichtsänderung mit ein.
- Die Fehlerinformation wird in die Netzschichten zurückpropagiert.
- Der Fehler für die verdeckten Schichten wird aus den Fehlern der übergeordneten Schichten abgeleitet.





Das Multi-Layer-Perzeptron & Backpropagation-Algorithmus (3)

Beispiele:



Für eine genauere Beschreibung siehe → *„VCI_07_Exkurs_NN_MLP_Backprop.pdf* “

Eine Beispiel-Anwendung

NETtalk (Erzeugung gesprochener Sprache, 1986)

- 1986 stellten Sejnowski & Rosenberg ein neuronales Netzwerk vor, das geschriebenen englischen Text mit der Qualität eines ca. 8 Jahre alten Schülers vorlesen konnte.
- NetTalk war ein 3 Layer Backpropagation-Perceptron mit ca. 300 Neuronen.
- Der größte Teil des geschriebenen Textes wurde korrekt ausgesprochen.

Wie spricht man das Kunstwort

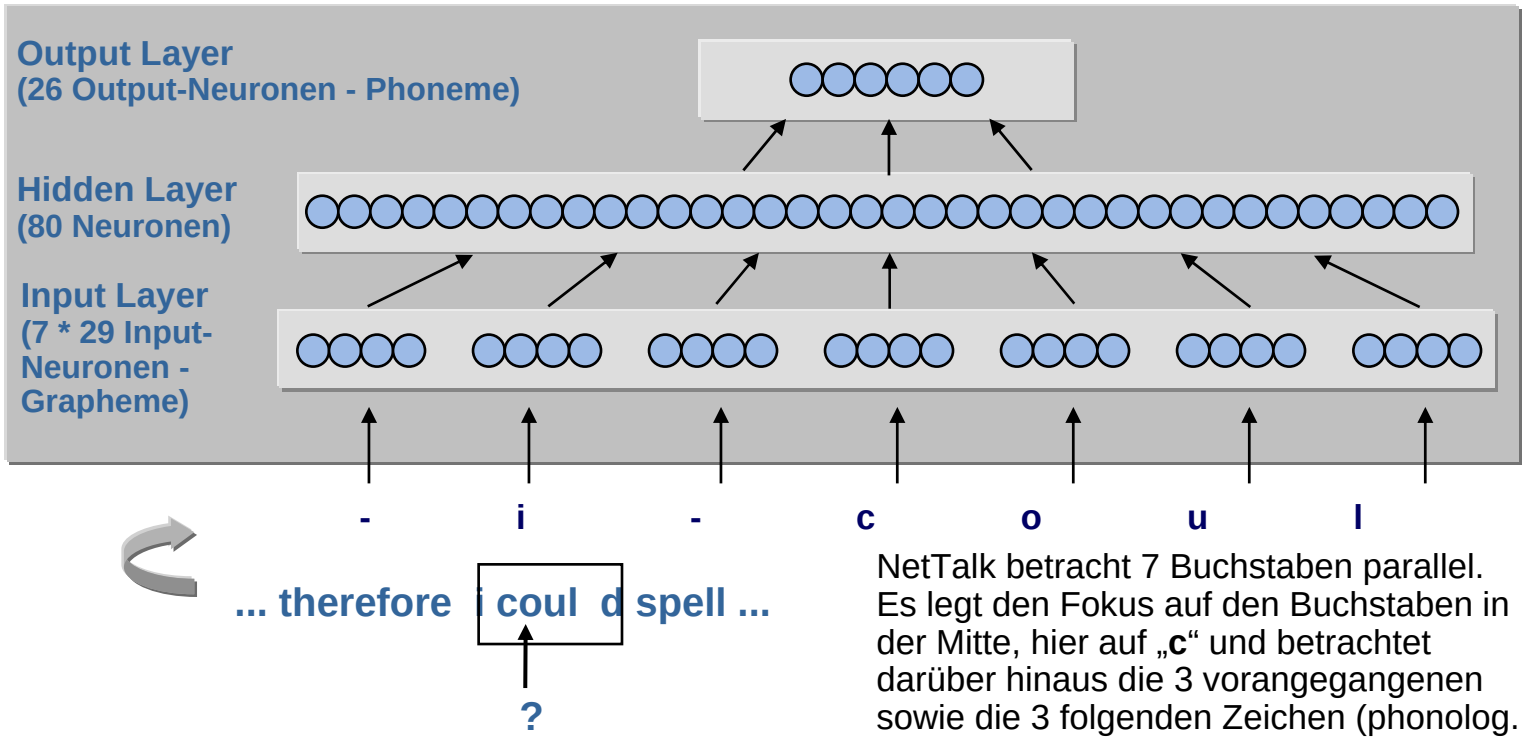
ghoti?

aus?

