Neuron models
(perceptron, sigmoid neuron, neuron of ADALINE-type, Pade neuron, Quadratic neuron and Sigma-Pi neuron, Hebb neuron, Stochastic model of neuron, WTA-type of neuron)

Perceptron

A neuron is an information-processing unit that is fundamental to the operation of a neural network. We may identify **three basic elements of the neuron model**. The structure of a neuron is depicted in Fig. 3.1.

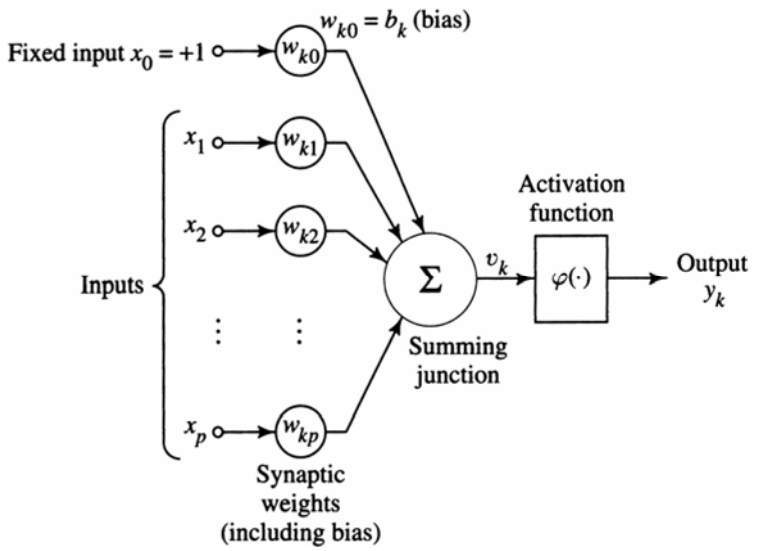


Fig. 3.1. Structure of a neuron model

- **A set of synapses**, each of which is characterized by a weight or strength of its own. Specifically, a signal x_j at the input of synapse j connected to neuron k is multiplied by the synaptic weight w_{kj}. It is important to make a note of the manner in which the subscripts of the synaptic weight w_{kj} are written. The first subscript refers to the neuron in question and the second subscript refers to the input end of the synapse to which the weight refers. The weight w_{kj} is positive if the associated synapse is excitatory; it is negative if the synapse is inhibitory.

The model of a neuron also includes an externally applied **bias** (threshold) $w_{k0} = b_k$ that has the effect of lowering or increasing the net input of the activation function.

- **A summator (an adder)** for summing the input signals, weighted by the respective synapses of a neuron.
- **An activation function** for limiting the amplitude of the output of a neuron. The activation function is also referred to as a squashing function in that it squashes (limits) the permissible amplitude range of the output signal to some finite value. Typically, the normalized amplitude range of the output of a neuron is written as the closed unit interval $[0,1]$ or alternatively $[-1,1]$.

In mathematical terms, a neuron *k* can be described by the following pair of equations:

$$v_k = \sum_{j=0}^{p} w_{kj} x_j \ ,$$

$$y_k = \varphi(v_k)$$

or in a matrix form

$$v_k = \begin{bmatrix} w_{k0} & w_{k1} & \dots & w_{kp} \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ \dots \\ x_p \end{bmatrix} = \mathbf{W}_k^T \mathbf{X},$$

where *T* is the transposition sign.

The simple types of activation functions are threshold and activation function similar to signum one. In these cases, a neuron is referred to as perceptron or McCulloch–Pitts model.

Threshold activation function (McCulloch–Pitts model) is shown in Fig. 3.2 and written as

$$\varphi(v) = \begin{cases} 1, & v \geq 0 \\ 0, & v < 0 \end{cases}.$$

In this model, the output of a neuron takes on the value of 1 if the total internal activity level of that neuron is nonnegative and 0 otherwise.

