



**UNIVERSITÀ
DEGLI STUDI
DI FOGGIA**



HR EXCELLENCE IN RESEARCH

Artificial Neural Networks

Basic concepts

Prof. Crescenzo Gallo

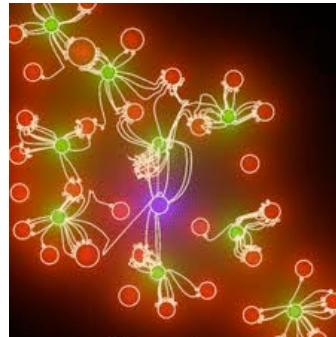
Aggregate Professor of Computer Science and Processing Systems

Department of Clinical and Experimental Medicine

crescenzo.gallo@unifg.it



Neural Networks



A neural network is an artificial representation of the human brain that attempts to simulate the learning process.

The term "artificial" means that neural networks are implemented in computer programs that are able to handle the large number of calculations required during the learning process.

To show where neural networks have their origin, let's take a look at the biological model: the human brain.

Neural Networks

The biological model: the human brain



The human brain is composed of a large number (~100 billion) of neural cells that process information.

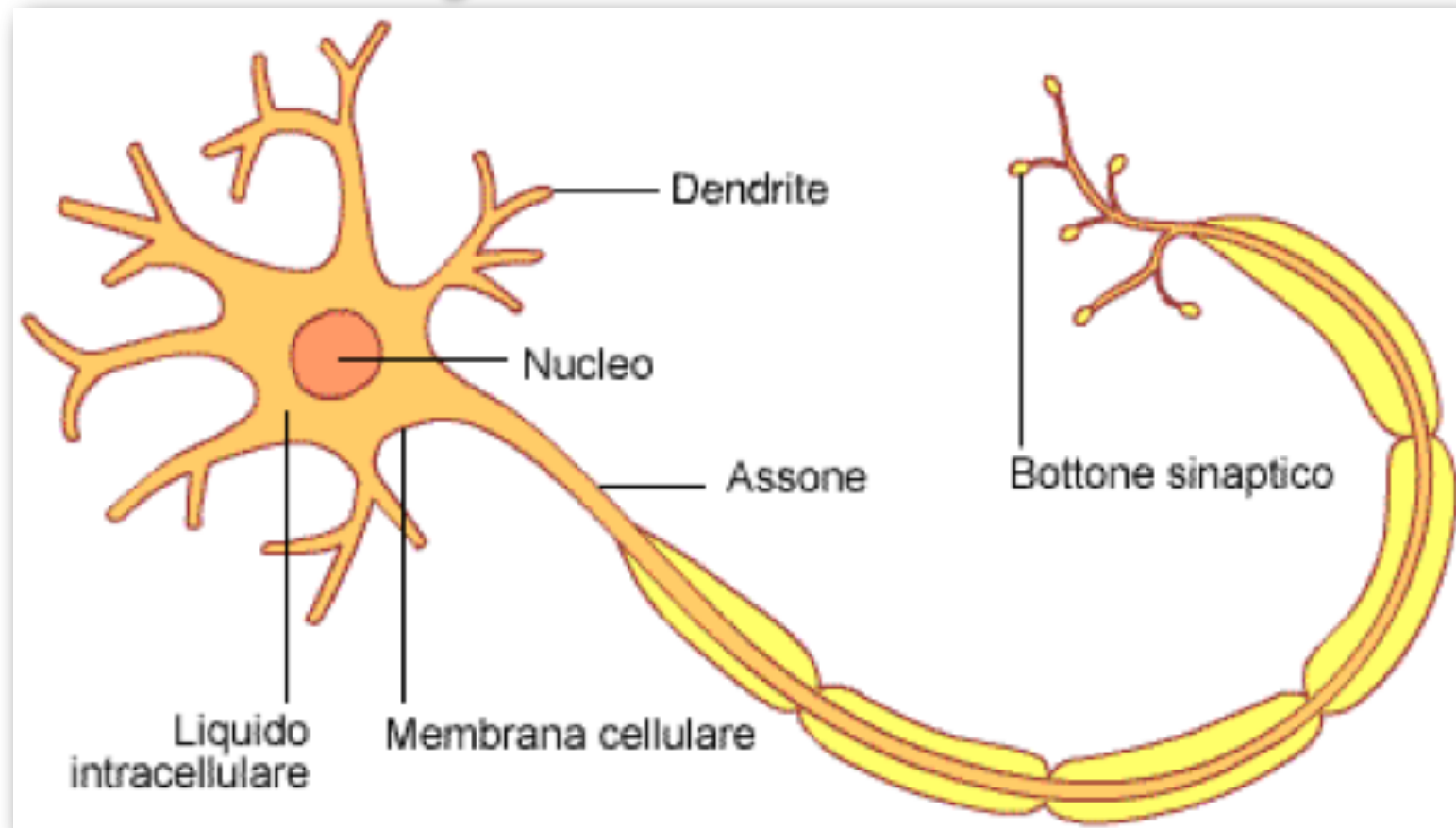
Each cell functions as a simple processor and only the massive interaction between all cells(*) and their processing in parallel makes brain capabilities possible.

Here is a diagram of such a neural cell, called a **neuron**:

(*) *Each cell develops, on average, about 10,000 connections with neighboring cells.*

Neural Networks

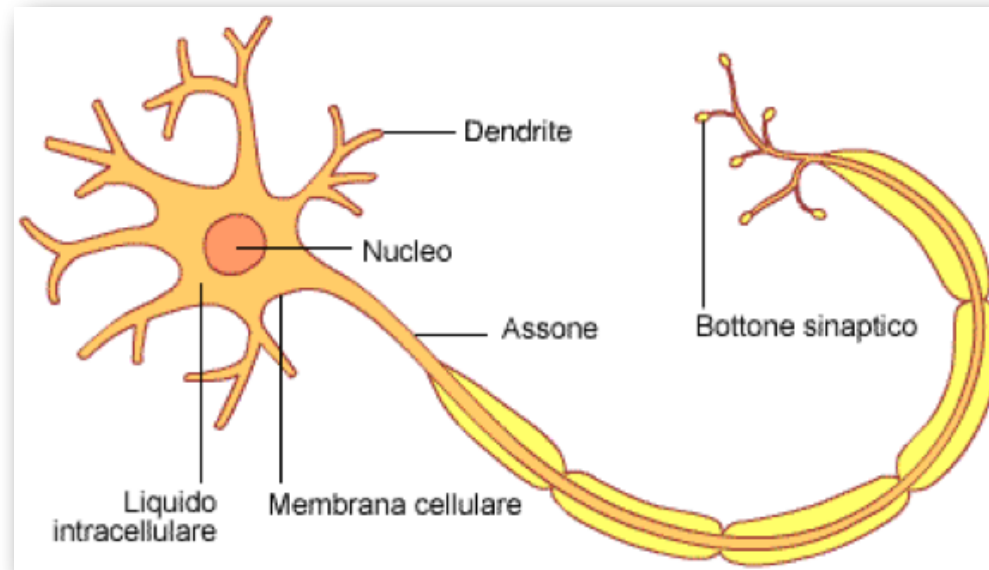
The biological model: the human brain



As the figure indicates, a neuron consists of a **nucleus**, **dendrites** for incoming information, and an **axon** with dendrites for outgoing information that is passed to connected neurons (**synapses**).

Neural Networks

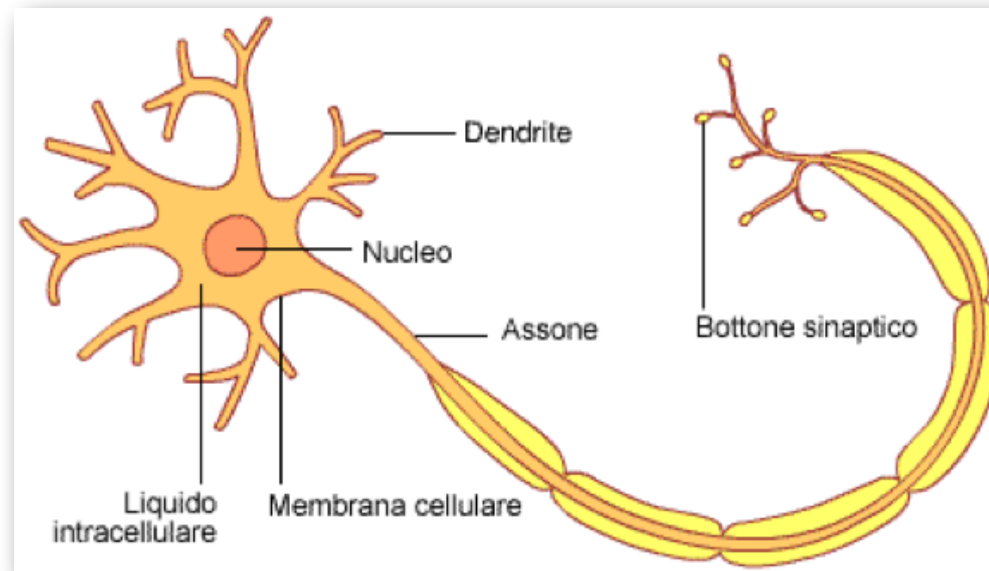
The biological model: the human brain



- Information is transported between neurons in the form of electrical stimuli along the dendrites.
- The incoming information that reaches the neuron's dendrites is summed and then sent along the axon to the dendrites at its end (synapses), where the outgoing information passes to connected neurons, if it exceeds a certain threshold. In this case, the neuron is said to have "**activated**".

Neural Networks

The biological model: the human brain

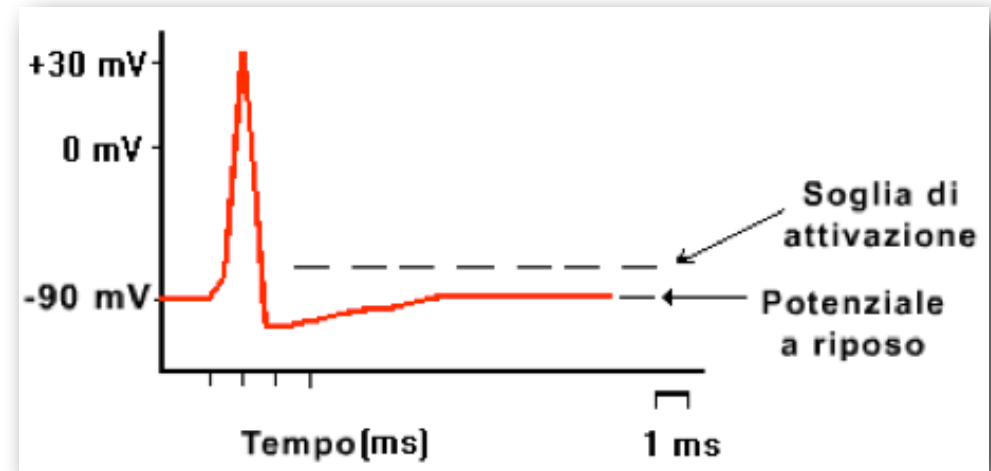


- If the incoming stimulation is not sufficient, the information is not transported further. In this case, the neuron is said to be "**inhibited**".
- The connections between neurons are adaptive, which means that the structure of the connections changes dynamically.
- It is commonly recognized that the human brain's ability to learn is based on this adaptation.

Neural Networks

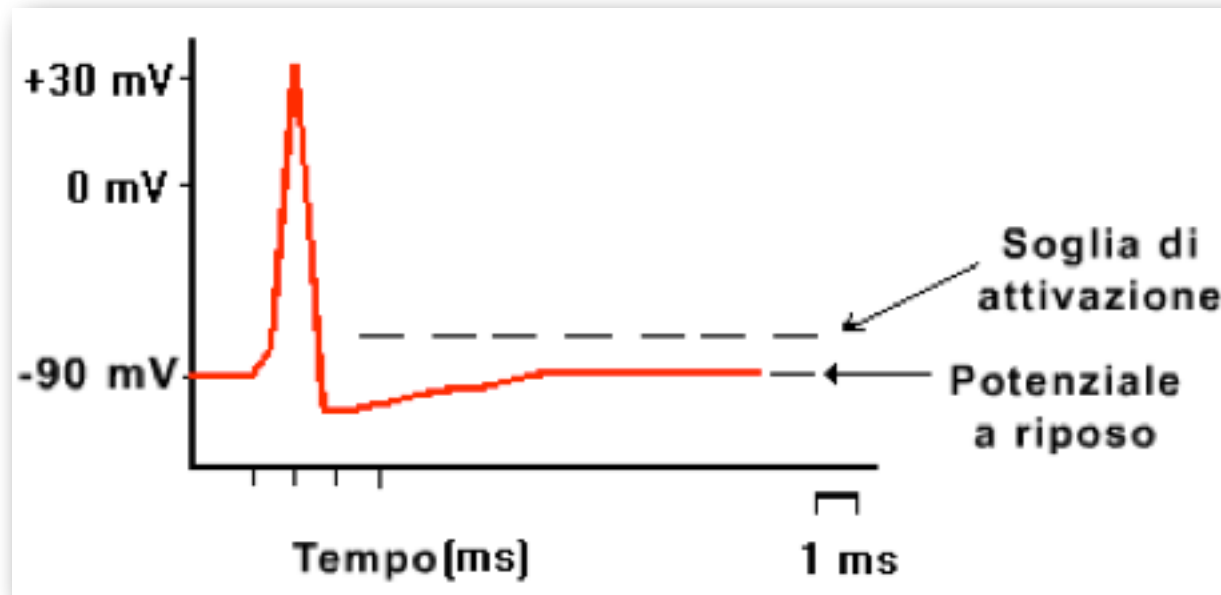
The biological model: the human brain

- The neuron, since it can emit or not an electrical signal will also have an **activation threshold**.
- As long as the **membrane** of the neuron remains undisturbed no action potential originates, but if any event causes a sufficient increase in potential from the -90mV level towards the zero level, it is the same rising voltage that causes many sodium channels to begin to open.
- This allows rapid entry of **sodium ions**, which again causes a rise in membrane potential, which opens even more sodium channels, increasing the flow of sodium ions entering the cell.
- The process is self-perpetuating with a positive feedback loop until all sodium channels are totally open.



