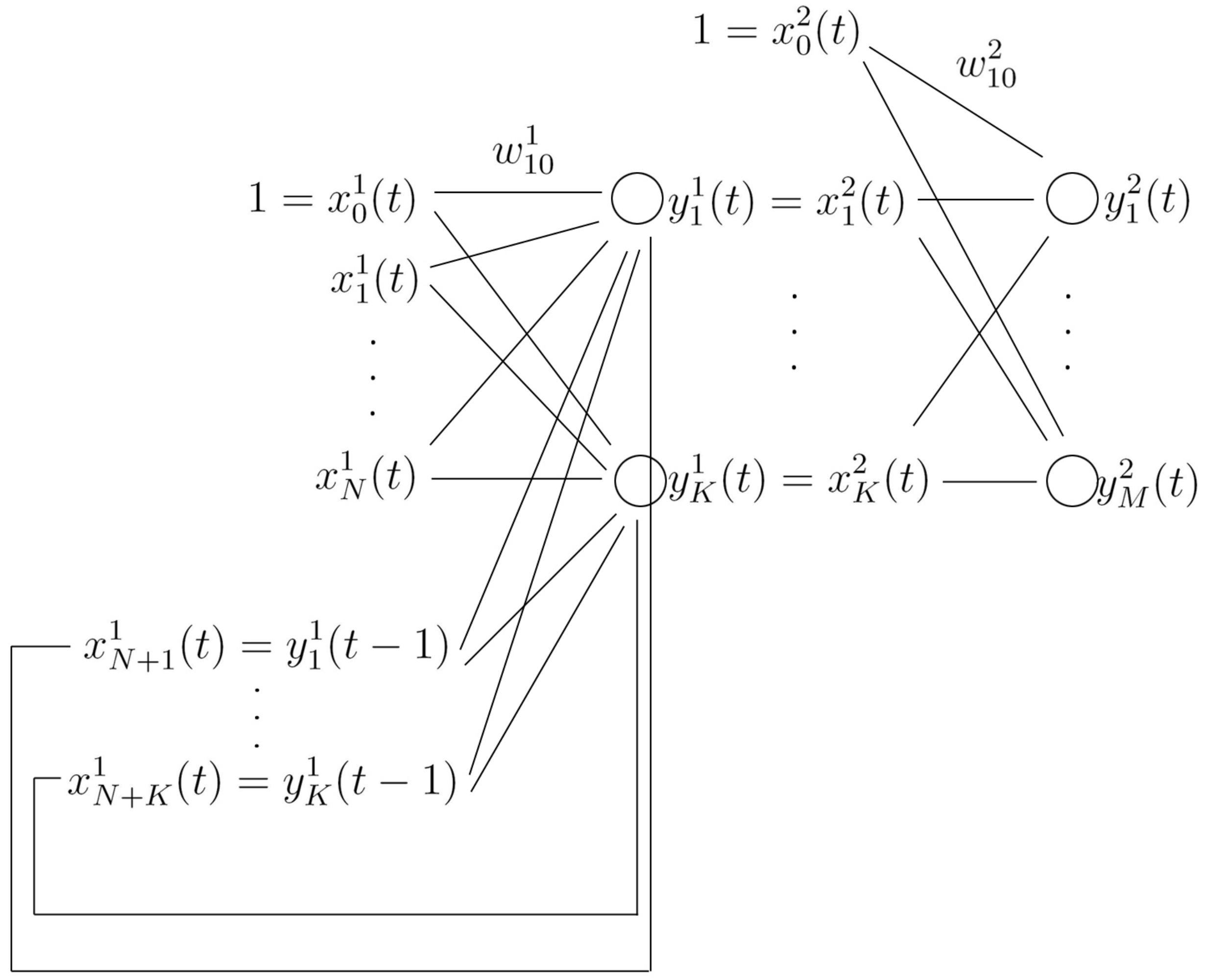
Neural Networks

Part 2

Elman neural network

- The Elman network is an example of a feedback neural network.
- Its structure consists of two layers.
- The first - hidden layer is the one whose output signals are directed as inputs to the neurons of the second layer and as a feedback signal to the input to the network.
- The feedback signals create the so-called context layer.
- The signals transferred to the context layer have no weights.