Bei Verwendung einer nicht-repräsentativen Trainingsmenge führt es zur Aus-sprache

fish

z.B.abgeleitet aus Wörtern wie
- enough
- women
- nation



NetTalk betracht 7 Buchstaben parallel. Es legt den Fokus auf den Buchstaben in der Mitte, hier auf „**c**“ und betrachtet darüber hinaus die 3 vorangegangenen sowie die 3 folgenden Zeichen (phonolog. Kontext).

Typische Anwendungsklassen

- Es gibt zahlreiche Problemklassen, bei denen neuronale Netze angewendet werden können:
  - Iris-Erkennung (biometrische Methode zur Identifikation von Personen)
  - Bilanzanalyse
  - Akustische Qualitätskontrolle
  - Datenkompression
  - Prozesskontrolle
  - Mustererkennung
  - Bild- und Szenenanalyse
  - Spracherkennung
  - Silicon Retina (Elektronisches Auge)
  - Silicon Ear (Elektronisches Ohr)
  - Optimierung
  - Sichtgestützte Inspektion
  - Autonome Roboter
  - etc.

Aufgabe:

Ein Netz bestehe aus 20 Input- und 2 Output-Neuronen N1 und N2. Es soll dazu dienen, die Buchstaben H und O zu erkennen. Die Gewichte w_{1j} seien, korrespondierend zur Abb. H = 1, falls Pixel schwarz, und = 0, falls Pixel weiß. Entsprechendes gelte für die Gewichte w_{2j} zu Abb. O. a) Wie lauten die beiden Gewichtsvektoren?

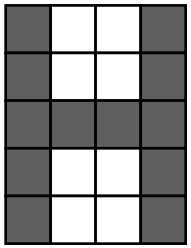


Abb. H

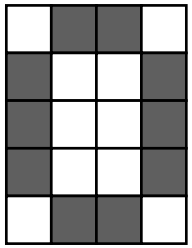


Abb. O

a) Die Bilder werden zeilenweise gelesen. Dann gilt

w_H = (1 0 0 1 | 1 0 0 1 | 1 1 1 1 | 1 0 0 1 | 1 0 0 1)

w_O = (0 1 1 0 | 1 0 0 1 | 1 0 0 1 | 1 0 0 1 | 0 1 1 0)

b) N1 liefert als Output 1, falls die Aktivierung ≥10 und 0 sonst. N2 liefert als Output 1, falls die Aktivierung ≥8 ist, und 0 sonst. Bei Anlegen von „H“ wird (1,0) geliefert, bei Anlegen von „O“ wird (0,1) geliefert.
Was wird erkannt, wenn folgende Bilder angelegt werden?

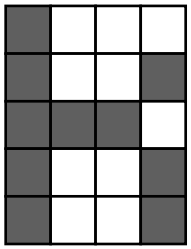


Bild 1

$B1*w_H = 10 \Rightarrow o(N1) = 1$

$B1*w_O = 6 \Rightarrow o(N2) = 0$

$\Rightarrow$ Erg = (1, 0) Es wird „**H**“ erkannt.

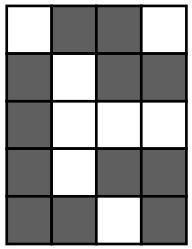


Bild 2

$B2*w_H = 7 \Rightarrow o(N1) = 0$

$B2*w_O = 8 \Rightarrow o(N2) = 1$

$\Rightarrow$ Erg = (0, 1) Es wird „**O**“ erkannt.

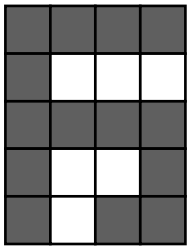


Bild 3

$B3*w_H = 11 \Rightarrow o(N1) = 1$

$B1*w_A = 8 \Rightarrow o(N2) = 1$

$\Rightarrow$ Erg = (1,1) Es wird **nichts** erkannt.