Neural Networks

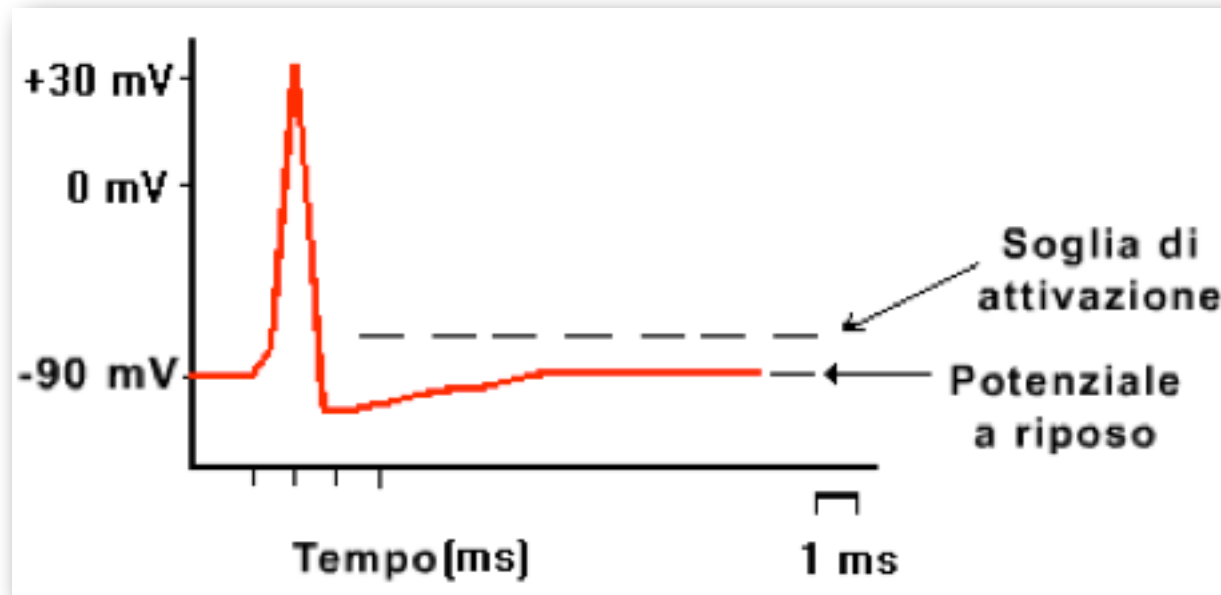
The biological model: the human brain



- But at this point in the next fraction of milliseconds the rising membrane potential causes a closure of sodium channels and an opening of potassium channels, after which the action potential ends.
- In order to trigger the action potential it is necessary that the membrane potential increases by 15/30mV, bringing it to about -65mV (activation threshold).
- So it is not always said that the activation threshold is reached, it depends on the speed of the above process.

Neural Networks

The biological model: the human brain

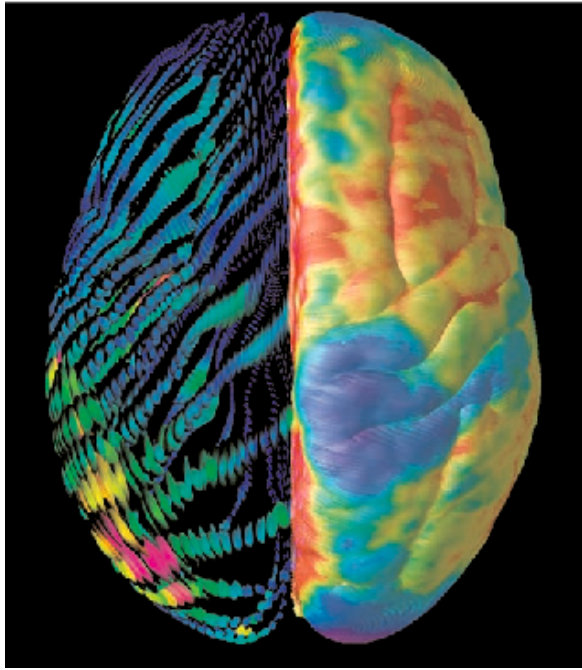


This **potential** is spread throughout the neuronal structure, even at the ends of synapses where the electrical signal becomes chemical to pass to the interconnected neuron.

If this occurs the synapse will then be of **excitation**, otherwise of **inhibition**.

Neural Networks

The biological model: the human brain

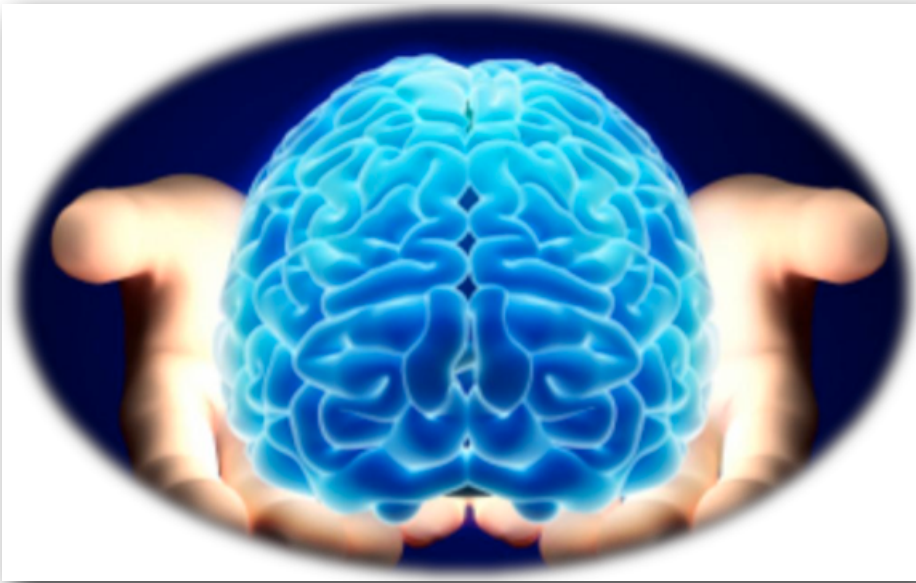


The brain, is the most complex and mysterious object known: 1,300-1,500 grams of gelatinous tissue composed of 100 billion cells (**neurons**), each of which develops an average of 10 thousand **connections** with neighboring cells.

- During fetal life, the body produces no less than 250,000 neurons per minute.
- But 15 to 30 days before birth, production stops, and for the brain begins a second phase that will last a lifetime: the creation of connections between cells.

Neural Networks

The biological model: the human brain



- In this process, the cells that fail to make connections are eliminated, so that at the time of birth they are already halved.
- The human brain (more correctly "**encephalon**") is the result of the superposition of the three types of brains that appeared during the evolution of vertebrates.

- *From below (at the base of the skull), the oldest brain (**rhombencephalon**), which specializes in controlling involuntary functions such as alertness, respiration, circulation, and muscle tone. It includes the cerebellum and the parts of the spinal cord that extend into the brain.*
- *Moving up, there is the **midbrain**: a small portion of nerve tissue consisting of the so-called cerebral peduncles and the lamina quadrigemina.*
- *Finally there is the **forebrain**, the most "modern" part, divided into diencephalon and telencephalon. The first, also called "limbic system", contains structures such as thalamus, hypothalamus, pituitary gland and hippocampus, from which come sensations such as hunger, thirst or sexual desire. Finally, the newest part of the brain: the cortex, where intelligence and language functions are located.*

Neural Networks

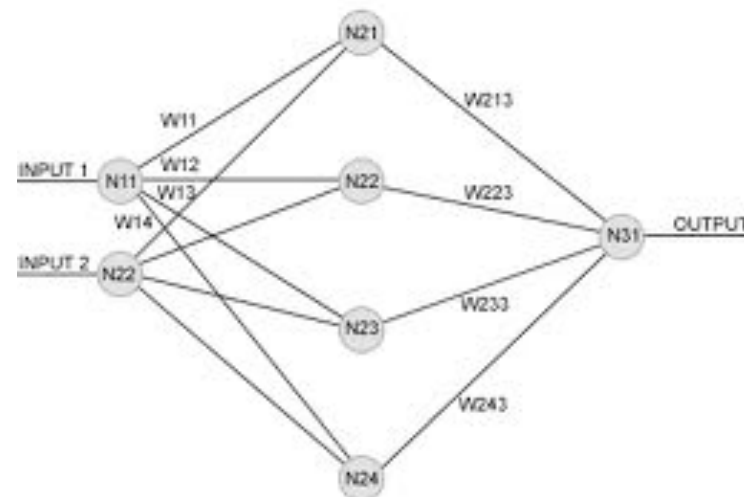
The components of a neural network

- From a general point of view there are **many different types of neural networks**, but they all have almost the same components.
- If one wants to simulate the human brain using a neural network, it is obvious that some drastic simplifications must be made.
- First of all, it is not possible to "replicate" the parallel operation of all neural cells. Although there are computers that have parallel processing capabilities, the large number of processors that would be required cannot be implemented by currently available hardware.
- Another limitation is that the internal structure of a computer cannot be changed while performing any task.
- **And how to implement electrical stimulations in a computer program?**

Neural Networks

The components of a neural network

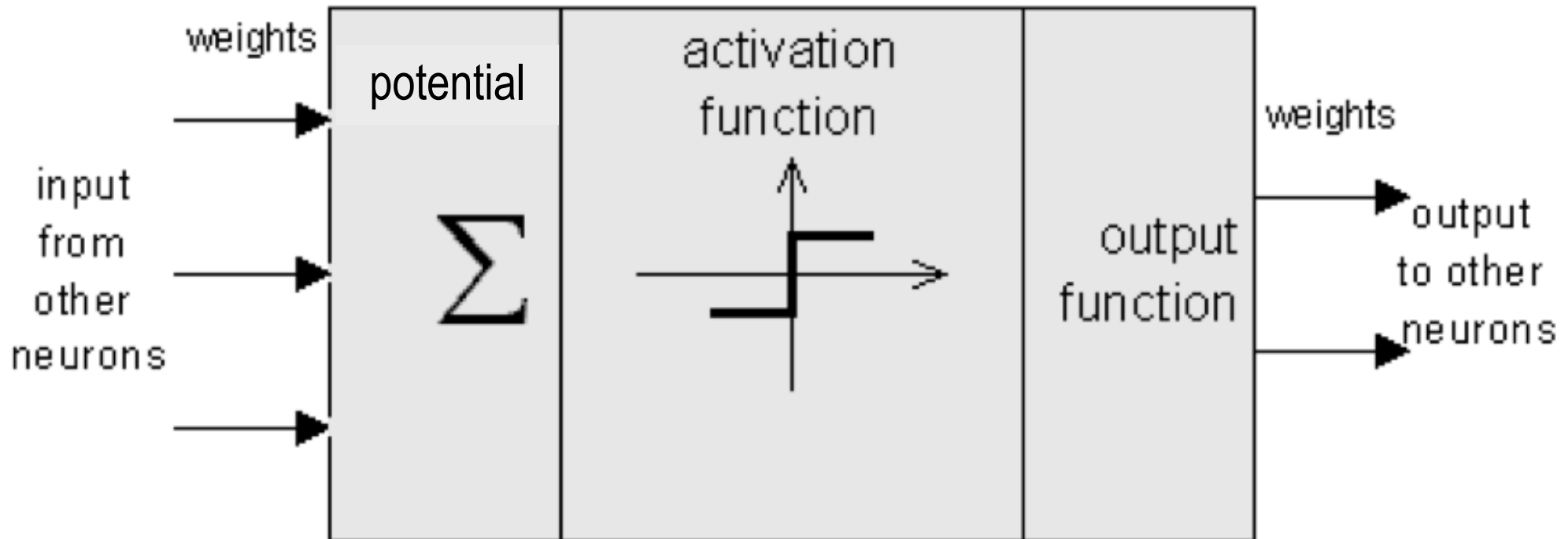
- ▶ From this, an idealized model is derived for simulation purposes.
- ▶ Like the human brain, an **artificial neural network** consists of neurons and the connections between them.
- ▶ Neurons carry incoming information on their outgoing connections to other neurons. In neural network terminology these connections are called **weights**.
- ▶ The "electrical" information is simulated with specific values stored in the weights.
- ▶ The change of the connection structure can be simulated by simply changing the values of the weights.



Neural Networks

The components of a neural network

The following figure shows a **neuron** in an artificial neural network:

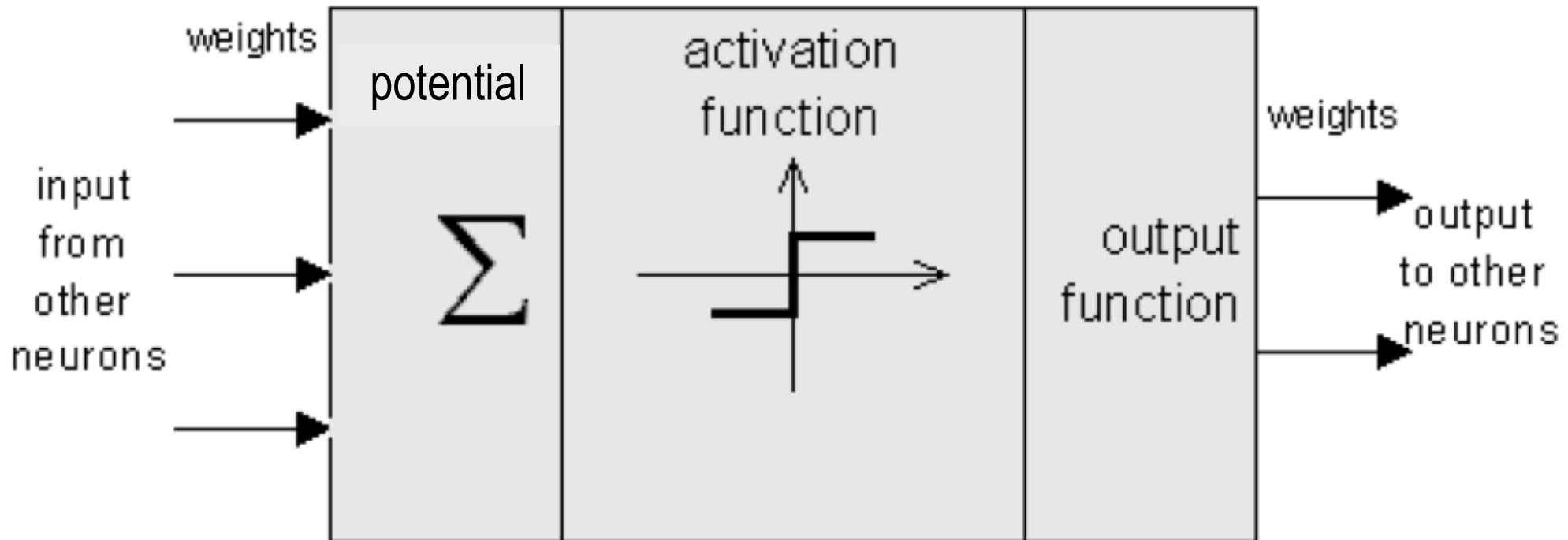


An **artificial neuron** is similar to a biological neural cell, and functions in the same way. Information (input values) arrive at the neuron and are combined (multiplied) with corresponding **weights**.

Neural Networks

The components of a neural network

The following figure shows a **neuron** in an artificial neural network:

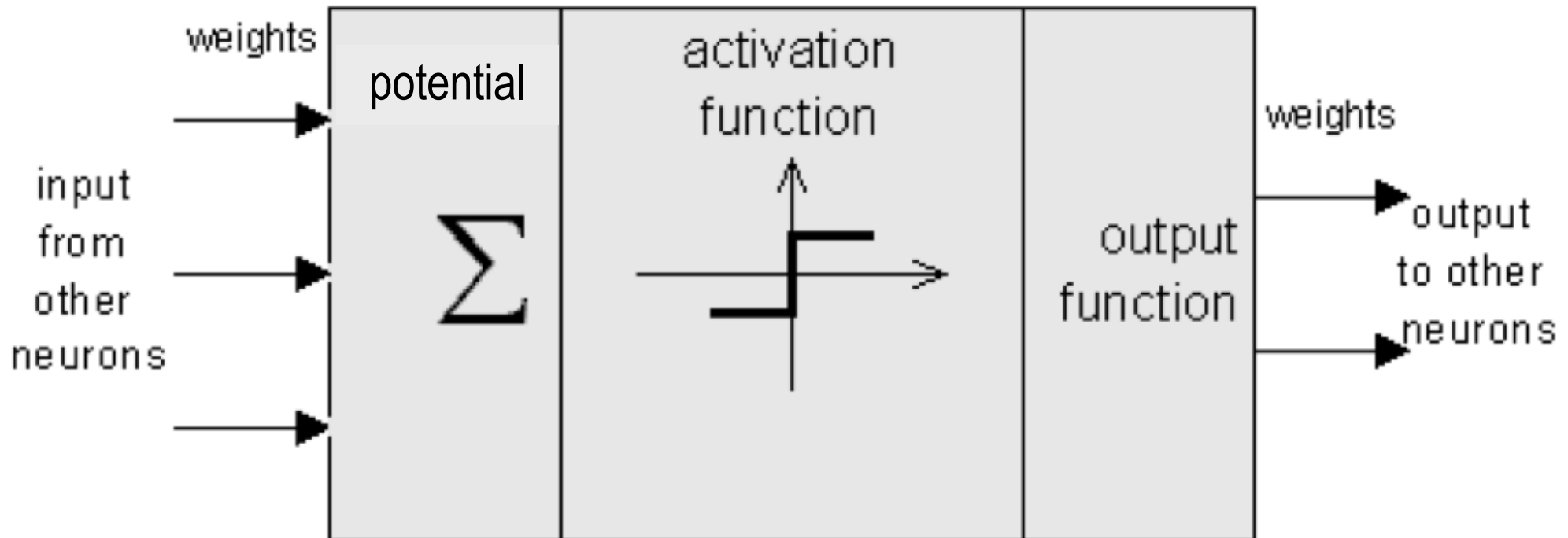


A propagation function sums these values, providing the **potential** of the neuron. The resulting value is compared to a given **threshold value** based on the neuron's **activation function**.

Neural Networks

The components of a neural network

The following figure shows a **neuron** in an artificial neural network:

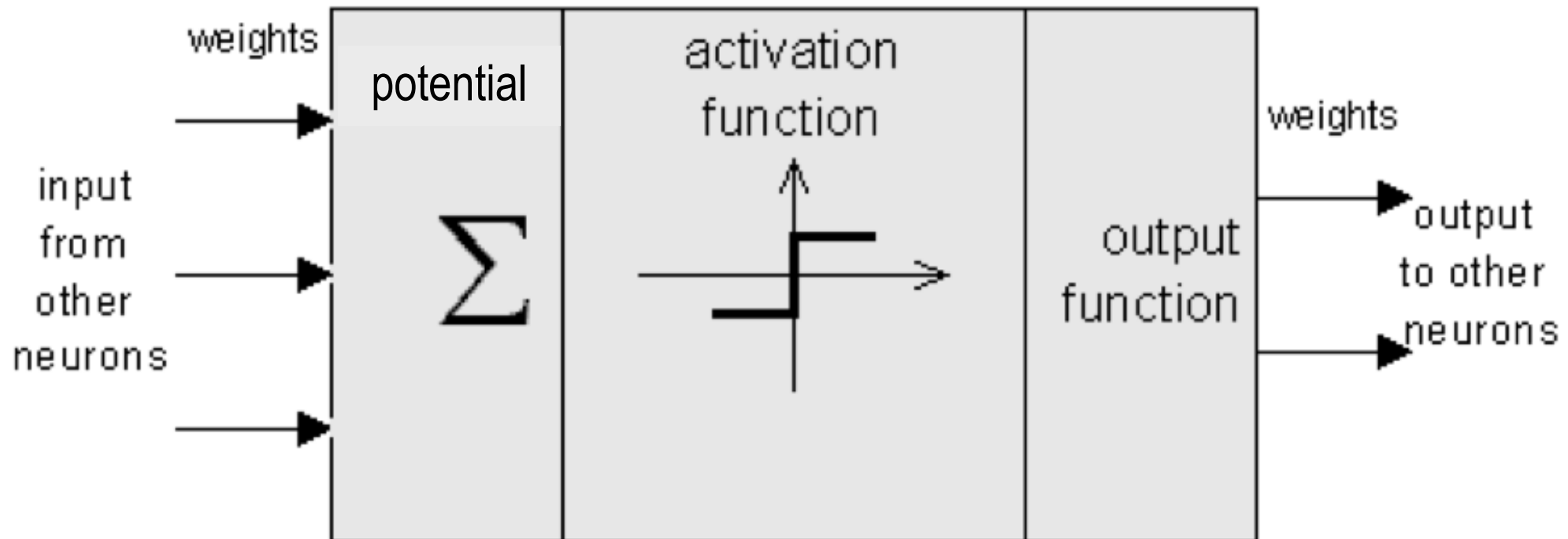


If the sum exceeds the threshold value, the neuron is **activated**, otherwise it will be **inhibited**.

Neural Networks

The components of a neural network

The following figure shows a **neuron** in an artificial neural network:



If activated, the neuron sends an output on the **output-weighted connections** to all connected neurons, and so on.

Neural Networks

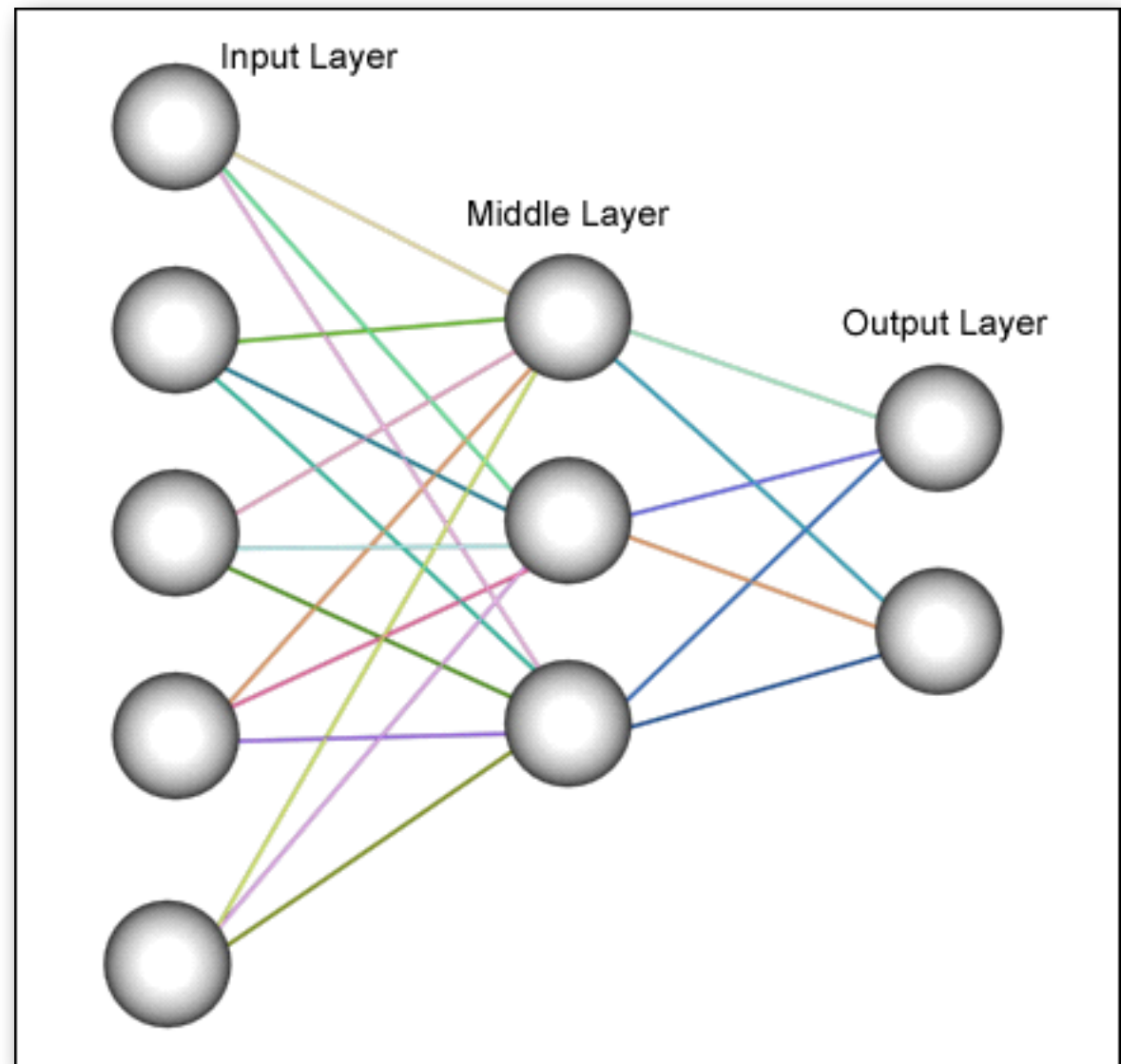
The components of a neural network

In a neural network, neurons are grouped into layers, called **neuron layers**.

Typically, each neuron in a layer is connected to all neurons in the previous and next layers (except the input layer and the output layer of the network).

The information provided by a neural network is propagated *layer-by-layer* from the input layer to the output layer through none, one or more hidden layers.

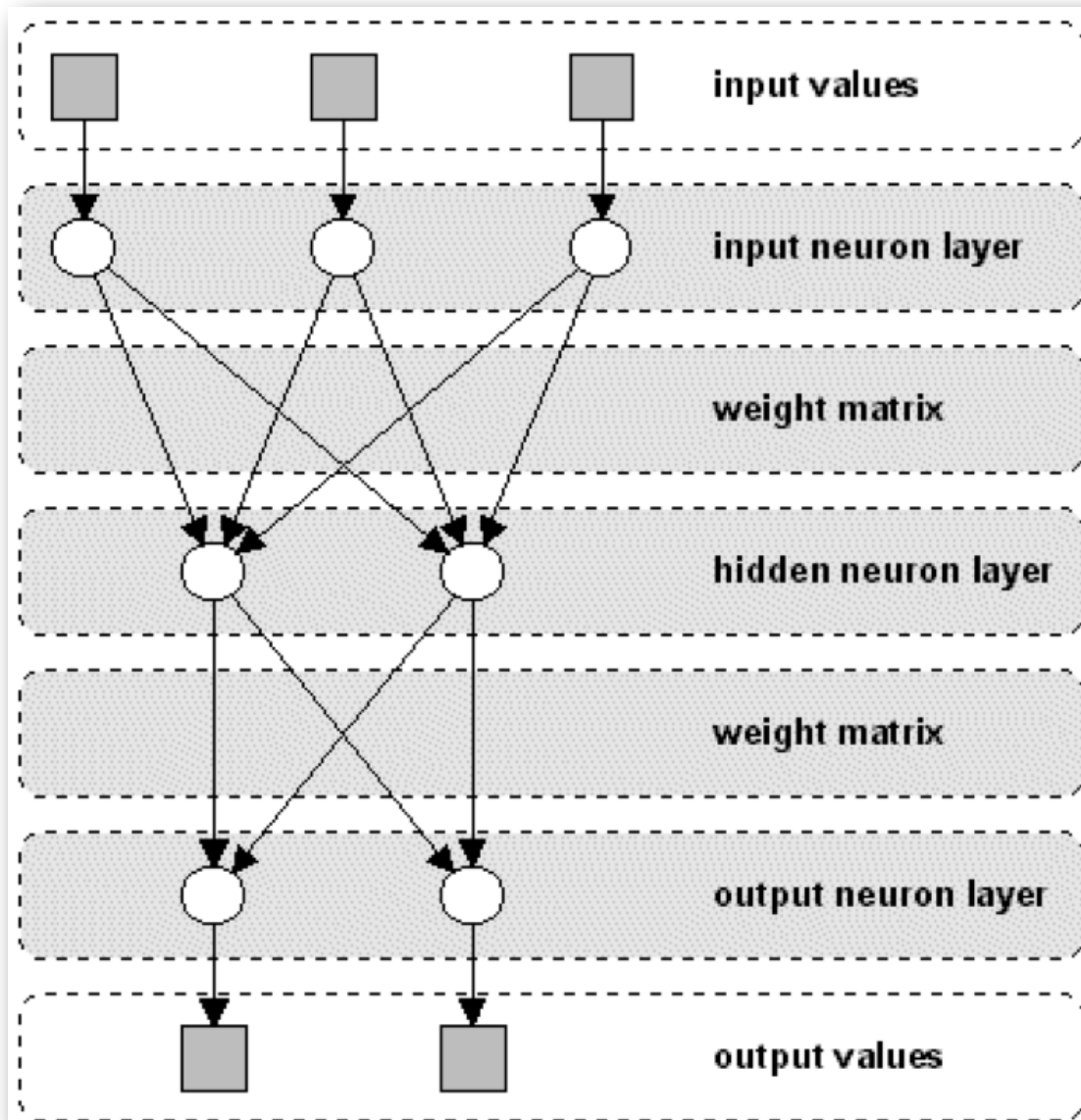
Depending on the **learning algorithm**, it is also possible for information to propagate backwards through the network (*backpropagation*).



Neural Networks

The components of a neural network

The following figure shows a neural network with three layers.



Note that this is not the general structure of a neural network.

For example, some types of neural networks do not have hidden layers or the neurons in a layer are arranged as a matrix.

What is **common** to all types of neural networks is the presence of at least one **weight matrix**, which identifies the connections between two layers of neurons.

Neural Networks

Possibilities and limitations

Neural networks are built to solve problems that cannot be solved by traditional algorithms. Such problems are usually **optimization** or **classification** problems.

The different domains in which neural networks can be used are:

- classification of models (pattern recognition)
- regularity detection
- image processing
- speech analysis
- optimization problems
- robot steering
- processing inaccurate or incomplete input
- quality assurance
- stock market forecasting
- simulation
- ...

Neural Networks

Possibilities and limitations

- There are many **different types of neural networks**, each with particular properties, so each application domain has its own network type.
- In general, it can be said that neural networks are **very flexible systems** for "*problem solving*" purposes.
- A special feature of neural networks is **fault tolerance**. This means that if a neural network is trained for a specific problem, it will be able to produce correct results even if the problem to be solved is not exactly the same as the one already learned.
- For example, suppose a neural network has been trained to recognize human language. During the learning process, a person utters some words, which are learned by the network. Then, if trained correctly, the neural network should be able to recognize the words spoken by another person as well.

Neural Networks

Possibilities and limitations

- Although neural networks are capable of finding solutions to difficult problems, the **results cannot be guaranteed.**
- They are only **approximations of the desired solution** and some error is always present.
- In addition, **there are problems that cannot be properly solved with neural networks.**

Neural Networks

Possibilities and limitations

An example on "pattern recognition" may clarify the concept better.

If you meet a person you have seen before, you usually recognize him/her a second time, even if he/she does not look the same as the first time you met him/her.

Suppose now that you have trained a neural network with a photograph of that person: the image will be surely recognized by the network.

But if you "disturb" the image or e.g. rotate it by a certain angle, the recognition will probably fail.

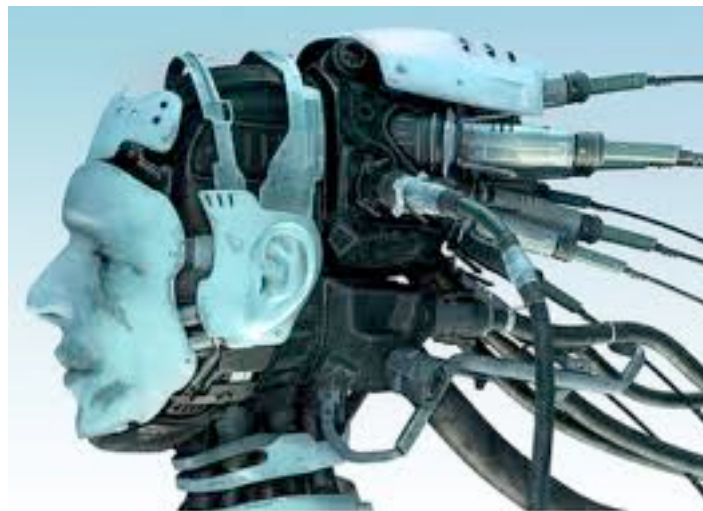


Neural Networks

Possibilities and limitations

Surely, no one would ever use a neural network in a sorting algorithm, as there are much better and faster algorithms.

But in application domains like the ones mentioned above, neural networks are always a **good alternative to existing algorithms** and definitely worth a try.



Types of Neural Networks

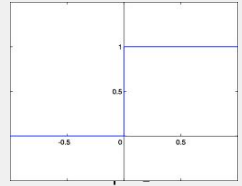


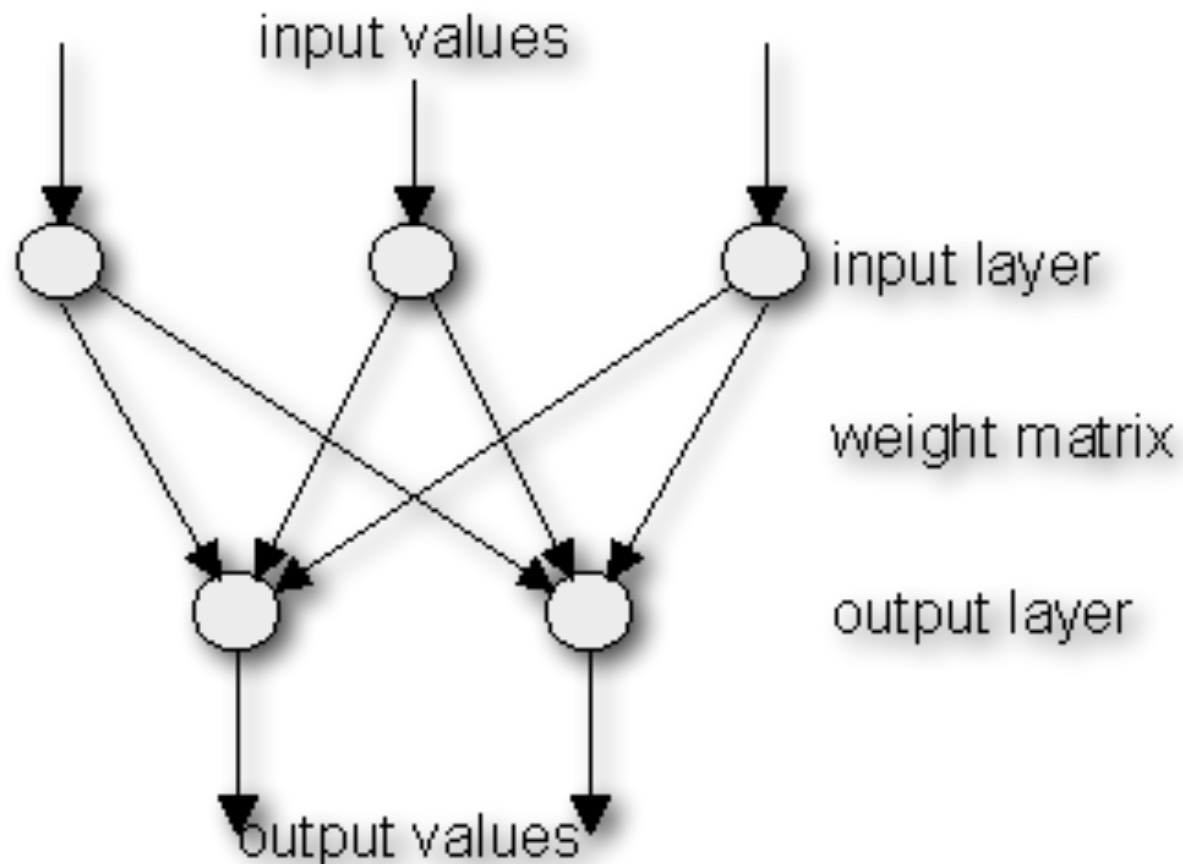
- [There are several types of neural networks, which can be distinguished by **type** (*feedforward* or *feedback*), **structure**, and **learning algorithm** used.
- [The **type** of a neural network indicates whether the neurons in the various layers of the network can be connected to each other.
- [Feedforward type neural networks allow only neuronal connections between two different layers, while Feedback type networks also allow connections between neurons in the same layer.
- [A selection of neural network types will be examined below.

Types of Neural Networks

Perceptron

- The Perceptron was introduced by F. Rosenblatt in 1958.
- It is a very simple type of neural network with two layers of neurons that accepts only binary input and output (0 and 1).
- The learning process is supervised and the network is capable of solving basic logical operations such as AND and OR.
- It is also used for pattern classification.
- More complex logical operations (such as XOR) cannot be solved by a Perceptron.

type	Feedforward
layers	1 input layer, 1 output layer
input/output values	binary
activation function	hard limit 
learning method	supervised
learning algorithm	Hebb's rule
use	simple logical operations, pattern classification

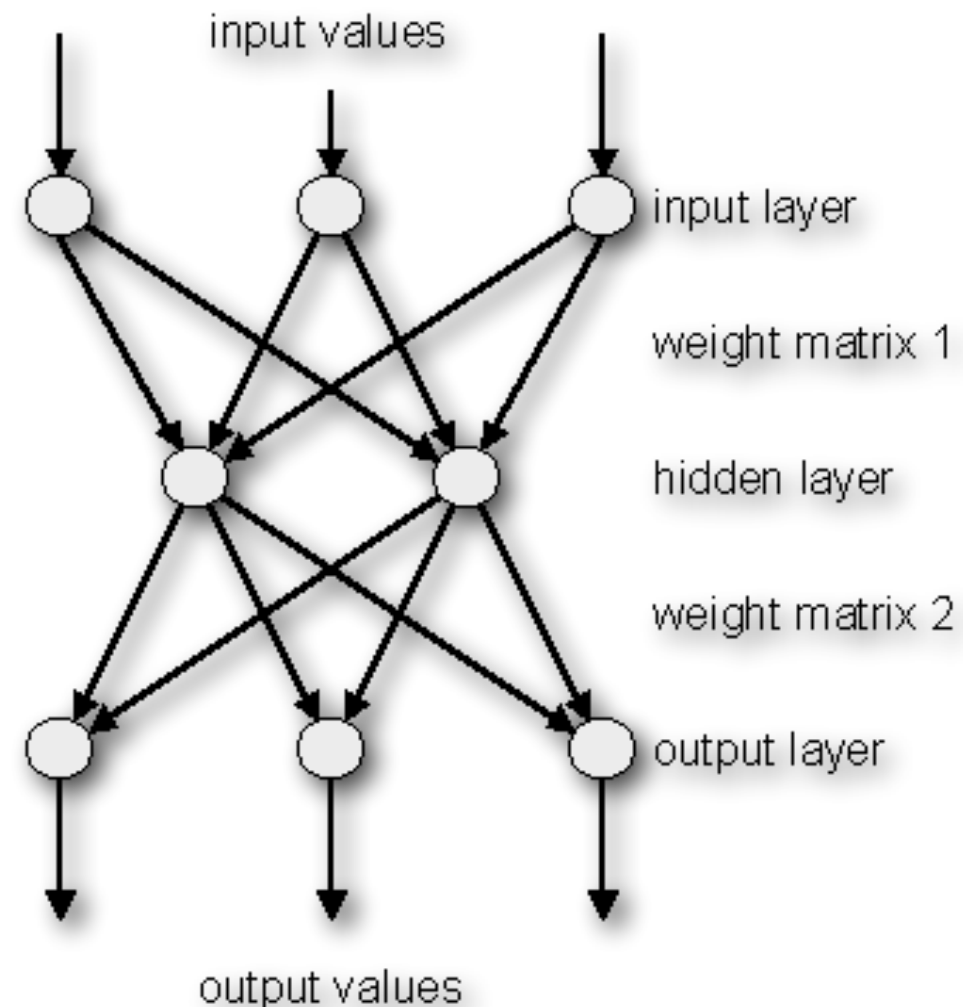


Types of Neural Networks

Multi-Layer Perceptron

- The Multi-Layer Perceptron was introduced by M. Minsky and S. Papert in 1969.
- It is an extended Perceptron, with one or more layers hidden between the input and output layers.
- Due to its extended structure, a Multi-Layer Perceptron can solve any logical operation, including the XOR problem.

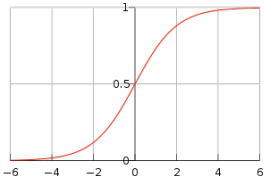
type	Feedforward
layers	1 input layer, 1 or more hidden layers, 1 output layer
input/output values	binary
activation function	hard limit, sigmoid
learning method	supervised
learning algorithm	delta rule
use	complex logical operations, pattern classification

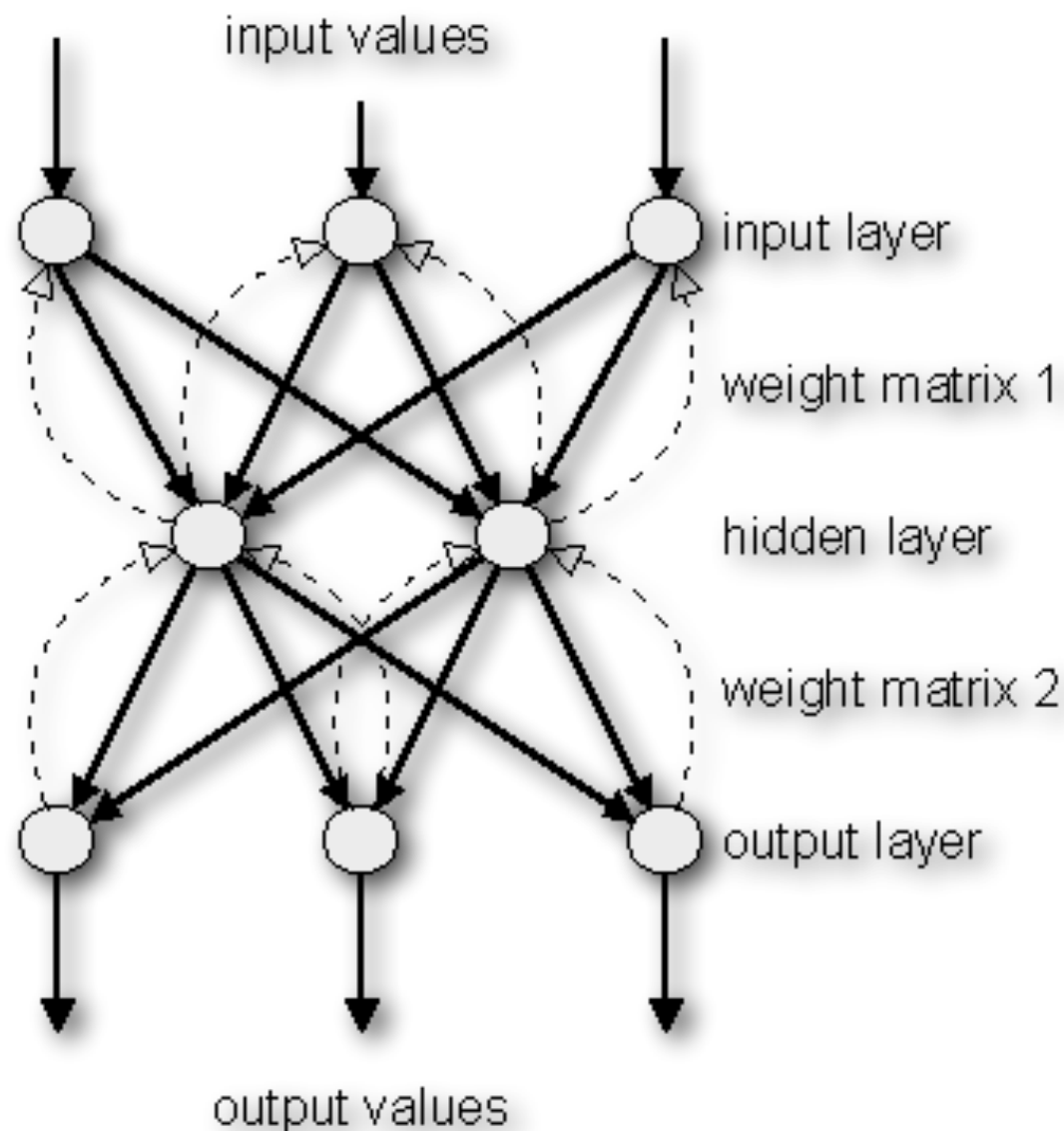


Types of Neural Networks

MLP Backpropagation

- The MLP Backpropagation network was introduced by Hinton, Rumelhart and Williams in 1986 and is one of the most powerful types of neural network.
- It has the same structure as the Multi-Layer Perceptron and uses the "backpropagation" learning algorithm.

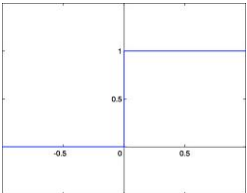
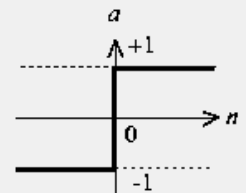
type	Feedforward
layers	1 input layer, 1 or more hidden layers, 1 output layer
input/output values	binary
activation function	sigmoid 
learning method	supervised
learning algorithm	backpropagation
use	complex logical operations, pattern classification, speech analysis

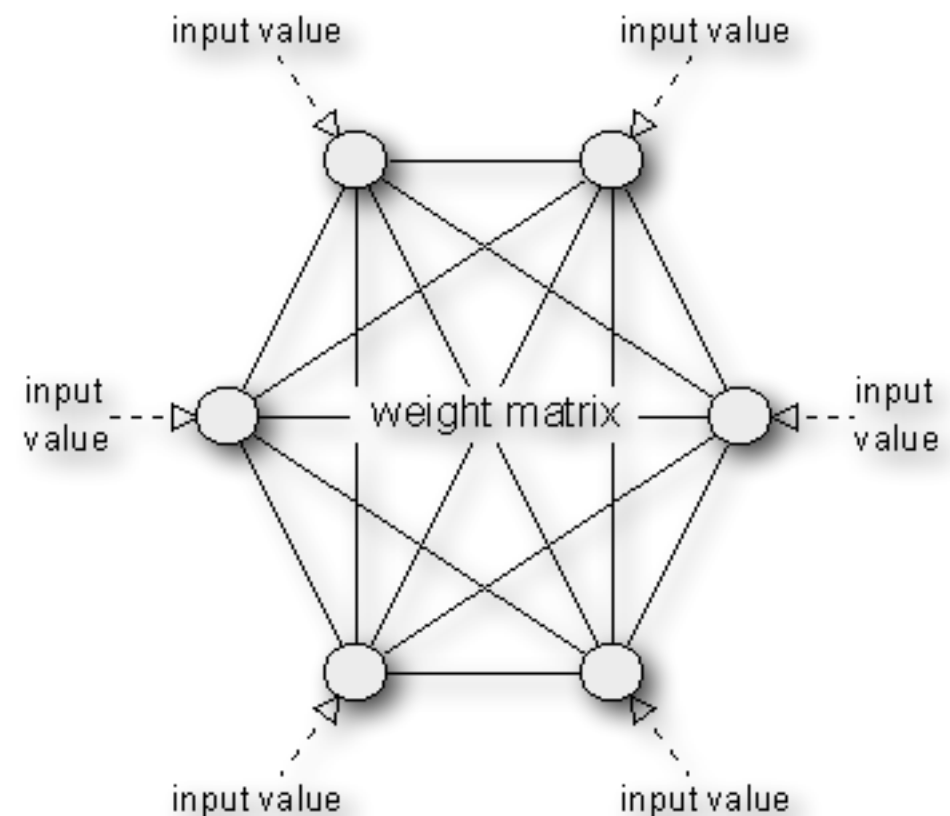


Types of Neural Networks

Hopfield

- This network was introduced by physicist Hopfield in 1982 and belongs to the types of neural networks called "thermodynamic models".
- It consists of a set of neurons, in which each neuron is connected to every other neuron.
- There is no differentiation between input and output neurons.
- The main application of a Hopfield network is the storage and recognition of patterns, for example images.

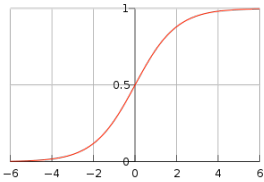
type	Feedback
layers	1 connection matrix
input/output values	binary
activation function	sign, hard limiter <div style="display: flex; justify-content: space-around; align-items: center;">   </div>
learning method	unsupervised
learning algorithm	delta rule simulated annealing (most used)
use	pattern association, optimization problems

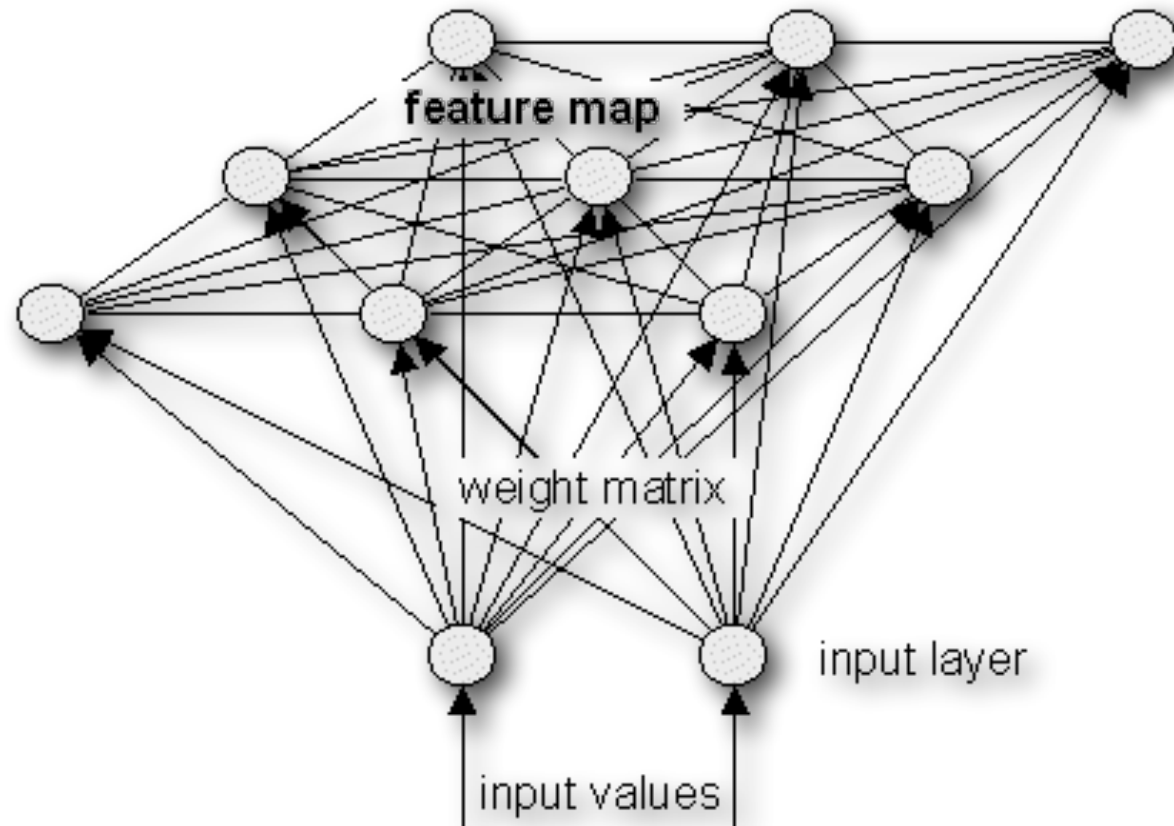


Types of Neural Networks

Mappa di Kohonen

- This network was introduced by Finnish professor Teuvo Kohonen (University of Helsinki) in 1982.
- It is probably the most useful type of neural network for simulating the learning process of the human brain.
- Its "core" is the feature map, a layer in which neurons organize themselves according to certain input values.
- This type of neural network is both feedforward (input→map neurons) and feedback (interconnected map neurons).

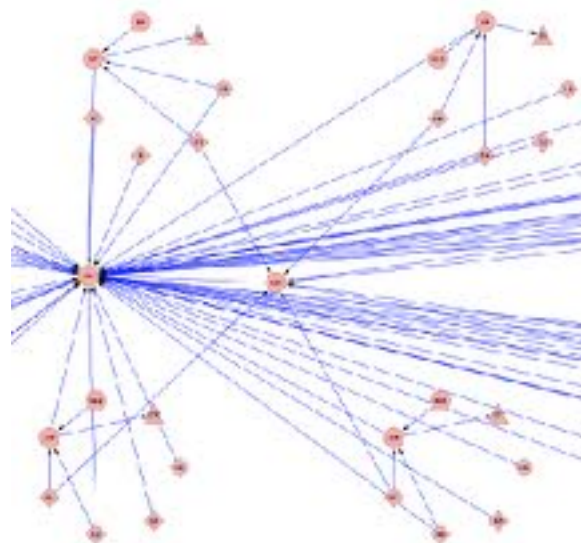
type	Feedforward/Feedback
layers	1 input layer, 1 "map" layer
input/output values	binary, real
activation function	sigmoid 
learning method	unsupervised
learning algorithm	self-organization (example)
use	pattern classification, optimization problems simulation



During the learning process, neurons on the map organize themselves according to input values. This results in a clustered neuronal structure, in which neurons with similar properties (values) are arranged in connected sectors on the map.

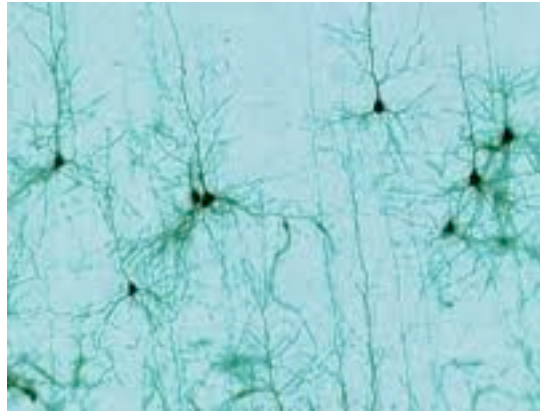
The learning process

- This section describes the learning capabilities of neural networks.
- First, the term **learning** is explained.
- Next, an overview of specific learning algorithms for neural networks is presented.



The learning process

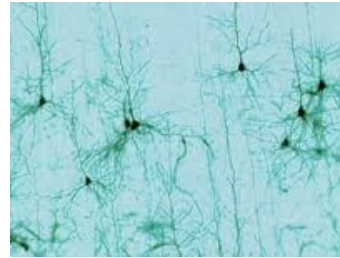
What does "learning" mean for a neural network?



- ▶ In the human brain, information is passed between neurons in the form of **electrical stimulation** along dendrites.
- ▶ If a certain amount of stimulation is received by one neuron, it **generates an output** to all other connected neurons and so the information takes its way to the destination, where some reaction will occur.
- ▶ If the incoming **stimulus** is **too low**, no output is generated by the neuron and subsequent information transport will be **blocked**.

The learning process

What does "learning" mean for a neural network?



Explaining how the human brain learns certain things is very difficult and no one knows exactly.

It is assumed that during the learning process the **structure of connection between neurons is changed**, so that certain stimulations are accepted only by certain neurons.

This means that there are **strong links between neural cells** once they have learned a specific fact, allowing for rapid recall of information.

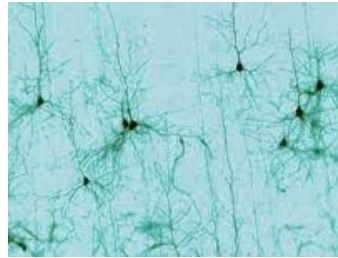
If similar information arrives later, the same nerve cells are stimulated and will **adapt their connection structure** according to this new information.

On the other hand, **if a specific piece of information is not recalled for a long time**, the connection structure established between the responsible neural cells will become "weaker".

This happens, for example, when one "forgets" a previously acquired fact or remembers it only vaguely.

The learning process

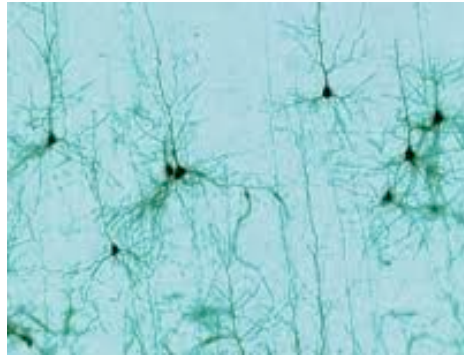
What does "learning" mean for a neural network?



- As mentioned earlier, **neural networks attempt to mimic** the learning capacity of the human brain.
- In fact, the artificial neural network is also made of neurons and dendrites. Unlike the biological model, **a neural network has an immutable structure**, built on a certain number of neurons and a certain number of connections between them (called "weights"), which have certain values.
- What changes in the **learning process** are the values of those weights.

The learning process

What does "learning" mean for a neural network?

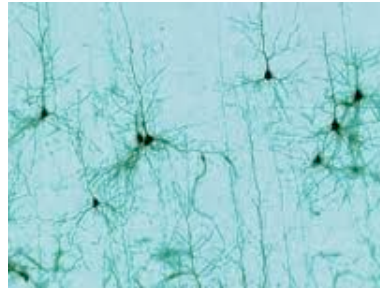


Compared to the organic original this means that:

1. **Incoming information** "stimulates" (exceeds a certain threshold value of) certain neurons.
2. These **pass information** to connected neurons or **prevent** further transport along "weighted" outgoing connections.
3. The **value of a weight** will be increased if the information is to be carried and reduced otherwise.

The learning process

What does "learning" mean for a neural network?



When learning different inputs, **the values of the weights are dynamically changed** until they are in balance, such that each input will lead to the desired output.

The training of a neural network results in a **matrix** that contains the values **of the weights between the neurons**.

Once a neural network has been properly trained, it will likely be able to find the desired outcome for a previously learned input using values from the weight matrix.

Why probably? Unfortunately, there is no guarantee that a neural network will produce the correct results in every case. Very often there is some residual error after the learning process, so the output generated is only a **good approximation** of the perfect output in most cases.

The learning process

Supervised learning

The learning algorithm of a neural network can be with or without **supervision**.

A neural network is supervised if the desired output is already known.

Example: pattern association

Suppose that, a neural network needs to learn to associate the following pairs of patterns. The input patterns are decimal numbers, each represented by a sequence of bits. The output patterns are provided in the form of binary values of the decimal numbers:

input pattern	output pattern
0001	001
0010	010
0100	011
1000	100

1. During the learning, one of the input patterns is "presented" at the input layer of the network.
2. This pattern propagates through the network (regardless of its structure) to the output layer.
3. The output layer generates an output pattern which is then compared to the actual pattern. Depending on the difference between the output produced by the network and the target, an error value is calculated.
4. This **error** indicates the learning effort of the network, which can be controlled by the "imaginary supervisor".
5. The higher the calculated error value, the more the values of the weights will be changed.

The learning process

Unsupervised learning

- ▶ **Unsupervised neural networks** (i.e., that "learn" without supervision) do not have predefined outputs.
- ▶ The outcome of the learning process cannot be determined a priori.
- ▶ During the learning process, the units (the weights) of the neural network are "arranged" within a certain range of values, depending on the input values.
- ▶ The goal is to group together similar units that are "close" in certain areas of the value range.
- ▶ This effect can be used effectively for the purpose of pattern classification (*clustering*).
- ▶ The following example of Kohonen's "*self organizing*" network clarifies the operation in detail.

A 3D Kohonen Feature Map



input values: 10	learning cycle: 0
input layer: 3 neurons	learning rate: 0.6
map size: 9x9 neurons	activation area: 3.0
weights: 243	elapsed time: 0.0010 sec

The learning process

Forward propagation

It is a supervised learning algorithm that describes the "flow of information" through a neural network from its input layer to its output layer.

The algorithm works as follows:

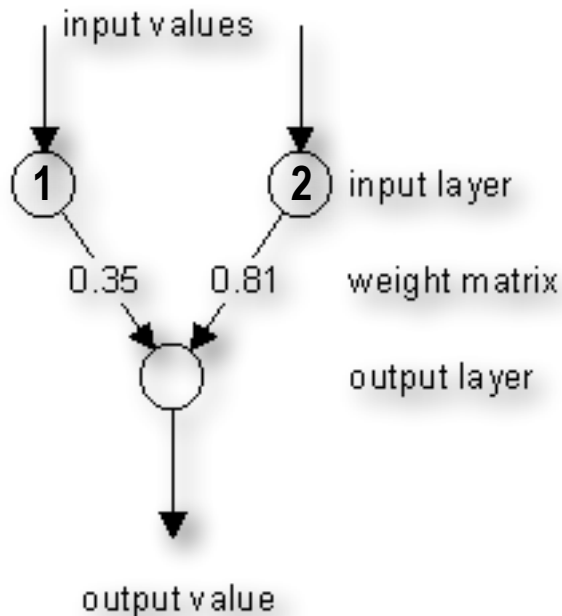
1. Set all weights to random values between -1.0 and +1.0
2. Provide an input pattern (binary values) to the neurons of the input layer
3. Activate each neuron in the next layer as follows:
 - *multiply the weight values of the connections leading to the neuron with the output values of the preceding neurons;*
 - *sum these values;*
 - *pass the result to an activation function, which calculates the output value of the neuron.*
4. Repeat this operation until the output layer is reached
5. Compare the calculated output pattern with the expected pattern and calculate the error
6. Change all weights by adding the error value to the (old) weight values
7. Return to step 2
8. The algorithm terminates if all output patterns match their predicted patterns

The learning process

Forward propagation

Example.

Suppose we have the following two-layer Perceptron



Pattern to be "learned":

input	expected output (target)
0 1	0
1 1	1

(1) Initially, the weights are set to random values (0.35 and 0.81) and the "learning rate" μ of the network is set to 0.25; then the values of the first input pattern (0 1) are provided. The neuron of the output layer is then activated:

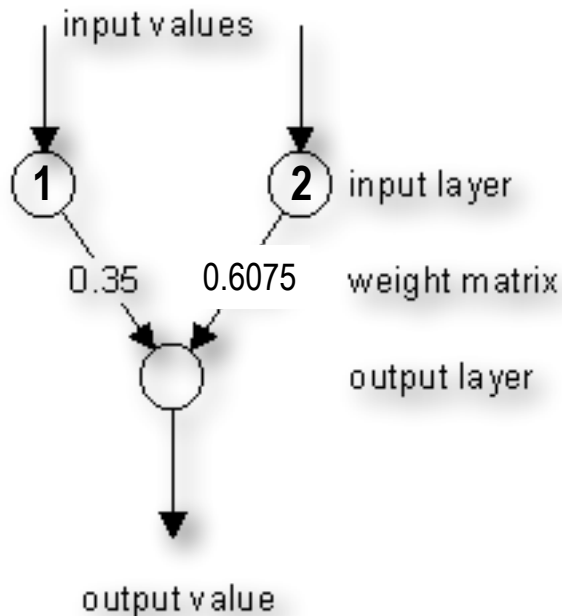
Input 1 of the output neuron	$0 \times 0.35 = 0$
Input 2 of the output neuron	$1 \times 0.81 = 0.81$
Sum of inputs	$0 + 0.81 = 0.81$ (= output)
Error calculation	$0 - 0.81 = -0.81$
Weight 1 change value	$0.25(\mu) \times 0(\text{input1}) \times (-0.81) = 0$
Weight 2 change value	$0.25(\mu) \times 1(\text{input2}) \times (-0.81) = -0.2025$
Weight 1 modification	$0.35 + 0 = 0.35$ (immutato)
Weight 2 modification	$0.81 + (-0.2025) = 0.6075$

The learning process

Forward propagation

Example.

Suppose we have the following two-layer Perceptron



Pattern to be "learned":

input	expected output (target)
0 1	0
1 1	1

(2) After the weights have been changed, the second input pattern (1 1) is provided and the output neuron is activated again, with the new values of the weights:

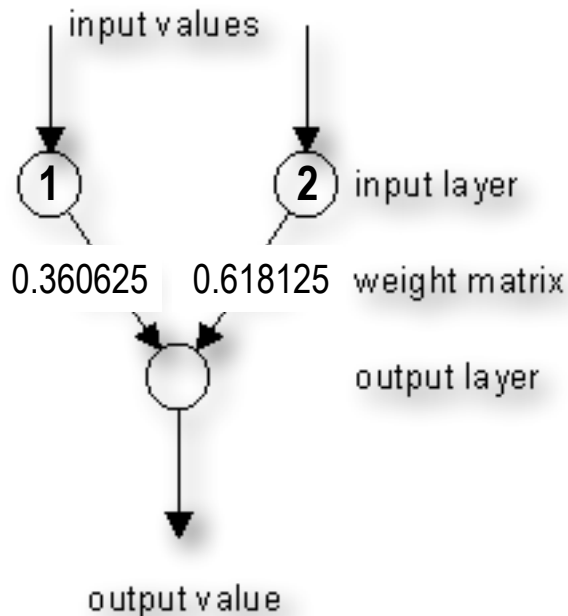
Input 1 of the output neuron	$1 \times 0.35 = 0.35$
Input 2 of the output neuron	$1 \times 0.6075 = 0.6075$
Sum of inputs	$0.35 + 0.6075 = 0.9575$ (= output)
Error calculation	$1 - 0.9575 = \mathbf{0.0425}$
Weight 1 change value	$0.25(\mu) \times 1(\text{input1}) \times 0.0425 = 0.10625$
Weight 2 change value	$0.25(\mu) \times 1(\text{input2}) \times 0.0425 = 0.10625$
Weight 1 modification	$0.35 + 0.10625 = 0.360625$
Weight 2 modification	$0.6075 + 0.10625 = 0.618125$

The learning process

Forward propagation

Example.

Suppose we have the following two-layer Perceptron



Pattern to be "learned":

input	expected output (target)
0 1	0
1 1	1

(3) This concludes the first learning step (or "epoch"), in which each input pattern has been propagated through the network and the weights have been updated.

The overall error of the network can now be calculated by averaging the squares of the output errors (MSE=Mean Squared Error):

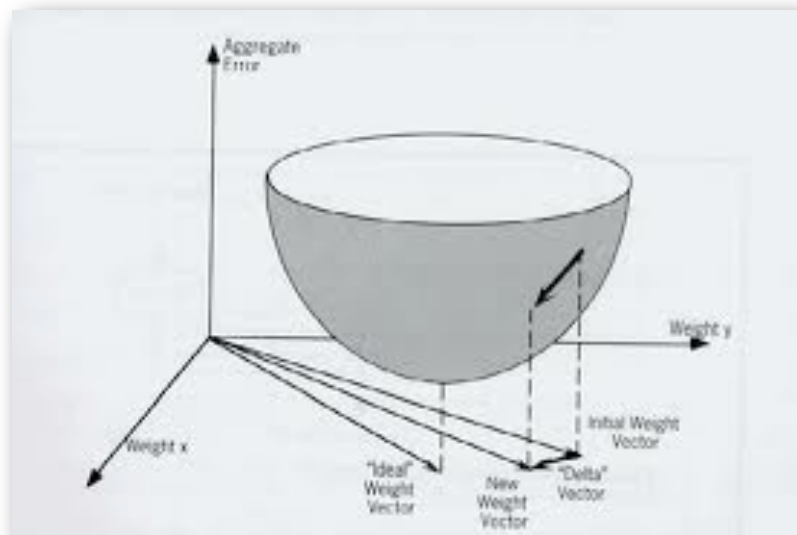
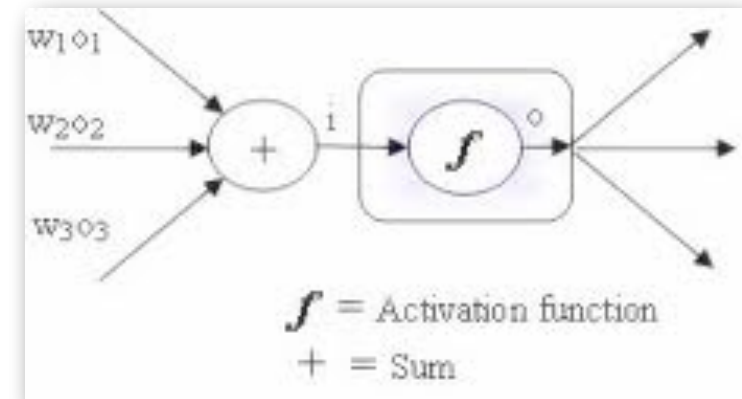
Network error (MSE)	$[(-0.81)^2 + (0.0425)^2] / 2 = 0.32895313$
---------------------	---------------------------------------------

By repeating this procedure, the network error is progressively reduced. The algorithm correctly terminates (converges) when the network error is zero (ideal situation) or almost...

The learning process

Backward propagation

Backpropagation is a supervised learning algorithm and is mainly used by the Multi-Layer Perceptron to change the weights attached to hidden neuron layers.



The backpropagation algorithm uses the output error to change the values of the backward weights.

To obtain the error, a "forward propagation" step must have been previously performed, during which neurons are activated using the sigmoid activation function.

The learning process

Backward propagation

The formula for **sigmoid activation** is:
$$f(x) = \frac{1}{1 + e^{-x}}$$

The algorithm works as follows.

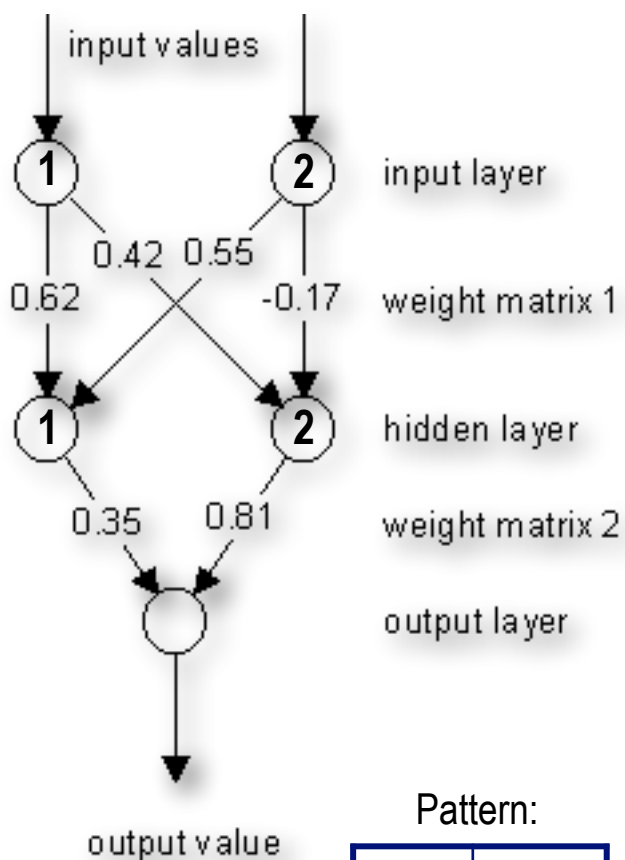
1. Perform the forward propagation step for an input pattern and calculate the output error
2. Change all values of each weight matrix using the formula:
$$(old)weight + learning\ rate \times output\ error \times output\ neuron\ (i) \times output\ neuron\ (i+1) \times [1 - output\ neuron\ (i+1)]$$
3. Go back to step 1
4. The algorithm terminates if all output patterns match the targets

The learning process

Backward propagation

Example.

Suppose we have the following three-layer Multi-Layer Perceptron:



Pattern:

input	target
0 1	0
1 1	1

(1) Initially, the weights are set to the random values (0.62, 0.42, 0.55, -0.17) for weight matrix 1 and (0.35, 0.81) for matrix 2, with $\mu = 0.25$.

Then the first input (0 1) to the input layer is provided and the neurons of the hidden layer are activated:

Input of hidden neuron 1	$0 \times 0.62 + 1 \times 0.55 = 0.55$
Input of hidden neuron 2	$0 \times 0.42 + 1 \times (-0.17) = -0.17$
Output of hidden neuron 1	$1 / (1 + \exp(-0.55)) = 0.634$
Output of hidden neuron 2	$1 / (1 + \exp(+0.17)) = 0.458$

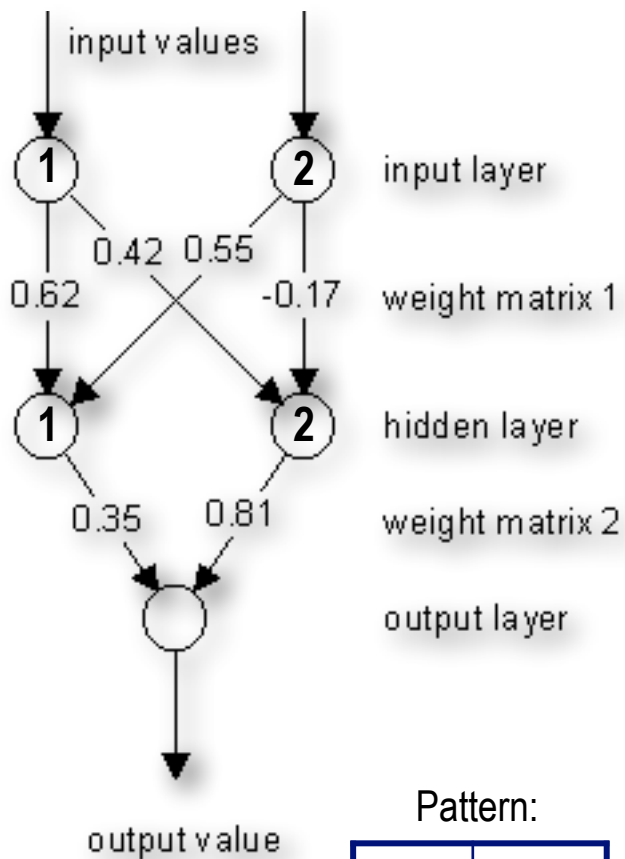
Calculations are rounded to the third decimal place.

The learning process

Backward propagation

Example.

Suppose we have the following three-layer Multi-Layer Perceptron:



(2) The output layer neuron is now activated:

Output neuron's input	$0.634 \times 0.35 + 0.458 \times 0.81 = 0.593$
Output neuron's output	$1 / (1 + \exp(-0.593)) = 0.644$
Output error	$0 - 0.644 = -0.644$

Calculations are rounded to the third decimal place.

Pattern:

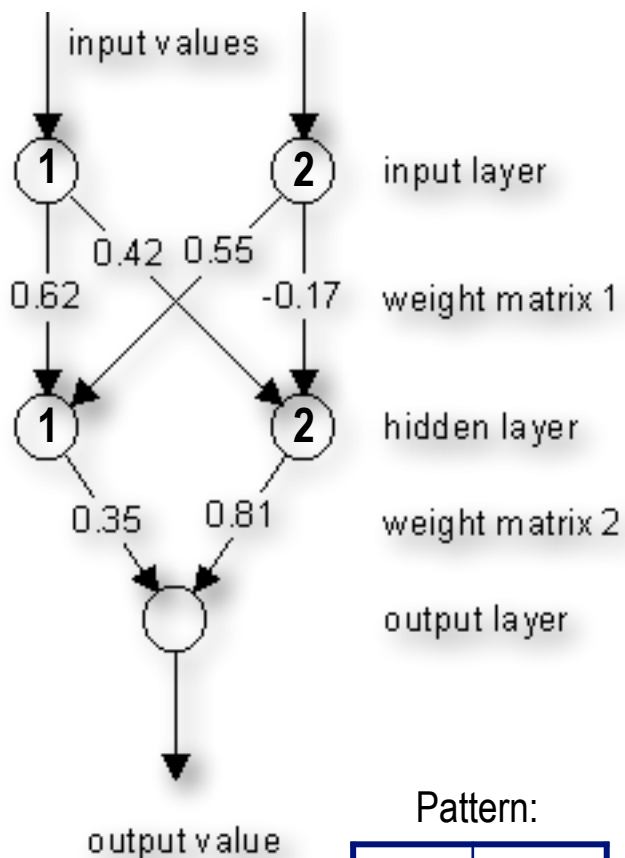
input	target
0 1	0
1 1	1

The learning process

Backward propagation

Example.

Suppose we have the following three-layer Multi-Layer Perceptron:



Pattern:

input	target
0 1	0
1 1	1

(3) After getting the output error, we can perform backpropagation. We start by modifying the weights of matrix 2:

Change value of weight 1	$0.25 \times (-0.644) \times 0.634 \times 0.644 \times (1-0.644) = -0.023$
Change value of weight 2	$0.25 \times (-0.644) \times 0.458 \times 0.644 \times (1-0.644) = -0.017$
Weight 1 modification	$0.35 + (-0.023) = 0.327$
Weight 2 modification	$0.81 + (-0.017) = 0.793$

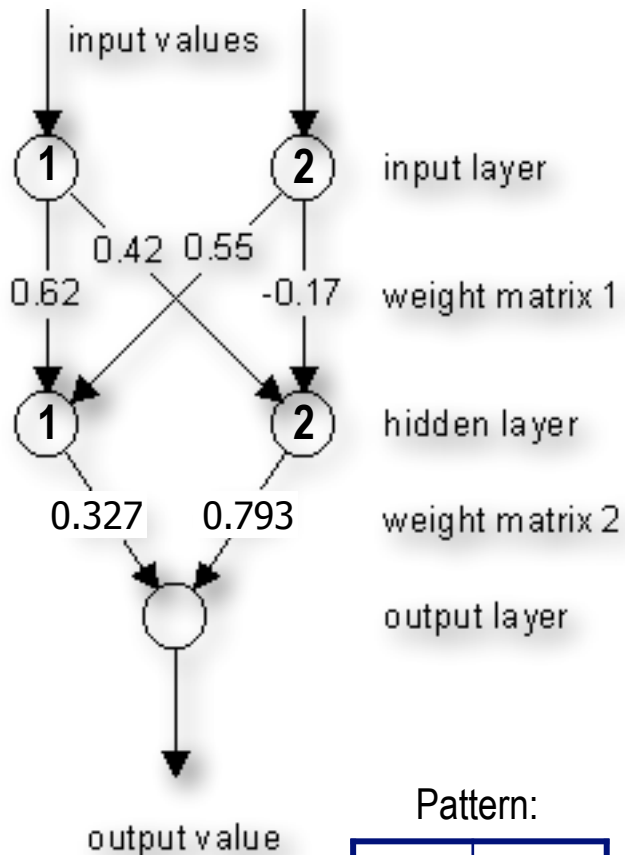
Calculations are rounded to the third decimal place.

The learning process

Backward propagation

Example.

Suppose we have the following three-layer Multi-Layer Perceptron:



Pattern:

input	target
0 1	0
1 1	1

(4) Now we can update the weights of matrix 1:

Change value of weight 1	$0.25 \times (-0.644) \times 0 \times 0.634$ $\times (1-0.634) = 0$
Change value of weight 2	$0.25 \times (-0.644) \times 0 \times 0.458$ $\times (1-0.458) = 0$
Change value of weight 3	$0.25 \times (-0.644) \times 1 \times 0.634$ $\times (1-0.634) = -0.037$
Change value of weight 4	$0.25 \times (-0.644) \times 1 \times 0.458$ $\times (1-0.458) = -0.040$
Weight 1 modification	$0.62 + 0 = 0.62$ (immutato)
Weight 2 modification	$0.42 + 0 = 0.42$ (immutato)
Weight 3 modification	$0.55 + (-0.037) = 0.513$
Weight 4 modification	$-0.17 + (-0.040) = -0.210$

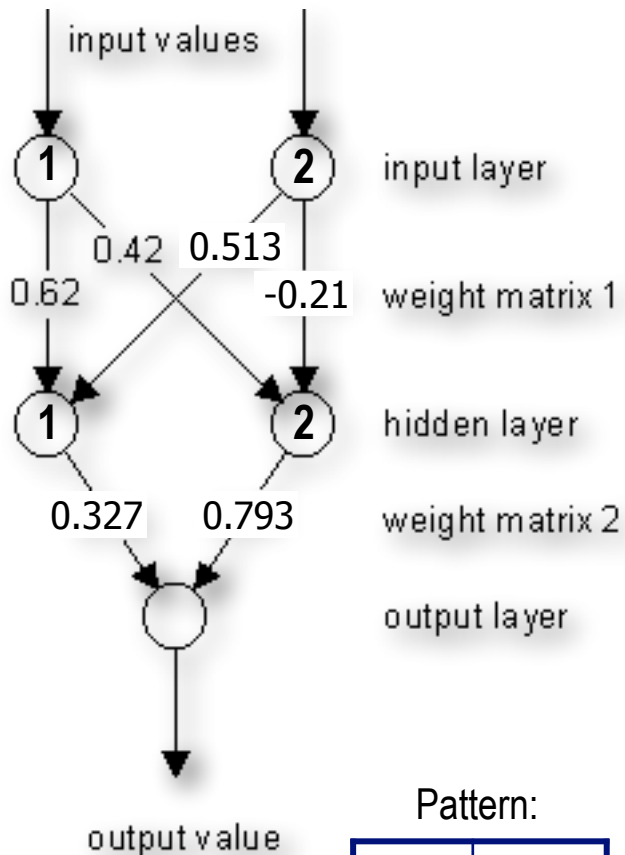
Calculations are rounded to the third decimal place.

The learning process

Backward propagation

Example.

Suppose we have the following three-layer Multi-Layer Perceptron:



Pattern:

input	target
0 1	0
1 1	1

(5) The first input pattern was propagated through the network.

- The same procedure will be used for the next input pattern, this time with the values of the weights changed.
- After forward and backward propagation of the second input, a learning step (or "epoch") is completed and then the network error can be calculated by averaging the output quadratic errors.
- By performing this procedure several times, the error becomes smaller and smaller.
- The algorithm terminates correctly if the network error is zero (perfect situation) or nearly so.
- Note that this algorithm is also applicable for Multi-Layer Perceptron with more than one hidden layer.

The learning process

Backward propagation

What happens if all values in an input pattern are zero?



If all values of an input pattern are zero, the weights of matrix 1 will never be changed for it and the network will not learn mails that pattern.

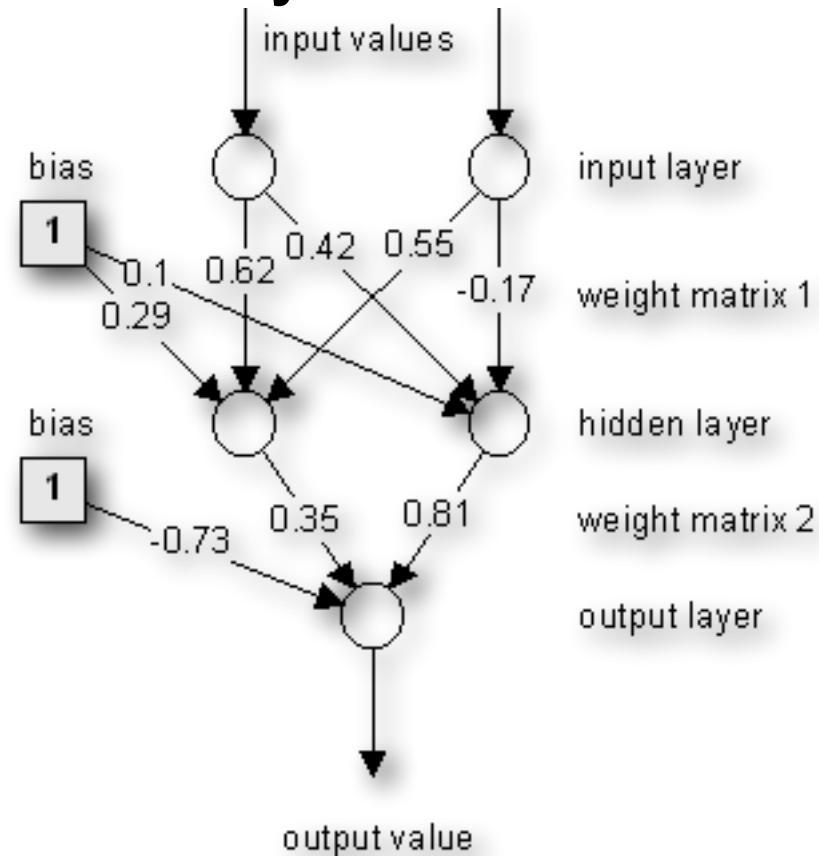
To avoid this, a "pseudo input" called **Bias** is created with a constant value of 1.

This changes the structure of the network in the following way:

The learning process

Backward propagation

Three layer MLP with Bias



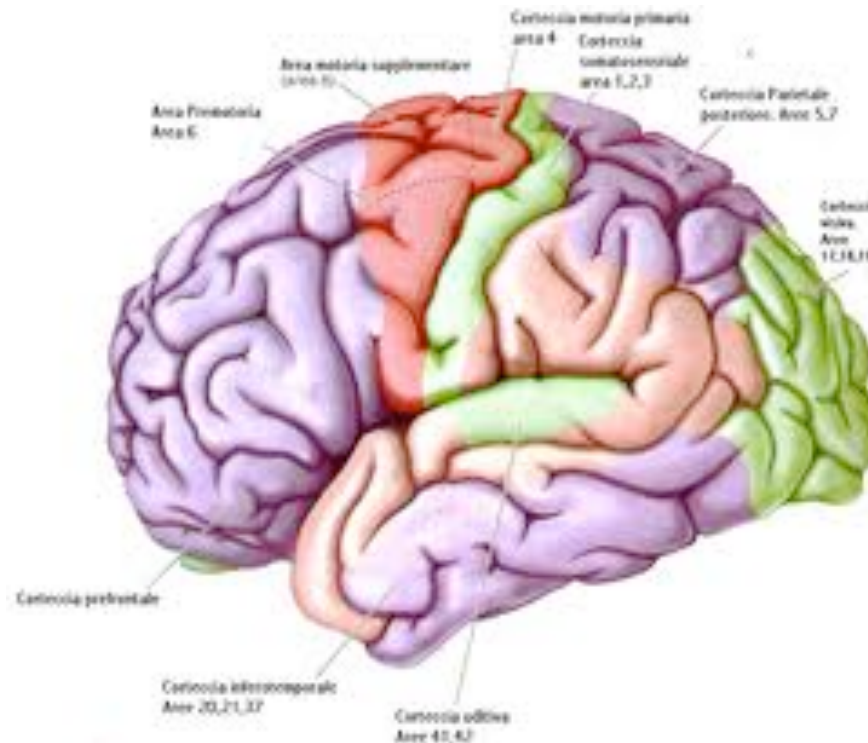
- ▶ The **additional weights**, associated with the input Biases to the neurons of the hidden and output layers, have random initial values and are updated similarly to the others.
- ▶ By sending a constant value 1 to the neurons, it is guaranteed that their input values are always different from zero.

The learning process

Self organization

“**Self organization**” is an unsupervised learning algorithm used by neural networks with a Kohonen Feature Map.

As mentioned in the previous paragraphs, a neural network tries to mimic the human biological brain, and self-organization is probably the best way to realize this.



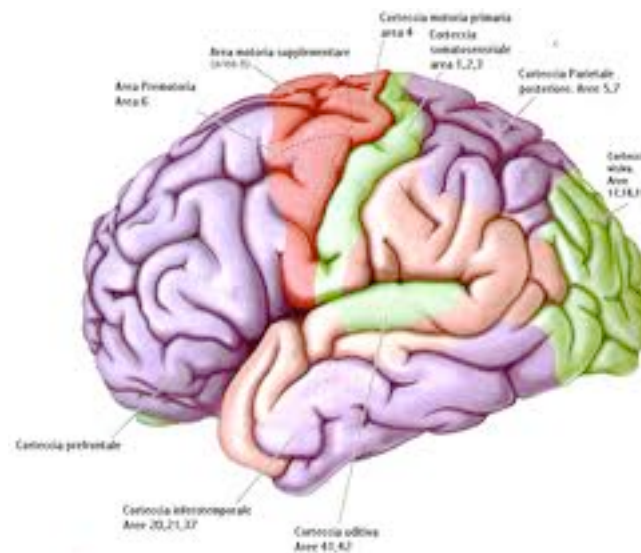
The learning process

Self organization

It is commonly known that the cortex of the human brain is divided into **several regions**, each responsible for certain functions. Neural cells organize themselves into **groups** according to incoming information.

Input information is not only received by individual neural cells, but also affects other nearby cells.

This organization results in a kind of **map**, in which neural cells with similar functions are arranged in homogeneous groupings (clusters).



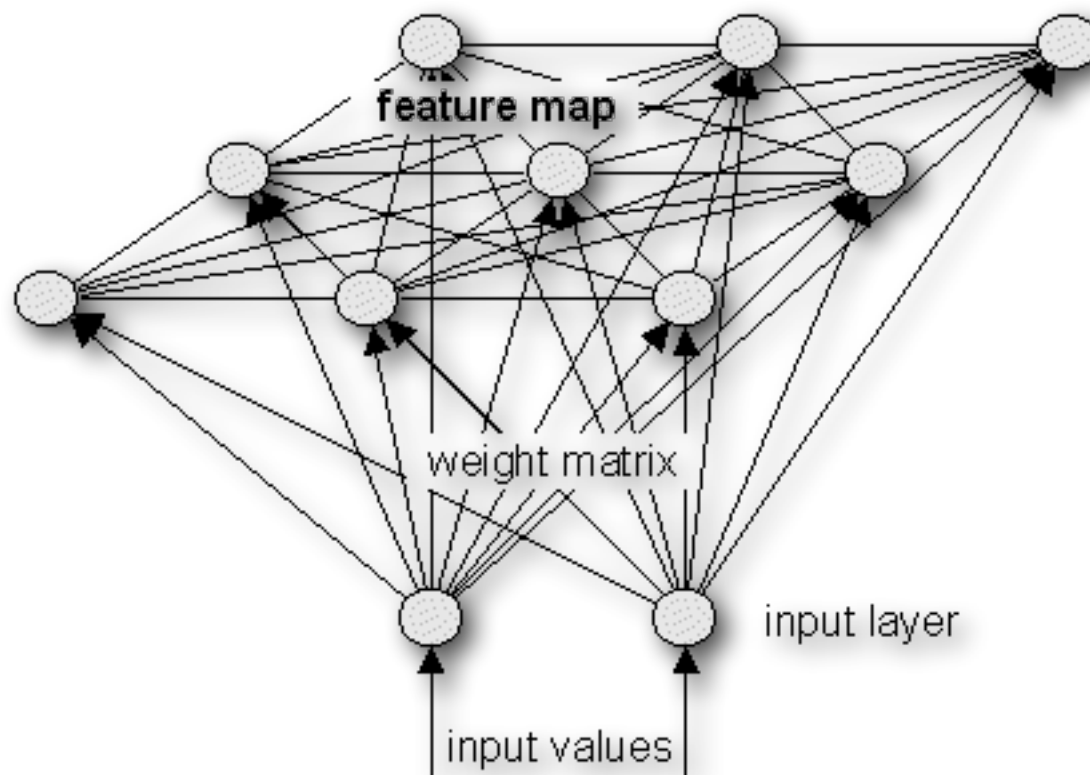
The learning process

Self organization

This self-organizing process can also be performed by a neural network.

This type of neural network is mostly used for **classification**, because similar input values are clustered in areas of the map.

A **typical structure** of a Kohonen map using the self-organization algorithm is as follows:

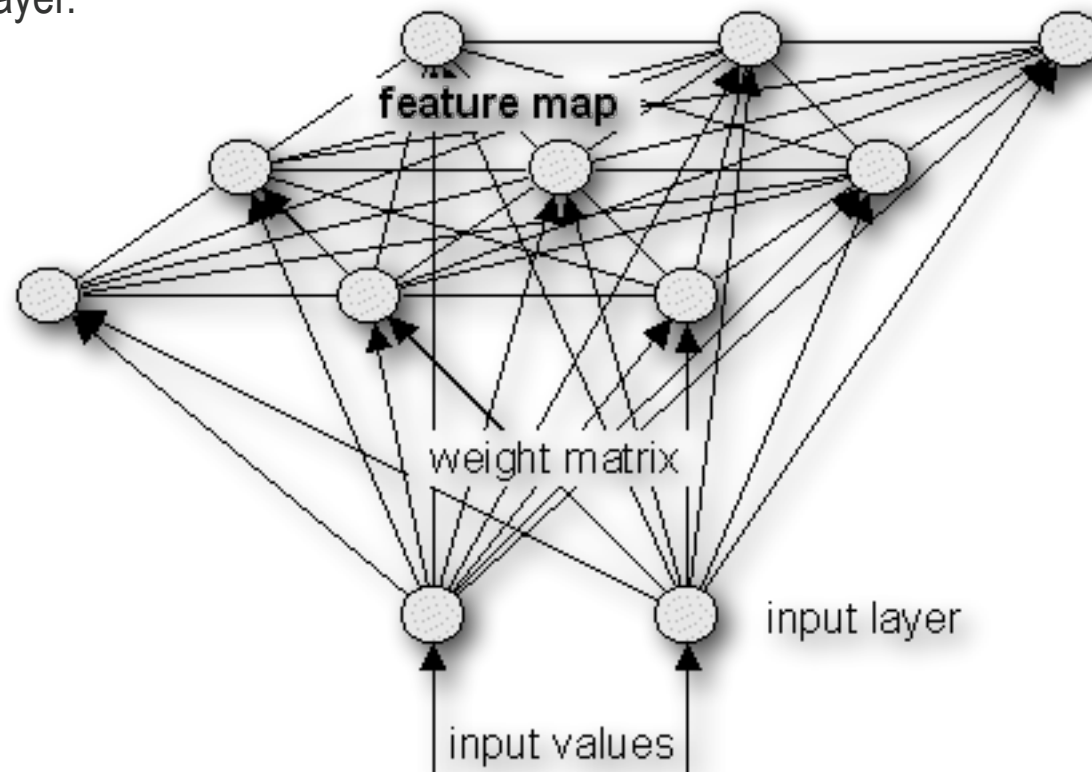


Kohonen Feature Map with two-dimensional input and two-dimensional map (3x3 neurons)

The learning process

Self organization

- As you can see, **each neuron in the input layer is connected to each neuron on the map.**
- The resulting matrix of weights is used to propagate the network input values to the neurons in the map.
- In addition, **all neurons in the map are connected to each other.** These connections are used to influence the neurons in a certain area of activation around the neuron with the maximum activation, received from the output of the input layer.



Kohonen Feature Map with two-dimensional input and two-dimensional map (3x3 neurons)

The learning process

Self organization

The amount of feedback between neurons in the map is usually calculated using the **Gauss function**:

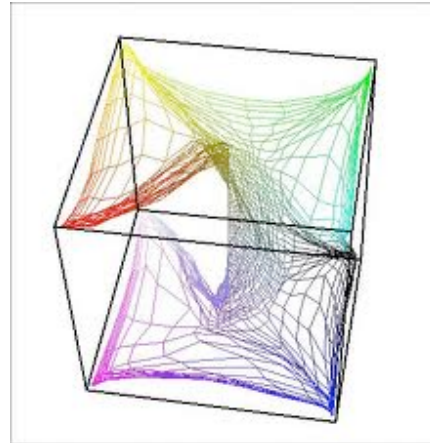
$$\text{feedback}_{ci} = e^{\frac{-|x_c - x_i|^2}{2r^2}}$$

where

- x_c is the position of the most activated neuron (centroid);
- x_i are the positions of the other neurons in the map;
- r is the activation area (radius).

The learning process

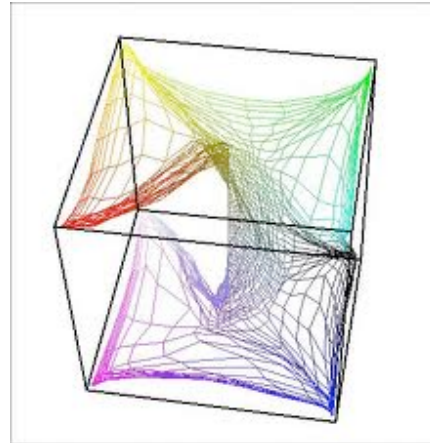
Self organization



- Initially, the **activation area** is large as is the feedback between neurons in the map.
- This results in activation of neurons in a large area around the **most active neuron**.
- As learning progresses, the area of activation is steadily decreased and only the neurons closest to the center of activation are affected by the more active neuron.

The learning process

Self organization



- Unlike the biological model, **map neurons do not change their position on the map.**
- The organization is simulated by changing the values in the weight matrix (in the same way as other neural networks).

Because this is an unsupervised learning algorithm, there are no pre-existing input/target patterns.

The input values to the network are taken from a predefined range, and represent the "data" that needs to be organized.

The learning process

Self organization

The "self organization" algorithm works as follows.

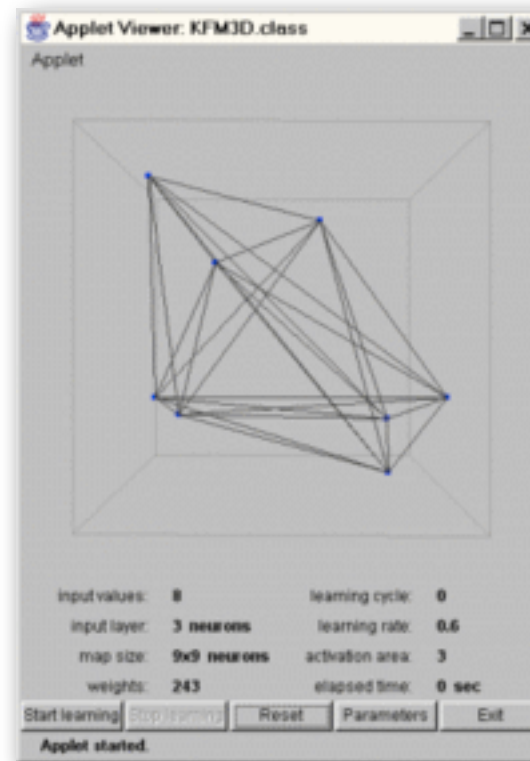
1. Define the range of input values
2. Set all weights to random values taken from the range of input values
3. Define the initial area (radius) of activation
4. Take a random value from the input set and pass it to the neurons in the input layer
5. Determine the most active neuron on the map:
 - *multiply the output of the input level with the values of the weights;*
 - *the map neuron with the largest resulting value is said to be "the most active";*
 - *calculate the feedback value of every other map neuron using the Gauss function.*
6. Get the new values of the weights w_{i+1} with the formula:
$$w_{i+1} = w_i + \text{feedback} \times (\text{input value} - w_i) \times \text{learning rate}$$
7. Reduce the activation area
8. Return to step 4
9. The algorithm terminates if the activation area is smaller than a specified value

The learning process

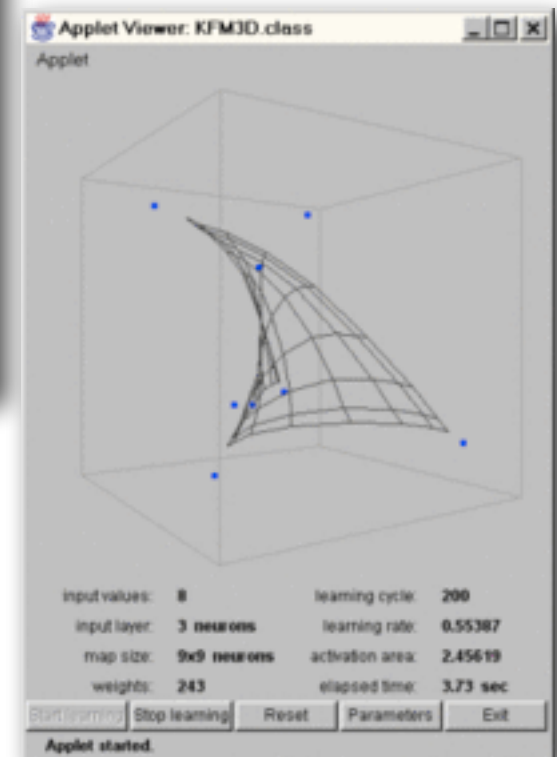
Self organization

Example: see [animation](#).

- ▶ The Kohonen Feature Map in the example has three neurons in its input layer that represent the values of the spatial dimensions x, y, and z.
- ▶ The feature map is initially two-dimensional, and has 9×9 neurons.
- ▶ The resulting matrix has $3 \times 9 \times 9 = 243$ weights, because each input neuron is connected to each neuron in the map.



Animation home screens

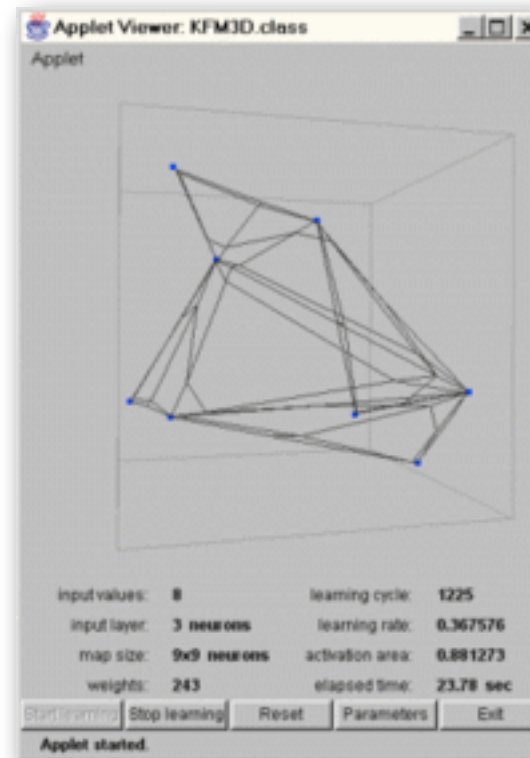


The learning process

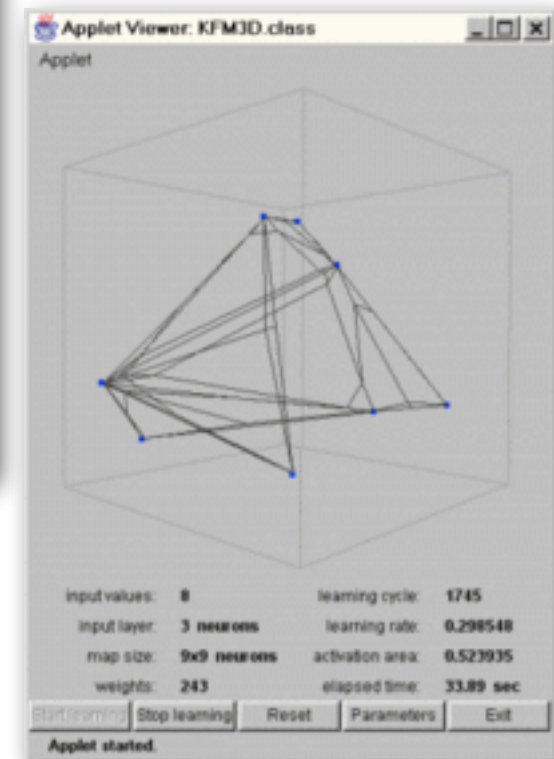
Self organization

Example: see [animation](#).

- ▶ As learning progresses, the map becomes more and more structured.
- ▶ From the animation you can see that the neurons in the map "try" to get as close as possible to their nearest input value (blue dot).



Animation home screens

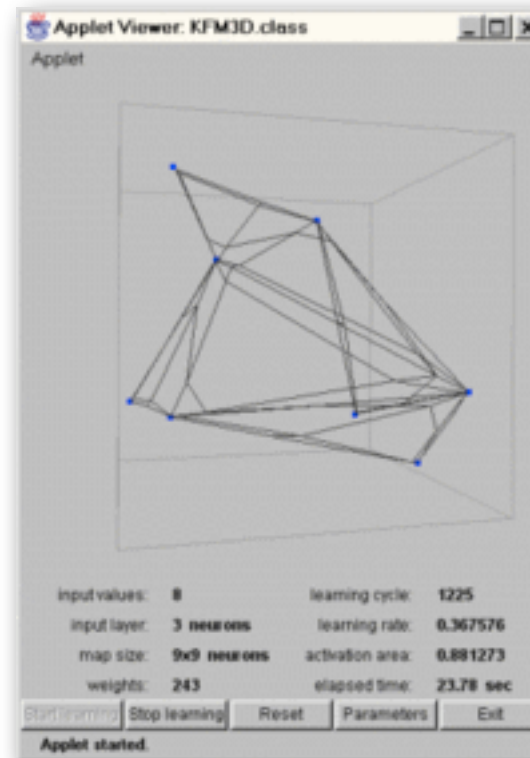


The learning process

Self organization

Example: see [animation](#).

- ▶ At the end of the learning process, the feature map "latches on" to all input values (blue dots).
- ▶ The grid is not geometrically regular because the neurons inside the map also try to approach the input points.
- ▶ This leads to a distorted view of the grid.



Animation home screens