Elman neural network



Elman neural network

$$x^1 = [x_0^1(t), \ldots, x_N^1(t), x_{N+1}^1(t), \ldots, x_{N+K}^1(t)]$$

$$= [x_0^1(t), \ldots, x_N^1(t), y_1^1(t-1), \ldots, y_K^1(t-1)],$$

$$t = 0,\ [x_{N+1}^1(t), \ldots, x_{N+K}^1(t)] \equiv 0.$$

Elman neural network

$$y^1 = [y_1^1, \ldots, y_K^1],$$

$$y_i^1 = f(net_i^1), \qquad i = 1, 2, \ldots, K$$

$$net_i^1 = \sum_{j=0}^{N} x_j^1 w_{ij}^1 + \sum_{j=1}^{K} x_{N+j}^1 w_{i\ N+j}^1$$

$$i = 1, 2, \ldots, K.$$

Elman neural network

$$x^2 = y^1 = [y_0^1, y_1^1, \ldots, y_K^1],$$

$$y^2 = [y_1^2, \ldots, y_M^2],$$

$$y_i^2 = f(net_i^2) = f(\sum_{j=0}^{M} x_j^2 w_{ji}^2)$$

$$, i = 1, 2, \ldots, M.$$

Elman neural network – learning procedure

- Learning set

$$L = \{V, T\},$$

$$V = \{v^1, \ldots, v^C\}, \quad v^i = [v_1^i, \ldots, v_N^i],\ i = 1, \ldots, C$$

$$T = \{t^1, \ldots, t^C\}, \quad t^i = [t_1^i, \ldots, t_M^i],\ i = 1, \ldots, C.$$

Elman neural network – error function

$$E(w) = \sum_{i=1}^{C} \sum_{j=1}^{M} \left(t_j^i - y_j^2(v^i) \right)^2 ,$$

Elman neural network - the gradient method

$$w(t+1) = w(t) - \eta \bigtriangledown E(w(t)),$$

Elman neural network - the gradient method

$$w = [w_{10}^1, \ldots, w_{1N+K}^1, \ldots, w_{K0}^1, \ldots, w_{KN+K}^1, w_{10}^2, \ldots, w_{1K}^2].$$

$$E(w_{10}^1, \ldots, w_{1N+K}^1, \ldots, w_{K0}^1, \ldots, w_{KN+K}^1, w_{10}^2, \ldots, w_{1K}^2) = \sum_{i=1}^{C} \left(t_1^i - y_1^2(v^i)\right)^2 .$$

Elman neural network - the gradient method

$$\nabla E = \left[\frac{\partial E}{\partial w_{10}^{1}}, \ldots, \frac{\partial E}{\partial w_{1N+K}^{1}}, \ldots, \frac{\partial E}{\partial w_{K0}^{1}}, \ldots, \frac{\partial E}{\partial w_{KN+K}^{1}}, \frac{\partial E}{\partial w_{10}^{2}}, \ldots, \frac{\partial E}{\partial w_{1K}^{2}}\right].$$

$$w_{ij}^{q}(t+1) = w_{ij}^{q}(t) - \eta\left(\nabla\frac{\partial E(w(t))}{\partial w_{ij}^{q}(t)}\right),$$

$$w^{q}ij \in \left\{w_{10}^{1}, \ldots, w_{1N+K}^{1}, \ldots, w_{K0}^{1}, \ldots, w_{KN+K}^{1}, w_{10}^{2}, \ldots, w_{1K}^{2}\right\}.$$

Elman neural network - the gradient method

$$E(w) = \frac{1}{2}\left(t^c - y_1^2(v^c)\right)^2 .$$

$$\begin{aligned}
\frac{\partial E}{\partial w_{1i}^2} &= \left(\tfrac{1}{2}\left(t^c - y_1^2(v^c)\right)^2\right)|_{w_{1i}^2} \\
&= -\left(t^c - y_1^2(v^c)\right)\left(y_1^2(v^c)\right)|_{w_{1i}^2} \\
&= -\left(t^c - y_1^2(v^c)\right)\left(f(x_0^2 w_{10}^2 + \cdots + x_K^2 w_{1K}^2)\right)|_{w_{1i}^2} \\
&= -\left(t^c - y_1^2(v^c)\right) y_1^2(v^c)(1 - y_1^2(v^c))\left(x_0^2 w_{10}^2 + \cdots + x_K^2 w_{1K}^2\right)|_{w_{1i}^2} \\
&= -\left(t^c - y_1^2(v^c)\right) y_1^2(v^c)(1 - y_1^2(v^c)) x_i^2 .
\end{aligned}$$

Elman neural network - the gradient method

$$\begin{aligned}
\frac{\partial E}{\partial w^1_{bc}} &= \left(\frac{1}{2}\left(t^c - y^2_1(t)\right)^2\right)|_{w^1_{bc}} \\
&= -\left(t^c - y^2_1(t)\right)\left(y^2_1(t)\right)|_{w^1_{bc}} \\
&= -\left(t^c - y^2_1(t)\right)\left(f\left(x^2_0(t)w^2_{10} + \cdots + x^2_K(t)w^2_{1K}\right)\right)|_{w^1_{bc}} \\
&= -\left(t^c - y^2_1(t)\right)y^2_1(t)(1 - y^2_1(t))\underbrace{\left(x^2_0(t)w^2_{10} + \cdots + x^2_K(t)w^2_{1K}\right)}_{net^2_1(t)}|_{w^1_{bc}}
\end{aligned}$$

$$x^2_i(t) = y^1_i(t),$$

Elman neural network - the gradient method

$$y_i^1(t) = f\underbrace{\left(\sum_{j=0}^{N+K} x_j^1(t) w_{ij}^1\right)}_{net_i^1(t)} = f\left(\sum_{j=0}^{N} x_j^1(t) w_{ij}^1 + \sum_{j=1}^{K} y_j^1(t-1) w_{i\ N+j}^1\right),$$

$$\begin{aligned}\frac{\partial E}{\partial w_{bc}^1} &= -\left(t^c - y_1^2(t)\right) y_1^2(t)\left(1 - y_1^2(t)\right)\left(x_0^2(t) w_{10}^2 + \cdots + x_K^2(t) w_{1K}^2\right)|_{w_{bc}^1} \\ &= -\left(t^c - y_1^2(t)\right) y_1^2(t)\left(1 - y_1^2(t)\right)\left(f(net_1^1(t)) w_{11}^2 + \cdots + f(net_K^1(t)) w_{1K}^2\right)|_{w_{bc}^1}\end{aligned}$$

Elman neural network - the gradient method

$$\begin{aligned}\frac{\partial E}{\partial w^1_{bc}} &= -\left(t^c - y^2_1(t)\right) y^2_1(t)(1 - y^2_1(t)) \\ &\left(w^2_{11} f'(net^1_1(t))\, net^1_1(t)|_{w^1_{bc}} + \cdots + w^2_{1K} f'(net^1_K(t))\, net^1_K(t)|_{w^1_{bc}}\right).\end{aligned}$$

$$\begin{aligned}net^1_i(t)|_{w^1_{bc}} &= \left(\sum_{j=0}^{N} x^1_j(t) w^1_{ij}\right)\Big|_{w^1_{bc}} + w^1_{i\ N+1} f'(net^1_1(t-1)) net^1_1(t-1)\Big|_{w^1_{bc}} \\ &+ \cdots + w^1_{i\ N+K} f'(net^1_K(t-1)) net^1_K(t-1)\Big|_{w^1_{bc}}.\end{aligned}$$

Elman neural network - the Levenberg - Marquard method

The Levenberg-Marquardt algorithm is a modification of the Gauss-Newton algorithm, in which the direction of minimization d(w(t)) takes the form

$$d(w(t)) = -[\nabla^2 E(w(t-1))]^{-1} \nabla E(w(t-1))$$

$$w(t+1) = w(t) - [\nabla^2 E(w(t))]^{-1} \nabla E(w(t))$$

Elman neural network - the Levenberg - Marquard method

$$\nabla E(w(t)) = \mathcal{J}^T(w(t))\varepsilon(w(t))$$

$$\nabla^2 E(w(t)) = \mathcal{J}^T(w(t))\mathcal{J}(w(t)) + \mathcal{S}(w(t)),$$

where $\mathcal{J}$

is the Jacobian, i.e. the matrix of the first partial derivatives of the error of each sample in individual neurons of the last layer.