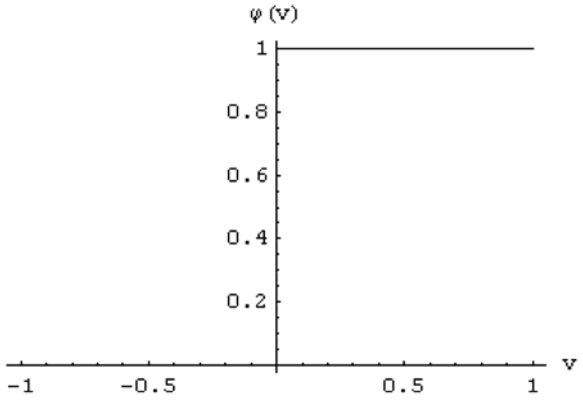


Fig. 3.2. Threshold activation function

McCulloch–Pitts model of a neuron is sometimes used with **activation function similar to signum one** (output signal is equal to either –1 or +1):

$$\varphi(v) = \begin{cases} 1, & v \geq 0; \\ -1, & v < 0. \end{cases}$$

There are various kinds of the activation function. Some of them are depicted in Fig. 3.3.

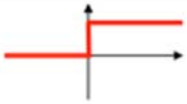
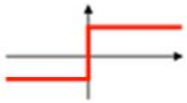
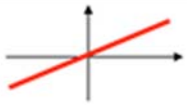
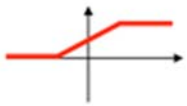
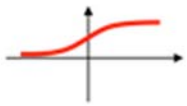
Activation function	Equation	Example	1D Graph
Unit step (Heaviside)	$\phi(z) = \begin{cases} 0, & z < 0, \\ 0.5, & z = 0, \\ 1, & z > 0, \end{cases}$	Perceptron variant	
Sign (Signum)	$\phi(z) = \begin{cases} -1, & z < 0, \\ 0, & z = 0, \\ 1, & z > 0, \end{cases}$	Perceptron variant	
Linear	$\phi(z) = z$	Adaline, linear regression	
Piece-wise linear	$\phi(z) = \begin{cases} 1, & z \geq \frac{1}{2}, \\ z + \frac{1}{2}, & -\frac{1}{2} < z < \frac{1}{2}, \\ 0, & z \leq -\frac{1}{2}, \end{cases}$	Support vector machine	
Logistic (sigmoid)	$\phi(z) = \frac{1}{1 + e^{-z}}$	Logistic regression, Multi-layer NN	
Hyperbolic tangent	$\phi(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$	Multi-layer NN	

Fig. 3.3. Various kinds of the activation function

Sigmoid neuron

A neuron of sigmoidal type has a structure similar to McCulloch-Pitts model with the difference that the activation function is sigmoidal. **Sigmoid functions are characterized by the properties of continuity and differentiability**. Note that the sigmoid function is differentiable, which is an important feature of neural network theory. Sigmoid functions can be expressed in the form of unipolar or bipolar sigmoid function.

Let's consider well-known sigmoid functions, such as **logistic and hyperbolic functions**:

- Logistic activation function (Fig. 3.4, *a*)

$$\varphi(v) = \frac{1}{1+\exp(-v)}$$

Whereas a threshold function assumes the value of 0 or 1, a sigmoid function assumes a continuous range of values form 0 and 1.

The derivative of the sigmoid function is easy to calculate:

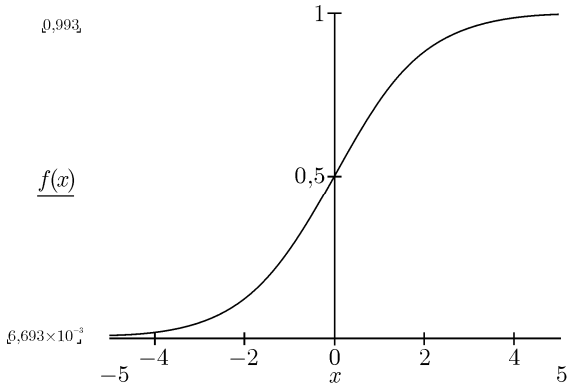
$$\frac{\partial\varphi(v)}{\partial v} = \varphi(v)(1-\varphi(v)).$$

