Neural Networks with Time Delay

Schoel, Gunnar; Puschmann, Peter, FHTW-Berlin

The structure of Neurons and Neural Networks with time delay and an algorithm to learn weight and delay of such networks.

Abstract

There are different Neural Networks distinguished by the learning algorithms, topologys or the number of layers. We will discuss Neural Networks from an other point of view. We divide them in networks without and with time delay. There are some very interesting properties in networks with internal memory. The paper introduces a network with Neurons that many memory cells and an algorithm to learn the weights and delays.

Properties of Neural Networks with internal memory

Time delay networks map bijektiv time functions of the input space $\mathbf{R}^m$ **x T** and it's neighborhood to the outputspace $\mathbf{R}^n$ **x T**. The output vector **Y**(t) depends on input vector **X**(t) and the state **Z**(t) of the network. The state depends on the history of the input vectors.

Neural Networks with internal memory are different from well known networks with external memory or networks with one discrete memory cell per neuron (e.g. networks with backpropagation through time). In such networks delay is learned by the weight of a neuron per delay step. This requires a great number of neurons and only discrete time steps can be learned. Duration networks are not capable to learn a great range of delays.

The neurons in our networks contain many memory cells. If it could be possible to learn the number of delays necessary in such neuron, only one neuron for all memory cells would be required. With such networks it is possible to realize difference equations (e.g. linear filters). When we further are capable to implement an algorithm to learn continuous delays, this would result in a new quality. Than it is possible to realize differential equations.

Physical and logical structure of the Neuron (**Fig. 1** and **Fig. 2**)

The input signals $q_{0,i}(t)$ to $q_{K-1,i}(t)$ of neuron i are collected (summarized) by the input-function fe_i. The output is the signal $net_i(t)$, which will be added to the bias WB_i. This signal is the input for the transfer function ft_i. The output of the transfer function is the signal $o_i(t)$ which will be stored in the master cell of the memory. With the next clock the contents of the master cell will be shifted to the next memory cell and all so on. All memory cells without the master cell can be used as output. In this Neural Networks it is allowed to connect the net output to the net input or the output of any neuron to the input of the same neuron or any other neuron. Therefor it is not possible to calculate the network, if the master cell is also an output. It would cause duration problems in a real network. For this reason every memory-neuron has an one step delay. If one doesn't use feed back connections in parts of the network, it is possible to use neurons without memory in combination with memory-neurons. The logical structure of the neuron is shown in (Fig. 2). Every input of the neuron is an output of an other neuron or a net input. Net inputs contain memory cells too. Therefore, the delay of an input is only depending on the memory cell used for the output.

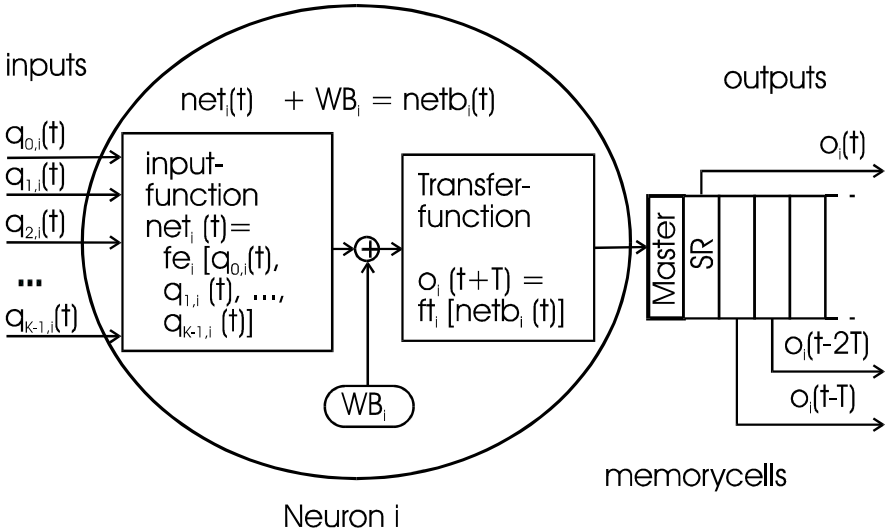

Fig. 1 pysical structure of the neuron

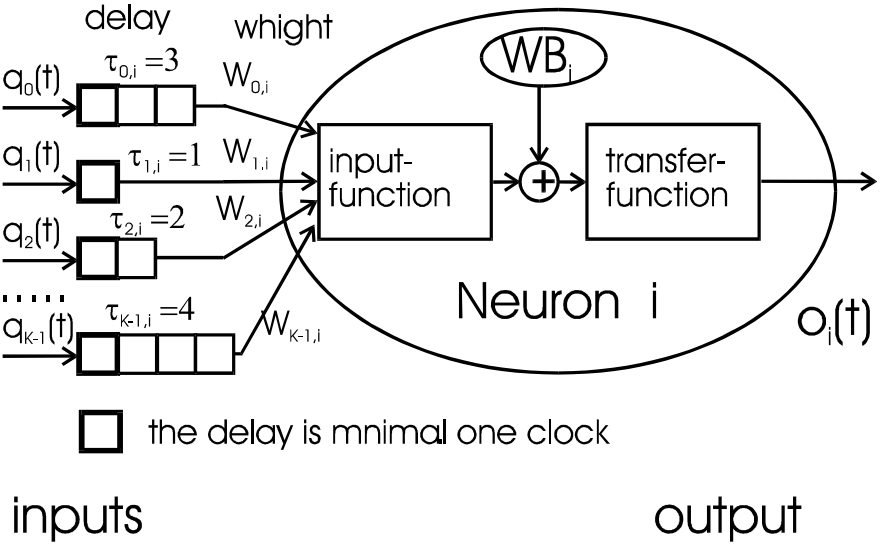

Fig. 2 logical structure of the neuron

Netstructure

In Fig. 3 the largest possible structure of a network with time delay-neurons is shown. In this case all neurons are connected with all net-inputs and all neuron-outputs are connected with an input of every neuron.

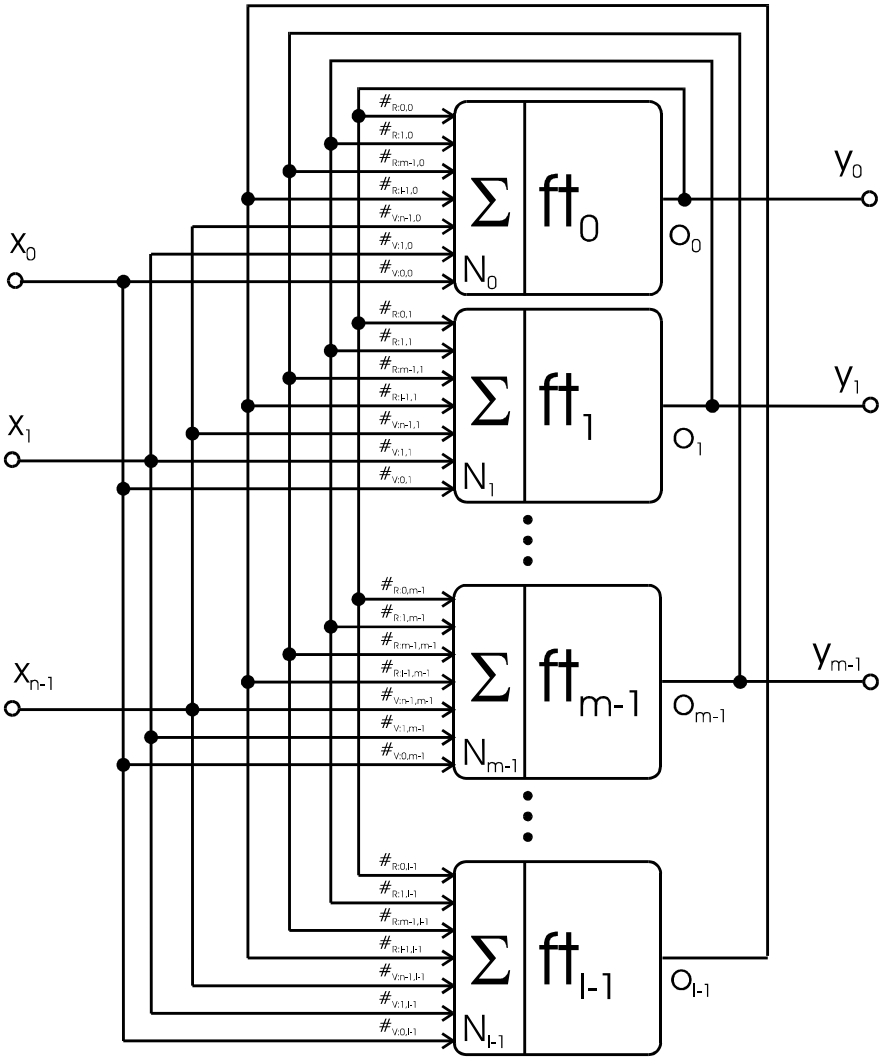

Fig. 3 largest possible netstructure

A new property of this network is the possibility to create parallel connections which are different by time delay and weight. Every connection in Fig. 3 represents 1, 2,.. ,n parallel connections. All

parallel paths of one connection are one part of the input function of the neuron. Equ. (1) and (2) describes the mathematical representation of one connection with H parallel paths (for the input function is a sum assumed) .

$$s_i(t) = \sum_{h=0}^{H_{j,i}-1} W_{j,i,h} q_j(t-\tau_{j,i,h}) \quad ; \quad S_i(p) = Q_j(p) \sum_{h=0}^{H_{j,i}-1} W_{j,i,h} e^{-p\tau_{j,i,h}} \tag{1}$$

$$g_{j,i}(t) = \sum_{h=0}^{H_{j,i}-1} W_{j,i,h} \delta(t-\tau_{j,i,h}) \quad ; \quad G_{j,i}(p) = \sum_{h=0}^{H_{j,i}-1} W_{j,i,h} e^{-p\tau_{j,i,h}} \tag{2}$$

The network shown in Fig. 3 may be depicted in a block structure as can be seen in Fig. 4.

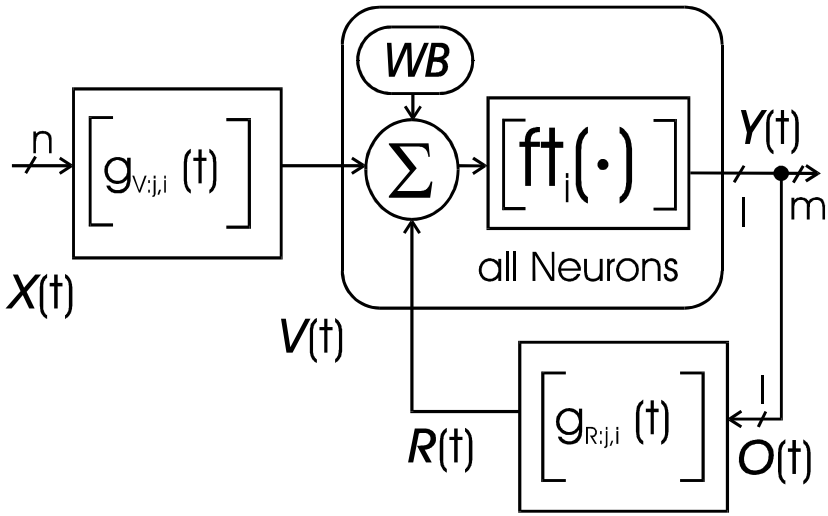

Fig. 4 Block structure of the entire network

The entire network can be described as a system of non-linear equations.

The forward vector V(t) results from convolution of input vector X(t) and forward matrix $[g_{V:j,i}(t)]$ Equ. (4).

$$\boldsymbol{V}(\mathrm{t}) = \left[\boldsymbol{g}_{\mathrm{V:j,i}}(\mathrm{t})\right] * \boldsymbol{X}(\mathrm{t}) \tag{3}$$

The feedback vector R(t) results from convolution of feedback matrix $[g_{R:j,i}(t)]$ and output vector O(t).

$$\boldsymbol{R(t)} = \left[\boldsymbol{g}_{\boldsymbol{R:i,j}}(t)\right] * \boldsymbol{O(t)} \tag{4}$$

The output vector is described in Equ. (5).

$$\boldsymbol{O}(t) = \boldsymbol{ft}(\boldsymbol{V}(t) + \boldsymbol{R}(t) + \boldsymbol{WB})$$

$$\boldsymbol{O}(t) = \boldsymbol{ft}(\left[\boldsymbol{g}_{V:j,i}(t)\right] * \boldsymbol{X}(t) + \left[\boldsymbol{g}_{R:j,i}(t)\right] * \boldsymbol{O}(t) + \boldsymbol{WB}) \tag{5}$$

Training by backpropagation

Based on that method network parameters are being changed by iteration such that the difference between the output vector and the target output vector becomes a minimum. The equation to be minimized is the squared error of these two vectors, Equ. (6).

$$E(t) = \frac{1}{2}\sum_{k=0}^{m-1}(y_k^*(t) - y_k(t))^2 = \frac{1}{2}\sum_{k=0}^{m-1} e_k^2(t) \tag{6}$$

The squared error of the network is a function of all network parameters and may be described as gradient , see Equ. (7).

$$\nabla_w E(t) = \left[\frac{\partial \mathrm{E(t)}}{\partial \mathrm{w}_{0,0,0}}, \cdots, \frac{\partial \mathrm{E(t)}}{\partial \mathrm{w}_{\mathrm{I}-1,\mathrm{I}-1,\mathrm{H}-1}}\right]$$

$$\nabla_\tau E(t) = \left[\frac{\partial \mathrm{E(t)}}{\partial \tau_{0,0,0}}, \cdots, \frac{\partial \mathrm{E(t)}}{\partial \tau_{\mathrm{I}-1,\mathrm{I}-1,\mathrm{H}-1}}\right] \tag{7}$$

$$\nabla_{WB} E(t) = \left[\frac{\partial \mathrm{E(t)}}{\partial WB_0}, \cdots, \frac{\partial \mathrm{E(t)}}{\partial WB_{I-1}}\right]$$

Parameters are changed stepwise in the direction of the negative gradient reducing the network error. In order to ensure convergence of the iteration a learning rate v was introduced to prevent that the size of steps will become to large Equ. (8).

$$\begin{aligned} w_{j,i,h} &:= w_{j,i,h} - v_w \frac{\partial \mathrm{E}(t)}{\partial \mathrm{w}_{j,i,h}} \\ \tau_{j,i,h} &:= \tau_{j,i,h} - v_\tau \frac{\partial \mathrm{E}(t)}{\partial \tau_{j,i,h}} \\ WB_i &:= WB_i - v_{WB} \frac{\partial \mathrm{E}(t)}{\partial \mathrm{WB}_i} \end{aligned} \tag{8}$$

By applying the chain rule every differential can be broken down to a product of coefficients as shown in Equ. (9). The first term at the right hand side of equation (9) represents the networks transfer function. The second term describes the transfer function of the neuron referring to w, ♦, and WB.

$$\frac{\partial \mathrm{E}(t)}{\partial \mathrm{w}_{j,i,h}} = \frac{\partial \mathrm{E}(t)}{\partial \mathrm{o}_i(t)} \frac{\partial \mathrm{o}_i(t)}{\partial \mathrm{w}_{j,i,h}}$$

$$\frac{\partial \mathrm{E}(t)}{\partial \tau_{j,i,h}} = \frac{\partial \mathrm{E}(t)}{\partial \mathrm{o}_i(t)} \frac{\partial \mathrm{o}_i(t)}{\partial \tau_{j,i,h}} \qquad (9)$$

$$\frac{\partial \mathrm{E}(t)}{\partial WB_i} = \frac{\partial \mathrm{E}(t)}{\partial \mathrm{o}_i(t)} \frac{\partial \mathrm{o}_i(t)}{\partial WB_i}$$

The internal transfer function can be seen in Fig. (5). Equ. (10), (11) and (12) describe the transfer function, output function and the differential ,respectively.

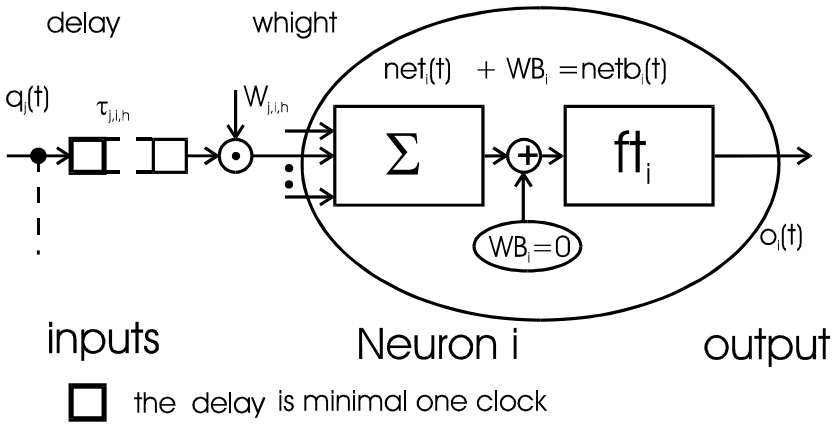

Fig. 5 Signalflow in using phase

$$o_i(t) = \mathrm{ft}_i(netb_i(t)) \qquad (10)$$

$$netb_i(t) = WB_i + \sum_{i=0}^{k_i-1} \sum_{h=0}^{H_{j,i}-1} w_{j,i,h} x_j(t - \tau_{j,i,h}) \qquad (11)$$

$$\frac{\partial o_i(t)}{\partial w_{j,i,h}} = \mathrm{ft}_i^{'}(netb_i(t))x_j(t-\tau_{j,i,h})$$

$$\frac{\partial o_i(t)}{\partial \tau_{j,i,h}} = \mathrm{ft}_i^{'}(netb_i(t))x_j^{'}(t-\tau_{j,i,h})w_{j,i,h} \qquad (12)$$

$$\frac{\partial o_i(t)}{\partial WB_{j,i,h}} = \mathrm{ft}_i^{'}(netb_i(t))$$

The network transfer function results from the structure of the connections and the interface behavior of the neurons. Therefore, all transfer paths from a given neuron i to every net output need to be considered and are described in a chain structure, see Equ. (13) and (14). Fig. 6 shows the signal flow for a single neuron.

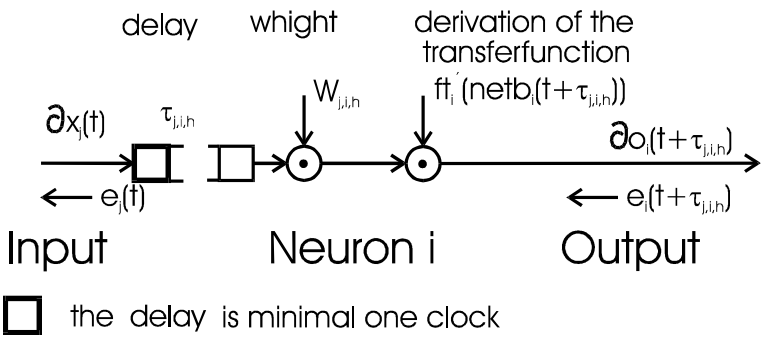

Fig. 6 Signal flow and error backpropagation in one neuron

$$\partial x_i(t-\tau_{j,i,h})w_{j,i,h}\ \mathrm{ft}_i^{'}(netb_i(t)) = \partial o_i(t) \qquad (13)$$

$$\partial x_i(t)w_{j,i,h}\ \mathrm{ft}_i^{'}(netb_i(t+\tau_{j,i,h})) = \partial o_i(t+\tau_{j,i,h}) \qquad (14)$$

Because each path from a given neurons output to a net output may have a different time delay. It is not possible to calculate the error gradient at the output of the net. Therefore, all errors need to be propagated back to the very neurons contributing to the error signal. Now, the error gradient may be calculated at these neurons. Equ (15) describes it for path h in neuron i.

$$e_i(t) = w_{j,i,h}\ \mathrm{ft}_i^{'}(netb_i(t+\tau_{j,i,h}))e_i(t+\tau_{j,i,h}) \qquad (15)$$

Fig. 7, Equ. 16, 17, 18 and 19 shows and describes the error backpropagation in the entire network.

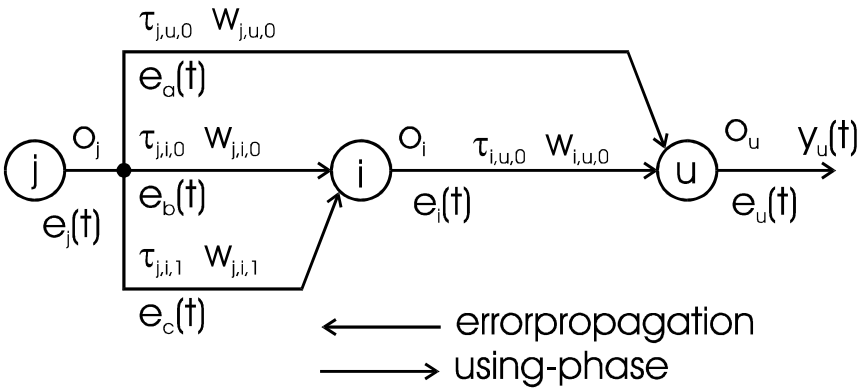

Fig. 7 Signal flow and error propagatin (e_x (t) - errors)in the network

$$e_j = e_a(t) + e_b(t) + e_c(t) \tag{16}$$

$$e_a(t) = \mathrm{ft}_u^{'}(netb_u(t+\tau_{j,u,0}))w_{j,u,0}e_u(t+\tau_{j,u,0})$$

$$e_b(t) = \mathrm{ft}_i^{'}(netb_i(t+\tau_{j,i,0}))w_{j,i,o}e_i(t+\tau_{j,i,0}) \tag{17}$$

$$e_c(t) = \mathrm{ft}_i^{'}(netb_i(t+\tau_{j,i,1}))w_{j,i,1}e_i(t+\tau_{j,i,1})$$

$$e_i(t+\tau_{j,i,0}) = \mathrm{ft}_u^{'}(netb_u(t+\tau_{j,i,0}+\tau_{i,u,0}))w_{j,u,0}e_u(t+\tau_{j,u,0}+\tau_{i,u,0})$$

$$e_i(t+\tau_{j,i,1}) = \mathrm{ft}_u^{'}(netb_u(t+\tau_{j,i,1}+\tau_{i,u,0}))w_{j,u,1}e_u(t+\tau_{j,u,1}+\tau_{i,u,0}) \tag{18}$$

As can be seen in Equ. (19), the time delay of the error is proportional to the time delay of the derivation of the neurons transfer function. Therefore, derivations must be assigned to errors in a given neuron. In the networks discussed in this paper they are stored together. When the error signal is evaluated in a neuron, then the stored pairs (derivation of transfer function and backpropageted error) are used to calculate changes of the parameters w, ♦, WB, see Fig. (8).

$$
\begin{aligned}
e_{aj}(t) \;=\; & \mathrm{ft}'_u(netb_u(t+\tau_{j,u,0})) && w_{j,u,0}\, e_u(t+\tau_{j,u,0}) \\
& + \mathrm{ft}'_i(netb_i(t+\tau_{j,i,0})) && w_{j,i,0}\ \cdot \\
& \quad \mathrm{ft}'_u(netb_u(t+\tau_{j,i,0}+\tau_{i,u,0})) && w_{i,u,0}\, e_u(t+\tau_{j,i,0}+\tau_{i,u,0}) \\
& + \mathrm{ft}'_i(netb_i(t+\tau_{j,i,1})) && w_{j,i,1}\ \cdot \\
& \quad \mathrm{ft}'_u(netb_u(t+\tau_{j,i,1}+\tau_{i,u,0})) && w_{i,u,0}\, e_i(t+\tau_{j,i,1}+\tau_{i,u,0})
\end{aligned}
\tag{19}
$$

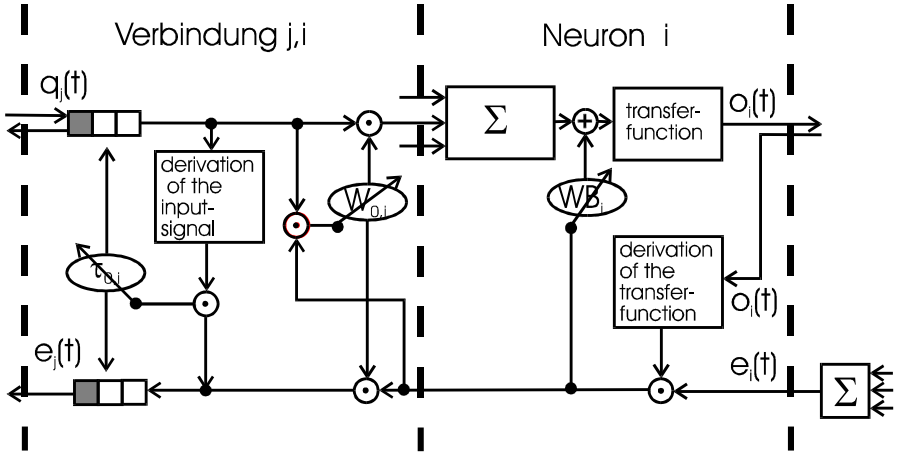

Fig. 8 Error backprogagation through a neuron

Literature

/CichocUnbeha93/ Cichocki, Andrzej; Unbehauen, Rolf: Neural Networks for Optimization and Signal Processing; B.G. Teubner Stuttgart 1993 ISBN 3_519_06444_8

/DayDavenport92/ Day, Shawn P.; Davenport, Michael R.: Continuous-Time Temporal Back-Propagation with Adaptable Time Delays; IEEE Transactions on Neural Networks August 1991 Revised: April 1992

/Rojas93/ Rojas, Raúl: Theorie der Neuronalen Netze: eine systematische Einführung; Springer 1993 ISBN 3_540_56353_9

/Zell94/ Zell, Andreas: Simulation Neuronaler Netze; Addison Wesley 1994 ISBN 3_89319_554_8