Elman neural network - the Levenberg - Marquard method

$$\mathcal{J}(w) = \begin{bmatrix}
\frac{\partial e_1^1}{\partial w_{10}^1} & \cdots & \frac{\partial e_1^1}{\partial w_{1\ N+K}^1} & \cdots & \frac{\partial e_1^1}{\partial w_{K0}^1} & \cdots & \frac{\partial e_1^1}{\partial w_{K\ N+K}^1} & \frac{\partial e_1^1}{\partial w_{10}^2} & \cdots & \frac{\partial e_1^1}{\partial w_{MK}^2} \\
\cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\
\frac{\partial e_M^1}{\partial w_{10}^1} & \cdots & \frac{\partial e_M^1}{\partial w_{1\ N+K}^1} & \cdots & \frac{\partial e_M^1}{\partial w_{K0}^1} & \cdots & \frac{\partial e_M^1}{\partial w_{K\ N+K}^1} & \frac{\partial e_M^1}{\partial w_{10}^2} & \cdots & \frac{\partial e_M^1}{\partial w_{MK}^2} \\
\frac{\partial e_1^2}{\partial w_{10}^1} & \cdots & \frac{\partial e_1^2}{\partial w_{1\ N+K}^1} & \cdots & \frac{\partial e_1^2}{\partial w_{K0}^1} & \cdots & \frac{\partial e_1^2}{\partial w_{K\ N+K}^1} & \frac{\partial e_1^2}{\partial w_{10}^2} & \cdots & \frac{\partial e_1^2}{\partial w_{MK}^2} \\
\cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\
\frac{\partial e_M^2}{\partial w_{10}^1} & \cdots & \frac{\partial e_M^2}{\partial w_{1\ N+K}^1} & \cdots & \frac{\partial e_M^2}{\partial w_{K0}^1} & \cdots & \frac{\partial e_M^2}{\partial w_{K\ N+K}^1} & \frac{\partial e_M^2}{\partial w_{10}^2} & \cdots & \frac{\partial e_M^2}{\partial w_{MK}^2} \\
\cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\
\frac{\partial e_1^C}{\partial w_{10}^1} & \cdots & \frac{\partial e_1^C}{\partial w_{1\ N+K}^1} & \cdots & \frac{\partial e_1^C}{\partial w_{K0}^1} & \cdots & \frac{\partial e_1^C}{\partial w_{K\ N+K}^1} & \frac{\partial e_1^C}{\partial w_{10}^2} & \cdots & \frac{\partial e_1^C}{\partial w_{MK}^2} \\
\cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\
\frac{\partial e_M^C}{\partial w_{10}^1} & \cdots & \frac{\partial e_M^C}{\partial w_{1\ N+K}^1} & \cdots & \frac{\partial e_M^C}{\partial w_{K0}^1} & \cdots & \frac{\partial e_M^C}{\partial w_{K\ N+K}^1} & \frac{\partial e_M^C}{\partial w_{10}^2} & \cdots & \frac{\partial e_M^C}{\partial w_{MK}^2}
\end{bmatrix}$$

Elman neural network - the Levenberg - Marquard method

$$e(w) = \begin{bmatrix} e_1^1 \\ \dots \\ e_M^1 \\ e_1^2 \\ \dots \\ e_M^2 \\ \dots \\ e_1^C \\ \dots \\ e_M^C \end{bmatrix}$$

Elman neural network - the Levenberg - Marquard method

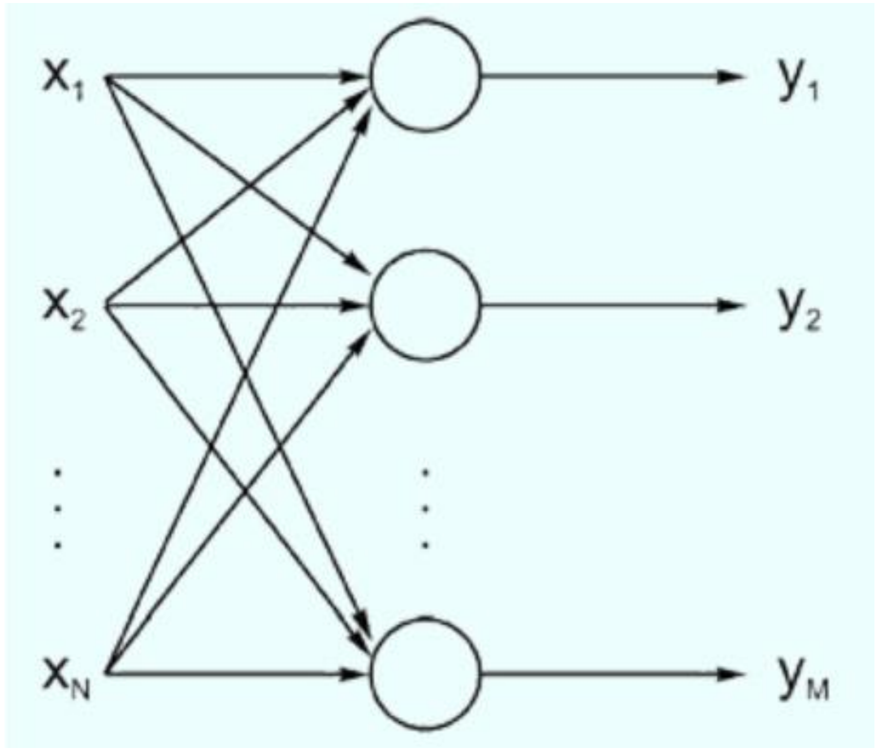
the modification of weights in the Levenberg-Marquardt method is as follows

$$w(t+1) = w(t) - [\mathcal{J}^T(w(t))\mathcal{J}(w(t)) + \mu\mathbb{I}]^{-1}\mathcal{J}^T(w(t))e(w(t))$$

where

$$d(w(t)) = -[\mathcal{J}^T(w(t))\mathcal{J}(w(t)) + \mu\mathbb{I}]^{-1}\mathcal{J}^T(w(t))e(w(t)).$$

Self-organizing neural networks



Self-organizing neural networks

- Self-learning networks are useful wherever we cannot provide information about the processed data in advance, or we are looking for this information.
- In particular, they may be useful for:
  - grouping input signals and combining them into classes,
  - detecting dependencies between found groups/classes,
  - detecting statistical properties of the input data by allocating a larger number of neurons to classify more frequently occurring data,
  - learning the topology of the input data; neurons located close to each other respond to similar signals.

Self-organizing neural networks - Algorithm

1. Determining the structure of the network, i.e. the number of neurons, as it determines the network's ability to detect a different number of groups.
2. Defining the input set.
3. Randomizing the weight values for each neuron.
4. Normalization of weights and loaded input signals according to the formula (where N is the number of inputs)

$$x_i = \frac{x_i}{\sqrt{\sum_{i=1}^{N} x_i^2}}$$

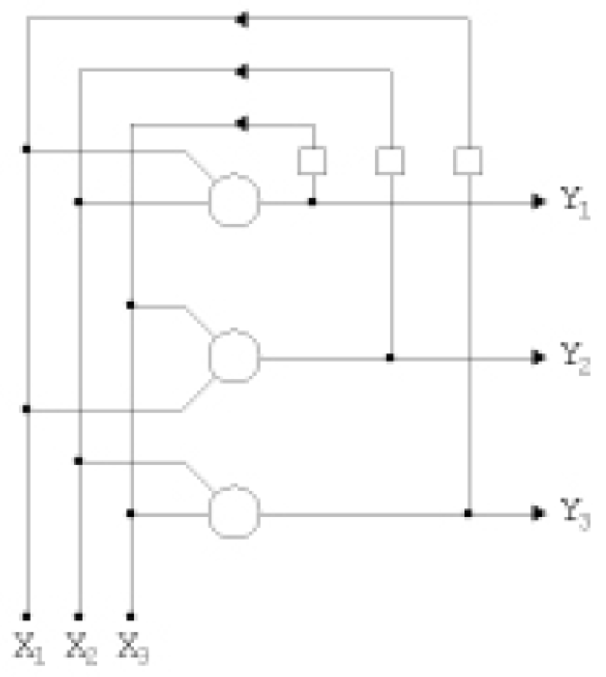
Self-organizing neural networks - Algorithm

5. Repeat the assumed number of steps:
   (a) Select the p pattern.
   (b) Calculate the output signals for the p pattern.
   (c) Find the neuron with the largest output signal. Mark it as winner.
   (d) Change the winner's weights to be closer to the p pattern according to the formula: (N is the number of inputs, is the learning coefficient. The learning rate should be reduced. It may be bigger at first, for example 0.2 and then it should tend to zero)

$$w_i = w_i + \eta[x_i - w_i]$$

6. Finally, we check to which groups the considered points belong. We assign each point the number of the neuron that responded the strongest.

Hopfield neural network



Hopfield neural network

The network proposed by Hopfield consists of N processing elements (neurons) working asynchronously. This means that the output state of only one (randomly selected) neuron is changed at a time. We assume that the output function is:

$$y(t+1) = \begin{cases} +1 & gdy \quad net > 0 \\ y(t) & gdy \quad net = 0 \\ -1 & gdy \quad net < 0 \end{cases}$$

Hopfield neural network

We assume that the weight matrix is symmetric and has zeros on the main diagonal (zeros on the main diagonal mean that the output of any neuron is not an input for itself). Because each element is connected to all the others, there are no clearly separated layers in the Hopfield network. Usually like this the network presents itself as single-layer.

Hopfield neural network

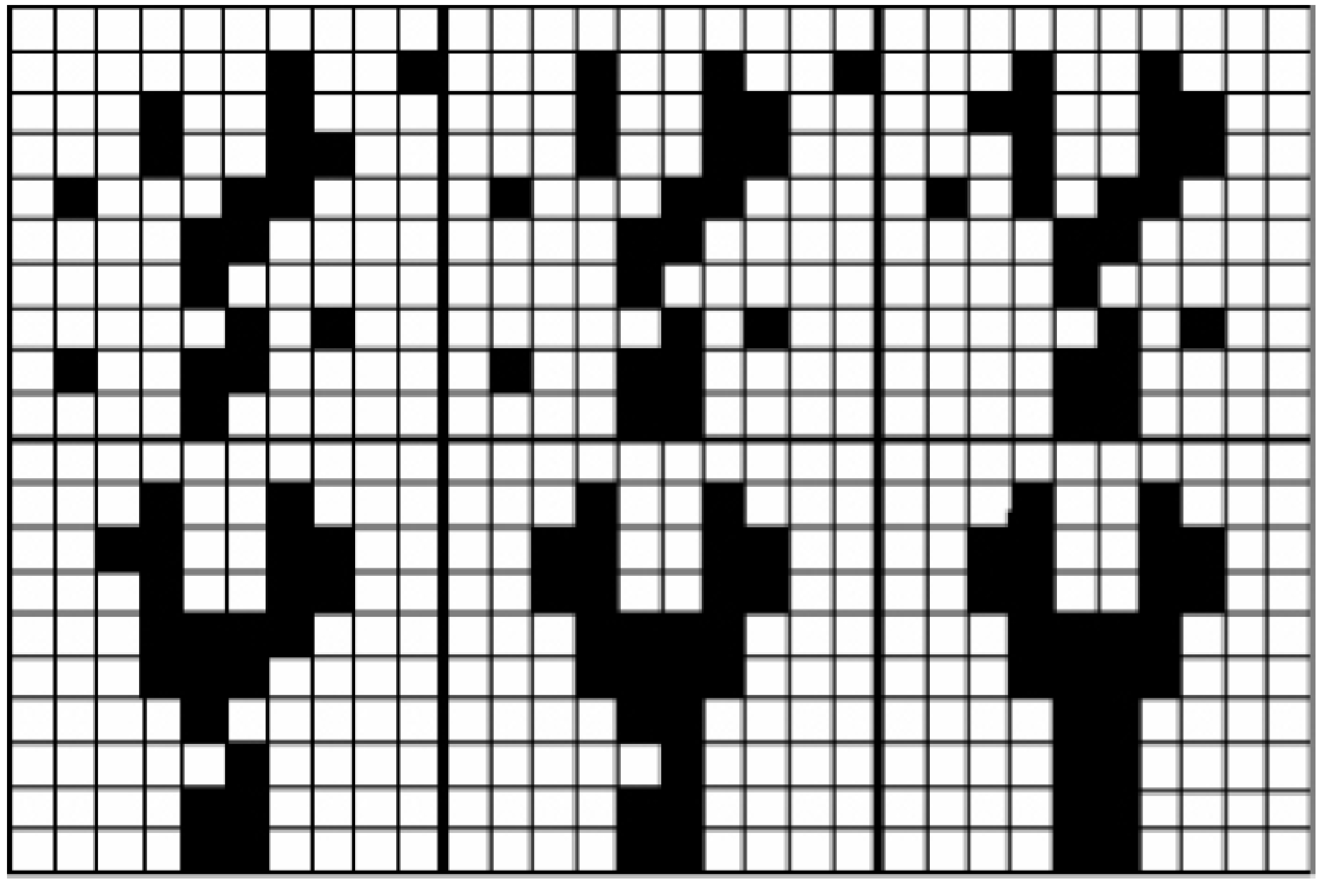
- The concept of associative memories is related to one of the basic functions of the brain - the ability to reproduce certain information based only on its outline, or more precisely, its distorted form.
- This is how we can recognize writing, often very different from the "model" letters learned in the first years of school.
- We solve crosswords where we only have a few letters match the word.
- We can recognize friends in a photo even when they differ significantly from the patterns we know.

Hopfield neural network

We will distinguish two types of this type of memory:

- **auto-associative memories** – these are memories where, based on the noisy input signal, we recreate its model, ideal form (e.g. solving a crossword);
- **heteroassociative memories** - these are memories where, based on a noisy input signal, we recreate an ideal form, but of a different signal that is associated with it.

Hopfield neural network



Hopfield neural network - Algorithm for determining weights

As in all neural networks, all knowledge in this network is hidden in the weight values. There are several algorithms that allow you to determine the weight values for the Hopfield network intended to work as an auto-associative memory.

We'll take a look here is the simplest of them.

We save pattern vectors with components −1 or +1. We determine the weights according to the formula:

$$w_{ij} = (1 - \delta_{ij}) \sum_{m=1}^{p} s_i^m s_j^m, \qquad \delta_{ij} = \begin{cases} 1 & gdy \quad i = j \\ 0 & gdy \quad i \neq j. \end{cases}$$

Hopfield neural network - Reading (recognition) algorithm

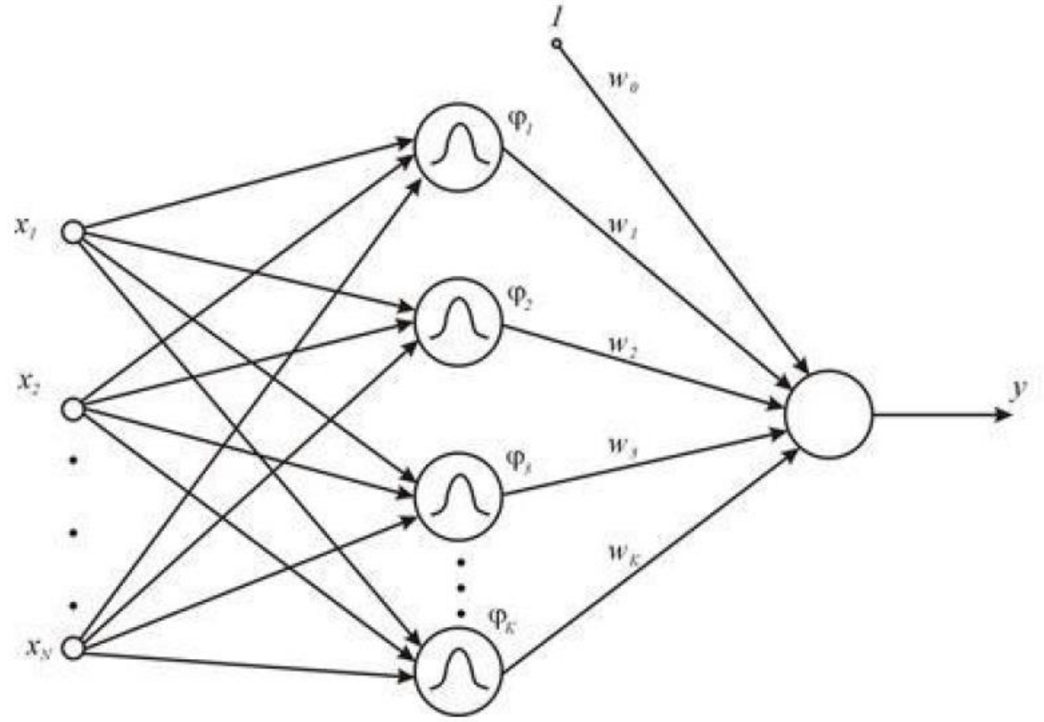
1. Determining the initial state by applying a signal to the neuron inputs; most often it is just noisy information.
2. Determining the random order in which neurons will calculate their output signals.
3. Calculation of output signals for all neurons (in the order determined in the previous point).
4. If there is no change in the output signal for any neuron, we finish the algorithm. Otherwise we go back to 2.

Hopfield neural network

Comments

While the algorithm is running, it may happen that the network recreates an image that complements the stored image (i.e. instead of −1 there will be +1 and vice versa). This is absolutely "normal" behavior.

Radial neural network



Radial neural network

- A radial neural network is a forward-directed network.
- It contains an input vector [x1, ..., xN] and two layers.
- The first one consists of K radial neurons, while the second one consists of one neuron operating as in a unidirectional network with a linear activation function.
- