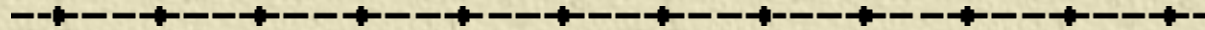
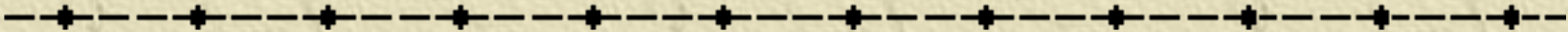


Artificial Neural Networks

- Introduction -



Overview

- 
1. Biological inspiration
 2. Artificial neurons and neural networks
 3. Learning processes
 4. Learning with artificial neural networks

Biological inspiration

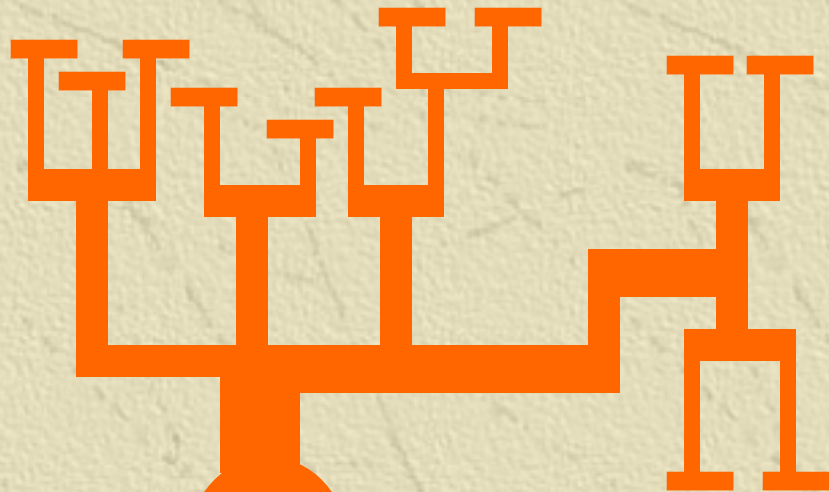
Animals are able to react adaptively to changes in their external and internal environment, and they use their nervous system to perform these behaviours.

An appropriate model/simulation of the nervous system should be able to produce similar responses and behaviours in artificial systems.

The nervous system is build by relatively simple units, the neurons, so copying their behavior and functionality should be the solution.

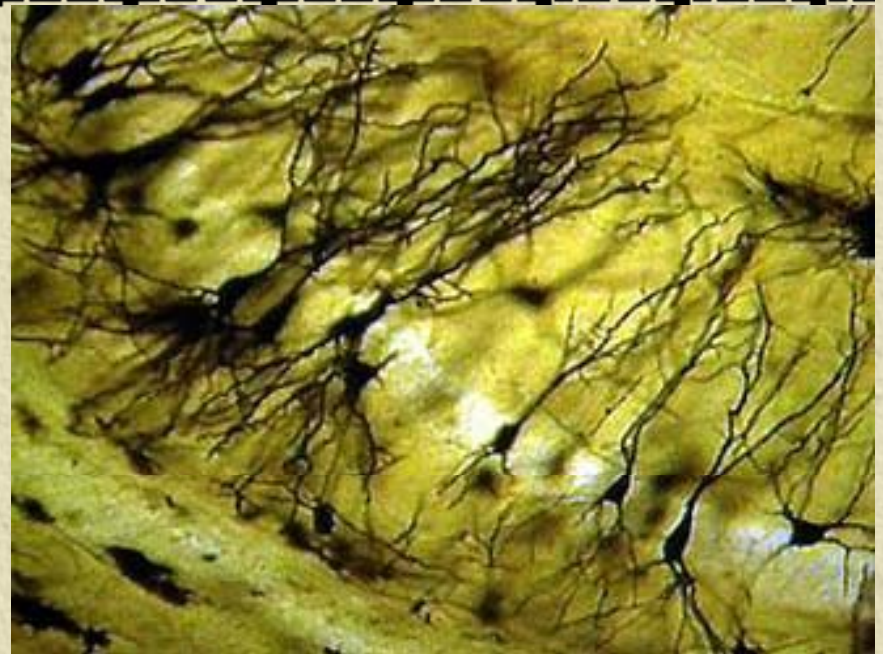
Biological inspiration

Dendrites

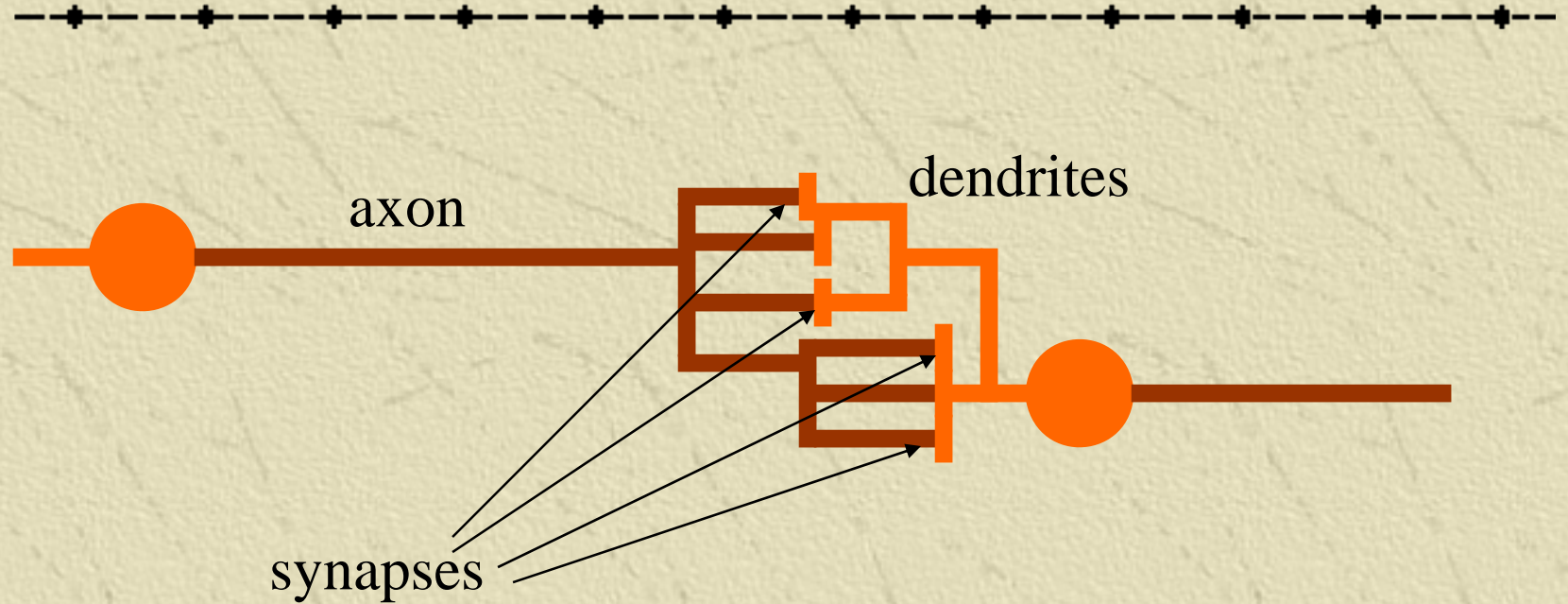


Soma (cell body)

Axon



Biological inspiration



The information transmission happens at the synapses.

Biological inspiration



The spikes travelling along the axon of the pre-synaptic neuron trigger the release of neurotransmitter substances at the synapse.

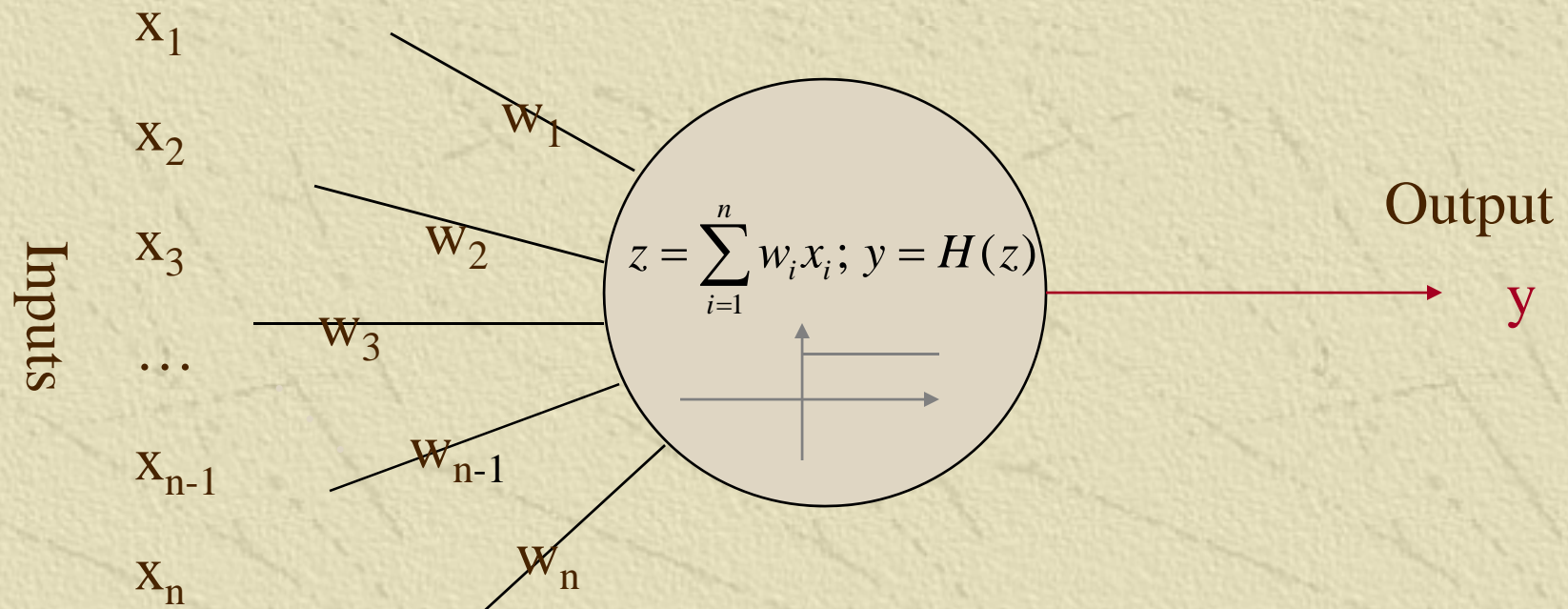
The neurotransmitters cause excitation or inhibition in the dendrite of the post-synaptic neuron.

The integration of the excitatory and inhibitory signals may produce spikes in the post-synaptic neuron.

The contribution of the signals depends on the strength of the synaptic connection.

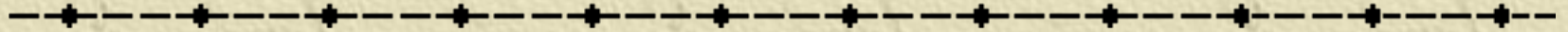
Artificial neurons

Neurons work by processing information. They receive and provide information in form of spikes.



The McCulloch-Pitts model

Artificial neurons



The McCulloch-Pitts model:

- spikes are interpreted as spike rates;
- synaptic strength are translated as synaptic weights;
- excitation means positive product between the incoming spike rate and the corresponding synaptic weight;
- inhibition means negative product between the incoming spike rate and the corresponding synaptic weight;

Artificial neurons

Nonlinear generalization of the McCulloch-Pitts neuron:

$$y = f(x, w)$$

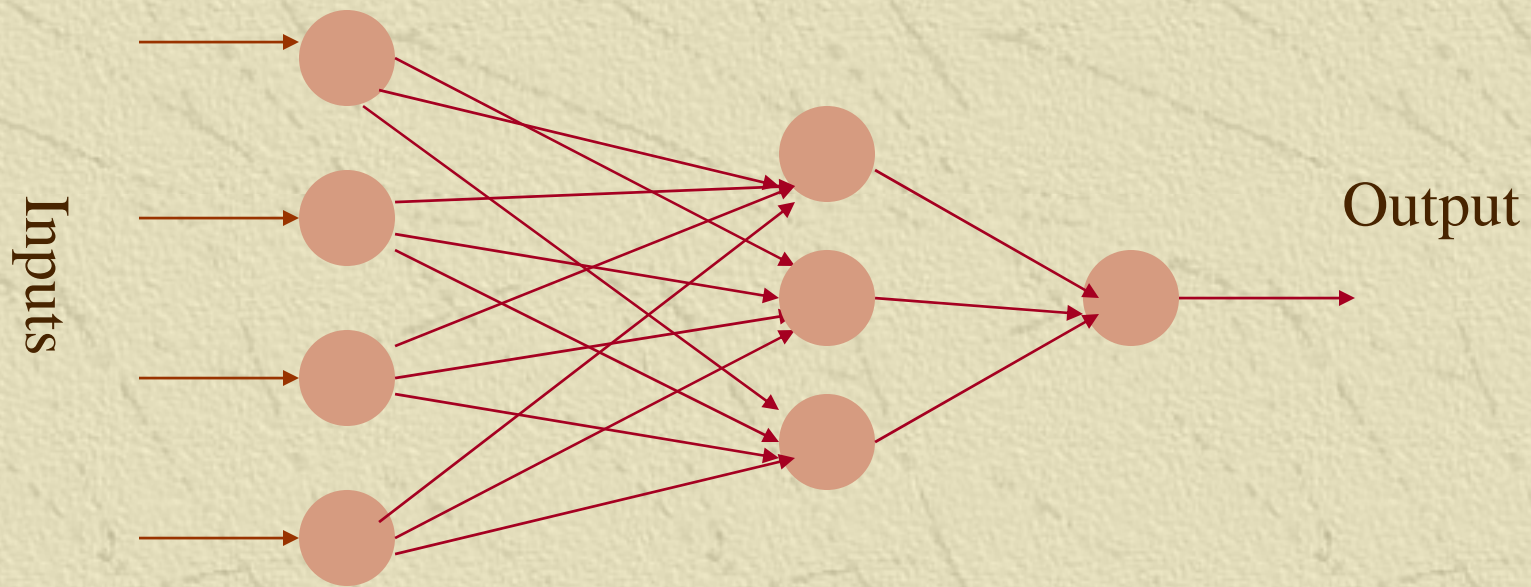
y is the neuron's output, x is the vector of inputs, and w is the vector of synaptic weights.

Examples:

$$y = \frac{1}{1 + e^{-w^T x - a}} \quad \text{sigmoidal neuron}$$

$$y = e^{-\frac{\|x - w\|^2}{2a^2}} \quad \text{Gaussian neuron}$$

Artificial neural networks



An artificial neural network is composed of many artificial neurons that are linked together according to a specific network architecture. The objective of the neural network is to transform the inputs into meaningful outputs.

Artificial neural networks



Tasks to be solved by artificial neural networks:

- controlling the movements of a robot based on self-perception and other information (e.g., visual information);
- deciding the category of potential food items (e.g., edible or non-edible) in an artificial world;
- recognizing a visual object (e.g., a familiar face);
- predicting where a moving object goes, when a robot wants to catch it.

Learning in biological systems

Learning = learning by adaptation

The young animal learns that the green fruits are sour, while the yellowish/reddish ones are sweet. The learning happens by adapting the fruit picking behavior.

At the neural level the learning happens by changing of the synaptic strengths, eliminating some synapses, and building new ones.

Learning as optimisation



The objective of adapting the responses on the basis of the information received from the environment is to achieve a better state. E.g., the animal likes to eat many energy rich, juicy fruits that make its stomach full, and makes it feel happy.

In other words, the objective of learning in biological organisms is to optimise the amount of available resources, happiness, or in general to achieve a closer to optimal state.

Learning in biological neural networks

The learning rules of Hebb:

- synchronous activation increases the synaptic strength;
- asynchronous activation decreases the synaptic strength.

These rules fit with energy minimization principles.

Maintaining synaptic strength needs energy, it should be maintained at those places where it is needed, and it shouldn't be maintained at places where it's not needed.

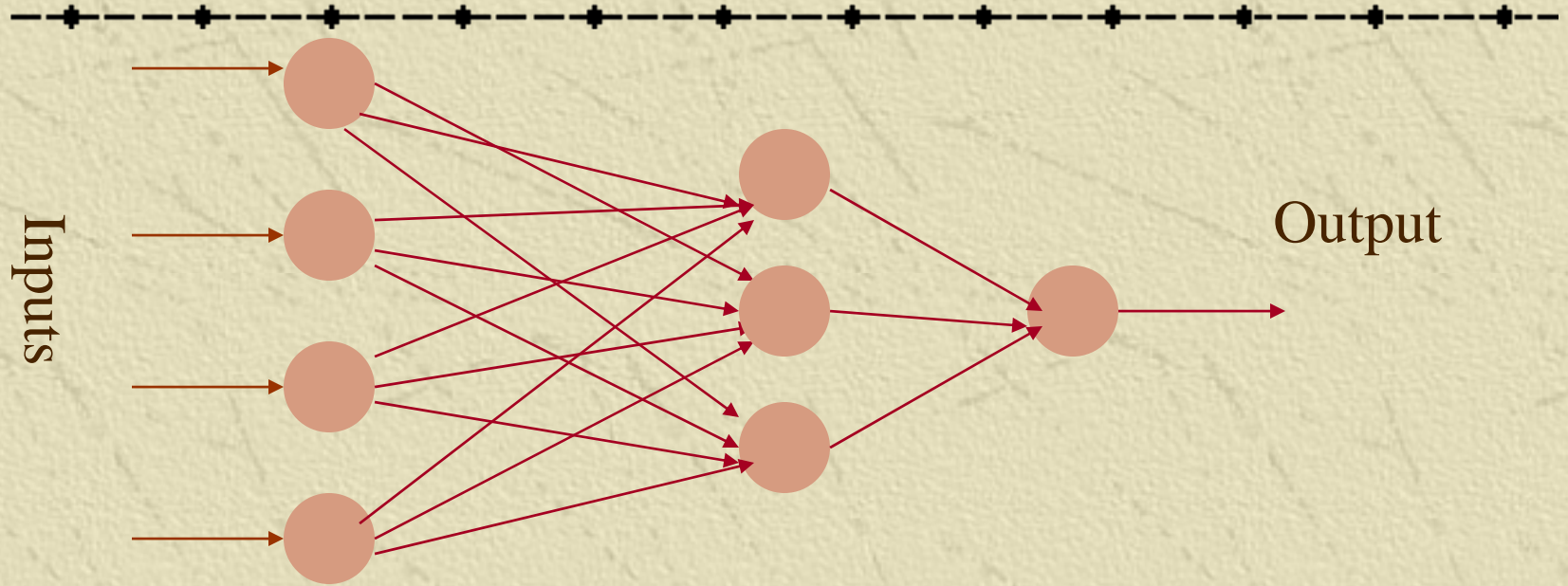
Learning principle for artificial neural networks

ENERGY MINIMIZATION

We need an appropriate definition of energy for artificial neural networks, and having that we can use mathematical optimisation techniques to find how to change the weights of the synaptic connections between neurons.

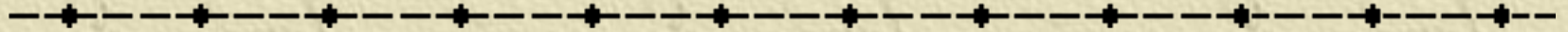
ENERGY = measure of task performance error

Neural network mathematics



$$\begin{aligned} y_1^1 &= f(x_1, w_1^1) \\ y_2^1 &= f(x_2, w_2^1) \\ y_3^1 &= f(x_3, w_3^1) \\ y_4^1 &= f(x_4, w_4^1) \end{aligned} \quad y^1 = \begin{pmatrix} y_1^1 \\ y_2^1 \\ y_3^1 \\ y_4^1 \end{pmatrix} \quad \begin{aligned} y_1^2 &= f(y^1, w_1^2) \\ y_2^2 &= f(y^1, w_2^2) \\ y_3^2 &= f(y^1, w_3^2) \end{aligned} \quad y^2 = \begin{pmatrix} y_1^2 \\ y_2^2 \\ y_3^2 \end{pmatrix} \quad y_{Out} = f(y^2, w_1^3)$$

Neural network mathematics



Neural network: input / output transformation

$$y_{out} = F(x, W)$$

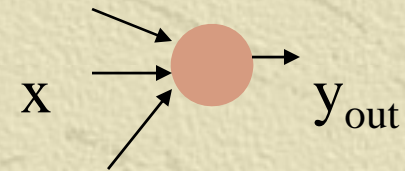
W is the matrix of all weight vectors.

MLP neural networks

MLP = multi-layer perceptron

Perceptron:

$$y_{out} = w^T x$$



MLP neural network:

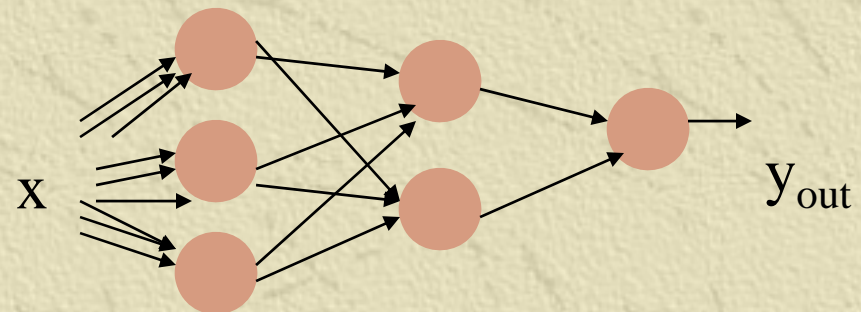
$$y_k^1 = \frac{1}{1 + e^{-w^{1kT} x - a_k^1}}, k = 1, 2, 3$$

$$y^1 = (y_1^1, y_2^1, y_3^1)^T$$

$$y_k^2 = \frac{1}{1 + e^{-w^{2kT} y^1 - a_k^2}}, k = 1, 2$$

$$y^2 = (y_1^2, y_2^2)^T$$

$$y_{out} = \sum_{k=1}^2 w_k^3 y_k^2 = w^{3T} y^2$$



RBF neural networks

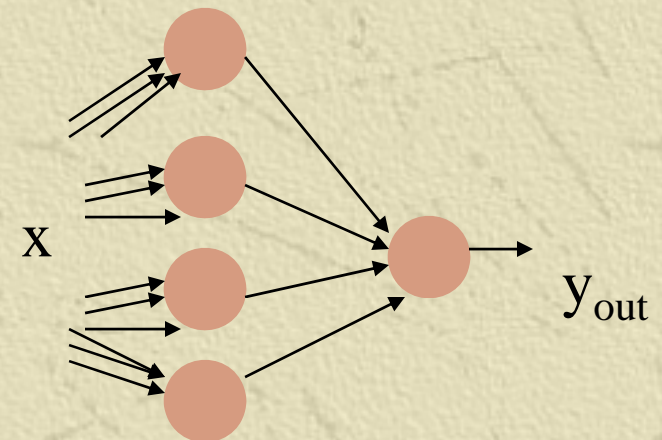
RBF = radial basis function

$$r(x) = \bar{r}(\|x - c\|)$$

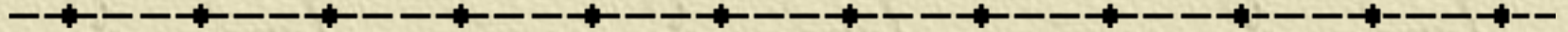
Example: $f(x) = e^{-\frac{\|x-w\|^2}{2a^2}}$

Gaussian RBF

$$y_{out} = \sum_{k=1}^4 w_k^2 \cdot e^{-\frac{\|x-w^{1,k}\|^2}{2(a_k)^2}}$$



Neural network tasks



- control
- classification
- prediction
- approximation

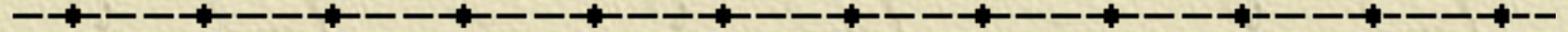
These can be reformulated
in general as

**FUNCTION
APPROXIMATION**

tasks.

Approximation: given a set of values of a function $g(x)$ build a neural network that approximates the $g(x)$ values for any input x .

Neural network approximation



Task specification:

Data: set of value pairs: (x^t, y_t) , $y_t = g(x^t) + z_t$; z_t is random measurement noise.

Objective: find a neural network that represents the input / output transformation (a function) $F(x, W)$ such that

$F(x, W)$ approximates $g(x)$ for every x

Learning to approximate

Error measure:

$$E = \frac{1}{N} \sum_{t=1}^N (F(x_t; W) - y_t)^2$$

Rule for changing the synaptic weights:

$$\Delta w_i^j = -c \cdot \frac{\partial E}{\partial w_i^j} (W)$$

$$w_i^{j, new} = w_i^j + \Delta w_i^j$$

c is the learning parameter (usually a constant)

Learning with a perceptron

Perceptron: $y_{out} = w^T x$

Data: $(x^1, y_1), (x^2, y_2), \dots, (x^N, y_N)$

Error: $E(t) = (y(t)_{out} - y_t)^2 = (w(t)^T x^t - y_t)^2$

Learning:

$$w_i(t+1) = w_i(t) - c \cdot \frac{\partial E(t)}{\partial w_i} = w_i(t) - c \cdot \frac{\partial (w(t)^T x^t - y_t)^2}{\partial w_i}$$

$$w_i(t+1) = w_i(t) - c \cdot (w(t)^T x^t - y_t) \cdot x_i^t$$

$$w(t)^T x = \sum_{j=1}^m w_j(t) \cdot x_j^t$$

A perceptron is able to learn a linear function.

Learning with RBF neural networks

RBF neural network: $y_{out} = F(x, W) = \sum_{k=1}^M w_k^2 \cdot e^{-\frac{\|x - w^{1,k}\|^2}{2(a_k)^2}}$

Data: $(x^1, y_1), (x^2, y_2), \dots, (x^N, y_N)$

Error: $E(t) = (y(t)_{out} - y_t)^2 = \left(\sum_{k=1}^M w_k^2(t) \cdot e^{-\frac{\|x^t - w^{1,k}\|^2}{2(a_k)^2}} - y_t \right)^2$

Learning: $w_i^2(t+1) = w_i^2(t) - c \cdot \frac{\partial E(t)}{\partial w_i^2}$

$$\frac{\partial E(t)}{\partial w_i^2} = 2 \cdot (F(x^t, W(t)) - y_t) \cdot e^{-\frac{\|x^t - w^{1,i}\|^2}{2(a_i)^2}}$$

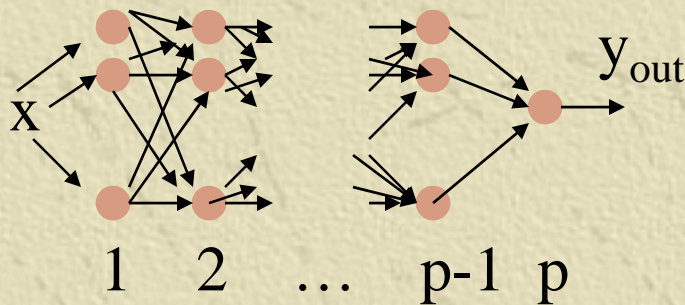
Only the synaptic weights of the output neuron are modified.

An RBF neural network learns a nonlinear function.

Learning with MLP neural networks

MLP neural network:

with p layers



Data: $(x^1, y_1), (x^2, y_2), \dots, (x^N, y_N)$

Error: $E(t) = (y(t)_{out} - y_t)^2 = (F(x^t; W) - y_t)^2$

$$y_k^1 = \frac{1}{1 + e^{-w^{1kT} x - a_k^1}}, k = 1, \dots, M_1$$

$$y^1 = (y_1^1, \dots, y_{M_1}^1)^T$$

$$y_k^2 = \frac{1}{1 + e^{-w^{2kT} y^1 - a_k^2}}, k = 1, \dots, M_2$$

$$y^2 = (y_1^2, \dots, y_{M_2}^2)^T$$

...

$$y_{out} = F(x; W) = w^{pT} y^{p-1}$$

It is very complicated to calculate the weight changes.

Learning with backpropagation



Solution of the complicated learning:

- calculate first the changes for the synaptic weights of the output neuron;
- calculate the changes backward starting from layer $p-1$, and propagate backward the local error terms.

The method is still relatively complicated but it is much simpler than the original optimisation problem.

Learning with general optimisation

In general it is enough to have a single layer of nonlinear neurons in a neural network in order to learn to approximate a nonlinear function.

In such case general optimisation may be applied without too much difficulty.

Example: an MLP neural network with a single hidden layer:

$$y_{out} = F(x; W) = \sum_{k=1}^M w_k^2 \cdot \frac{1}{1 + e^{-w^{1,kT} x - a_k}}$$

Learning with general optimisation

-----◆-----
Synaptic weight change rules for the output neuron:

$$w_i^2(t+1) = w_i^2(t) - c \cdot \frac{\partial E(t)}{\partial w_i^2}$$

$$\frac{\partial E(t)}{\partial w_i^2} = 2 \cdot (F(x^t, W(t)) - y_t) \cdot \frac{1}{1 + e^{-w^{1,iT} x^t - a_i}}$$

Synaptic weight change rules for the neurons of the hidden layer:

$$w_j^{1,i}(t+1) = w_j^{1,i}(t) - c \cdot \frac{\partial E(t)}{\partial w_j^{1,i}}$$

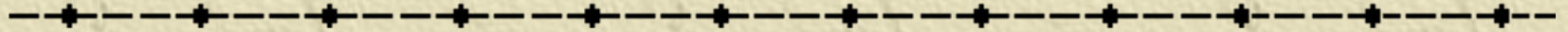
$$\frac{\partial E(t)}{\partial w_j^{1,i}} = 2 \cdot (F(x^t, W(t)) - y_t) \cdot \frac{\partial}{\partial w_j^{1,i}} \left(\frac{1}{1 + e^{-w^{1,iT} x^t - a_i}} \right)$$

$$\frac{\partial}{\partial w_j^{1,i}} \left(\frac{1}{1 + e^{-w^{1,iT} x^t - a_i}} \right) = \frac{e^{-w^{1,iT} x^t - a_i}}{(1 + e^{-w^{1,iT} x^t - a_i})^2} \cdot \frac{\partial}{\partial w_j^{1,i}} (-w^{1,iT} x^t - a_i)$$

$$\frac{\partial}{\partial w_j^{1,i}} (-w^{1,iT} x^t - a_i) = -x_j^t$$

$$w_j^{1,i}(t+1) = w_j^{1,i}(t) - c \cdot 2 \cdot (F(x^t, W(t)) - y_t) \cdot \frac{e^{-w^{1,iT} x^t - a_i}}{(1 + e^{-w^{1,iT} x^t - a_i})^2} \cdot (-x_j^t)$$

New methods for learning with neural networks



Bayesian learning:

the distribution of the neural network parameters is learnt

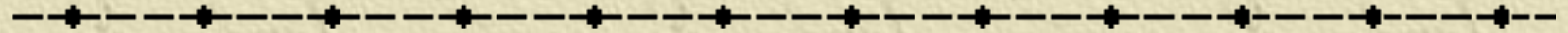
Support vector learning:

the minimal representative subset of the available data is used to calculate the synaptic weights of the neurons

Summary

-
- Artificial neural networks are inspired by the learning processes that take place in biological systems.
 - Artificial neurons and neural networks try to imitate the working mechanisms of their biological counterparts.
 - Learning can be perceived as an optimisation process.
 - Biological neural learning happens by the modification of the synaptic strength. Artificial neural networks learn in the same way.
 - The synapse strength modification rules for artificial neural networks can be derived by applying mathematical optimisation methods.

Summary



- Learning tasks of artificial neural networks can be reformulated as function approximation tasks.
- Neural networks can be considered as nonlinear function approximating tools (i.e., linear combinations of nonlinear basis functions), where the parameters of the networks should be found by applying optimisation methods.
- The optimisation is done with respect to the approximation error measure.
- In general it is enough to have a single hidden layer neural network (MLP, RBF or other) to learn the approximation of a nonlinear function. In such cases general optimisation can be applied to find the change rules for the synaptic weights.