- Hyperbolic tangent function (Fig. 3.4, *b*) is given as

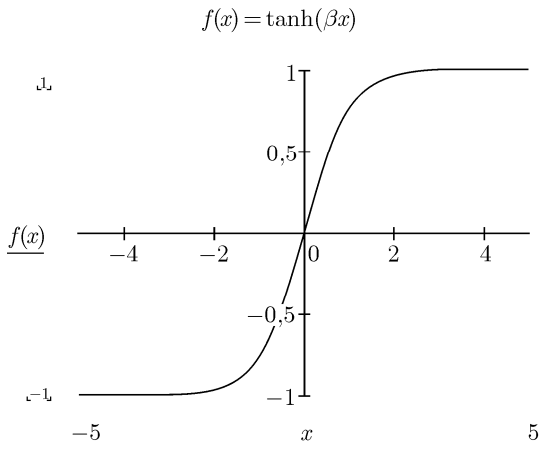
$$\varphi(v) = \tan gh(\frac{v}{2}) = \frac{1-\exp(-v)}{1+\exp(-v)}.$$

Hyperbolic tangent function can be easily expressed in terms of the logistic function: (2* logistic function – 1). Its derivative is also easy to calculate:

$$\frac{\partial\varphi(v)}{\partial v} = \frac{1}{2}(1+\varphi(v))(1-\varphi(v)).$$



a



b

Fig. 3.4. Sigmoid activation functions:
a – logistic activation function,
b – hyperbolic tangent activation function,

Neuron of ADALINE-type

One of the simplest learning neurons is **ADALINE (ADAptive LInear NEuron** or **ADAptive LINear Element)** proposed by B. Widrow and shown in Fig. 3.5.

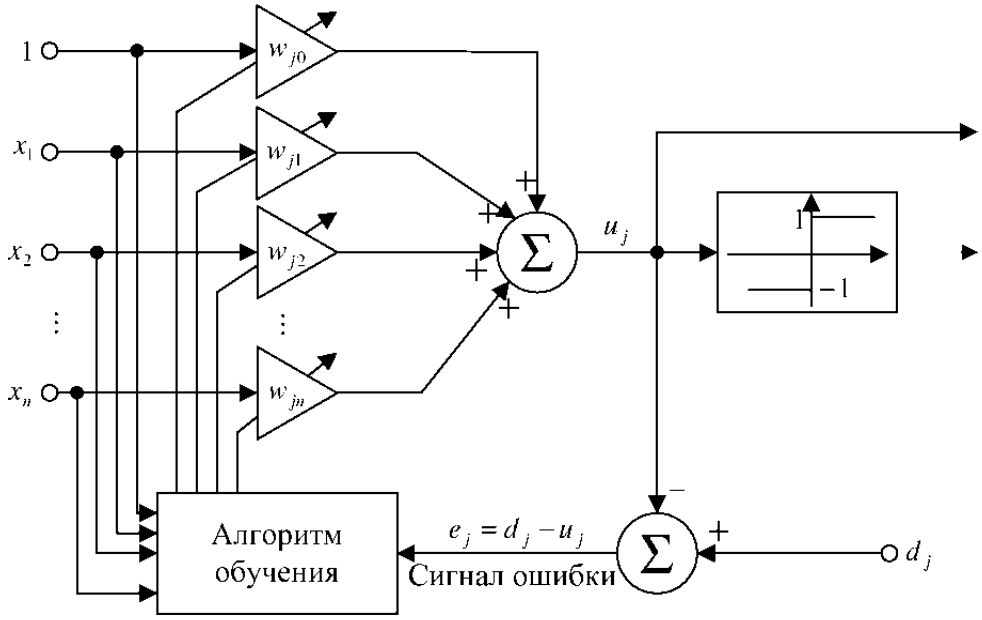


Fig. 3.5. ADALINE structure

ADALINE model is structurally very similar to McCulloch–Pitts neuron with activation function similar to signum one. It consists of two main parts:

– an adaptive linear associator,

– non-linear activation function.

ADALINE model has $(n+1)$ inputs (x_1, x_2, …, x_n) and two outputs (analog u_j and binary y_j). Moreover, there is an additional input to which training signal d_j, showing what should be the desired reaction of the neuron for each specific set of input signals, was given.

Analog output u_j is described by the weighted sum of inputs:

$$u_j = \sum_{i=0}^{n} w_{ji} x_i = \mathbf{W}_j^T \mathbf{X}.$$

Binary output y_j has values of either +1 or –1 in depending on the polarity of analog signal, namely,

$$y_j = \begin{cases} 1, \text{ under } u_j \geq 0; \\ -1, \text{ under } u_j < 0. \end{cases}$$

Analog output signal u_j is compared with external training signal d_j. The emerged signal of error ($e_j = d_j - u_j$) enters the learning algorithm, which changes the synaptic weights for the purpose of minimizing square error written as

$$E(\mathbf{W}_j) = \frac{1}{2} e_j^2 = \frac{1}{2}\left[d_j - \left(\sum_{i=0}^{n} w_{ji} x_i \right) \right]^2.$$

The represented objective function contains linear members in spite of the nonlinear character of ADALINE model.

ADALINE model can be used as both an elementary unit in complicated neural networks (in this case we have MADALINE (Many ADALINE) and an independent unit in the tasks of pattern recognition, signal processing, implementation of logic functions.

Pade neuron

Pade-neuron can be used as an extension of ADALINE neuron in the cases where the linear functions is not enough, particularly in tasks of interpolation of empirical relationships.

Pade neuron calculates the derivative of the fractional-linear function of vector X. Just as for adaptive summator, **numerator and denominator can be represented as the linear functions of vector X**:

$$\frac{U[\mathbf{X}]}{L[\mathbf{X}]},\ U[\mathbf{X}]=\sum_{i=0}^{N}U_i x_i,\ L[\mathbf{X}]=\sum_{i=0}^{N}L_i x_i.$$

In the case of Pade neuron, the square error is defined as

$$E(U,L)=\frac{1}{2}\left(d-\frac{U[\mathbf{X}]}{L[\mathbf{X}]}\right)^2=\frac{e^2}{2},$$

where

$$e=d-\frac{U[\mathbf{X}]}{L[\mathbf{X}]}.$$

The values of weight coefficients are specified by the following formulas:

$$U_{i,j+1}=U_{i,j}+\alpha e\frac{x_i}{\sum_{j=0}^{N}L_j x_j},$$

$$L_{i,j+1}=L_{i,j}+\alpha e\,x_i\frac{\sum_{j=0}^{N}U_j x_j}{\left(\sum_{j=0}^{N}L_j x_j\right)^2}.$$

Quadratic neuron and Sigma-Pi neuron

In some cases, non-linear transformation can also be achieved using a linear associator by special pretreatment of input signals.

The quadratic neuron depicted in Fig. 3.6 calculates the following function

$$y_j=\theta_j+\sum_{l=1}^{n}w_{jl}x_l+\sum_{p=1}^{n}\sum_{l=1}^{n}w_{jpl}x_p x_l$$

by means of a set of elementary blocks (signal multipliers and summators): θ_j, $w_{j1}x_1$, …, $w_{jn}x_n$, $w_{j12}x_1x_2$, …, $w_{jnn}x_n^2$.

It is natural, that the number of synaptic weights should be essentially increased in this case, but the implementation simplicity provides often an advantage of such models, by which the polynomial transformation of any desired degree can be yielded.

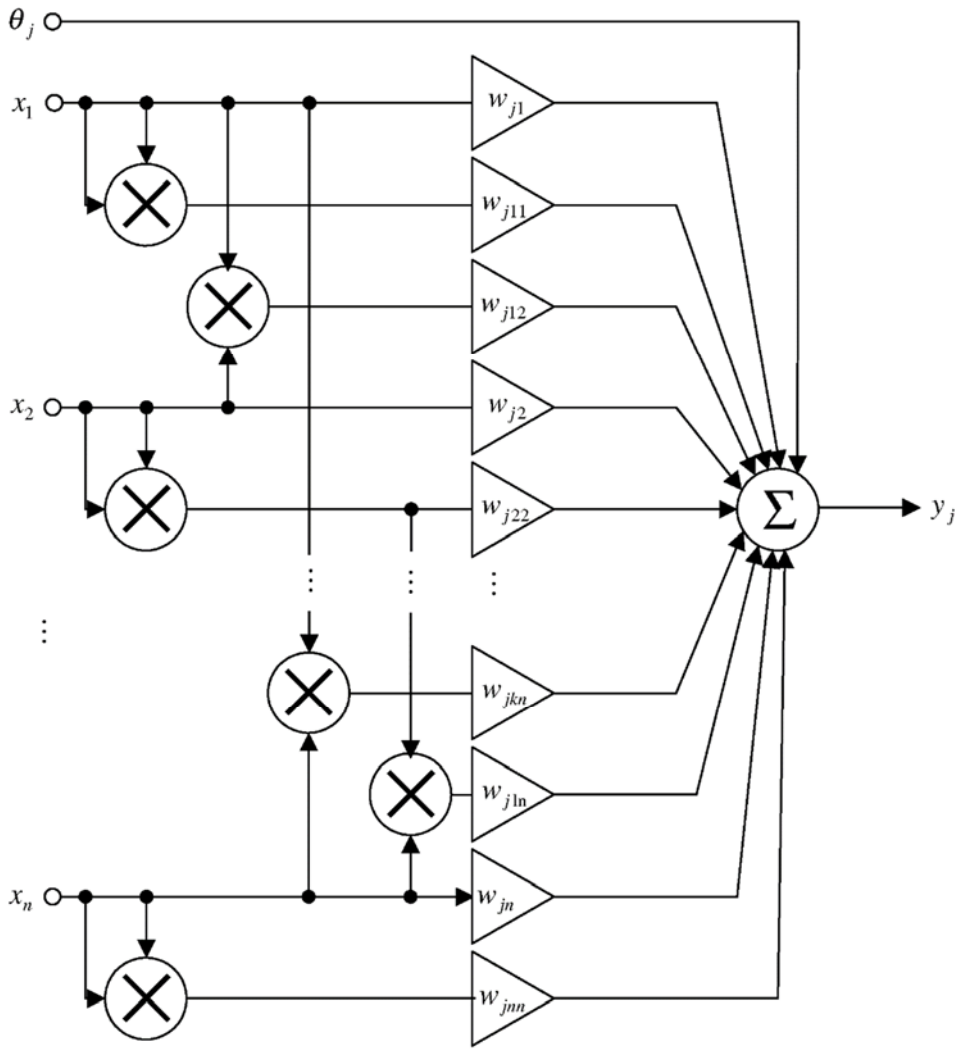


Fig. 3.6. Quadratic neuron

Neurons with linear and quadratic activation functions were above mentioned.

Sigma-Pi neuron is the extension of those neurons on the case of polynomial activation function *U* of the *N*-th degree:

$$U = \sum_{k=1}^{M} w_k \prod_{i \in I_k} x_i ,$$

where I_k is the index set containing one of the possible combinations (2^N) of the first *N* integral numbers, *N* is the number of neuron inputs, $M = 2^N$.

Hebb neuron

In the process of investigating the properties of nerve cells, D. Hebb noted that the connection between two cells is enhanced, when two cells are activated

simultaneously. He suggested the formal learning rule in accordance with **the neuron weight w_i is changed proportionally to the product of its input and output signals.**

Hebb rule can be applied to various types of neural networks with any activation functions of neurons.

The structure of Hebb neuron corresponds to the standard form of McCulloch-Pitts model.

According to Hebbian rule **at every stage of neuron training without a teacher**, the summation of the current weight w_i and its increment Δw_{ij} are given as

$$w_i(t+1) = w_i(t) + \Delta w_i \ ,$$

$$\Delta w_i = \alpha x_i y \ ,$$

where α, $\alpha \in (0,1)$ is the coefficient of learning; y is the output signal of a neuron.

Under learning neuron with a teacher, it is possible to use the following increment

$$\Delta w_i = \alpha x_i y^o \ ,$$

where y^o is the desirable output signal of a neuron.

Using Hebbian rule results in the fact that neuron weights can be arbitrarily high values as the current weight w_i is summarized with its increment. The convergence of the learning process is ensured by the introduction of the forgetting coefficient γ in the equation

$$w_i(t+1) = w_i(t)(1-\gamma) + \Delta w_i \ .$$

The value of γ is selected in the interval $(0,1)$. It is usually equal to some per cents of the learning factor α. The recommended values of the forgetting coefficient correspond to the following inequation $\gamma < 0.1$.

Stochastic model of neuron

In stochastic model, the neuron output state depends not only on the weighted sum of inputs but also on a certain random variable, which values are selected from the interval $(0,1)$.

In stochastic model, the neuron output signal y takes values ± 1 with probabilities:

$$P(y=1) = \frac{1}{1+\exp(-2\beta u)} \ ,$$

$$P(y=-1)=\frac{1}{1+\exp(2\beta u)},$$

where u is the weighted sum of neuron input signals, β is a positive constant (it is usually equal to 1).

The learning process of neuron in stochastic model consists of the following steps:

1. The calculation of the weighted sum

$$u=\sum_{i=0}^{N} w_i x_i$$

for every neurons.

2. The calculation of probability $P(y=\pm 1)$.

3. The generation of random variable $R\in(0,1)$ and the formation of output signal y if $R<P(y)$ or $-y$ if $R>P(y)$.

4. The adaptation of weights w_i under learning with a teacher according to Widrow–Hoff rule is carried out by the formula

$$\Delta w_i=\alpha x_i(d-y).$$

WTA-type of neuron

The input module of WTA (Winner Takes All) model are a group of competitive neurons, receiving the same input signals x_j (Fig. 3.7).

Training WTA model by teacher is not required.

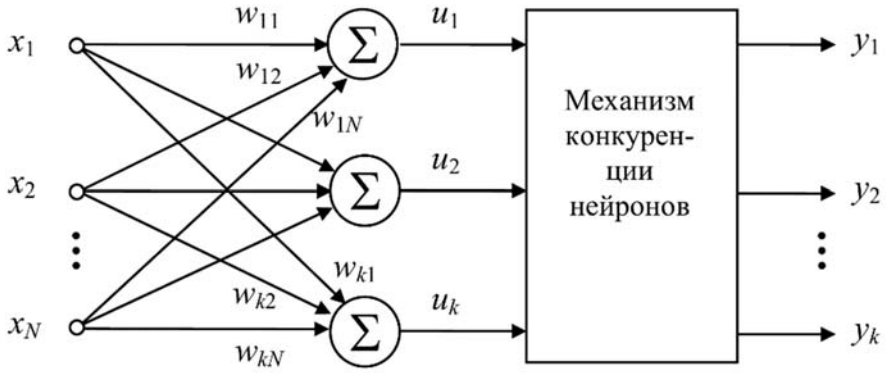


Fig. 3.7. Scheme of WTA neuron connection

At the initial stage, the weight coefficients w_{ij} of every neuron are selected randomly and normalized to 1 by the formula

$$w_{ij} \leftarrow w_{ij} / \left(w_{i1}^2 + w_{i2}^2 + ... + w_{iN}^2 \right)^{1/2}.$$

The components of input vector signal are normalized to 1 by the formula

$$x_j \leftarrow x_j / \left(x_1^2 + x_2^2 + ... + x_N^2 \right)^{1/2}.$$

Then, the neuron-winner is determined according to the following procedure.

1. The output signal of the *i*-th summator is defined by the equation

$$u_i = \sum_{j=0}^{N} w_{ij} x_j .$$

2. By comparing the output values of summators, the neuron-winner with the maximum value u_i is determined. There is the equation

$$y_i = 1$$

at the output of the neuron-winner.

The output signals of rest neurons (neuron-losers) are equal to 0, and the process of correcting their weights is blocked.

3. The neuron-winner has the state 1, and its weight coefficients are corrected by the following rule

$$w_{ij}(t+1) = w_{ij}(t) + \alpha(x_j - w_{ij}(t)).$$

The neuron-losers have the state 0, and their weight coefficients are not corrected.

The consequence of neuronal competition becomes the self-organization of learning process.

It should be noted the following. The output signal on the *i*-th neuron can be written as the vector mapping

$$u_i = \sum_{j=0}^{N} w_{ij} x_j = \mathbf{W}_i^T \mathbf{X} = \|\mathbf{W}_i\| \|\mathbf{X}\| \cos\phi_i .$$

As $\|\mathbf{W}_i\| = \|\mathbf{X}\| = 1$, the value u_i depends on an angle between the input vector $\mathbf{X}$ and the coefficient vector $\mathbf{W}_i$. The neuron-winner possesses coefficient vector, which is the nearest the current input vector $\mathbf{X}$.

WTA model is most often used for classifying vectors.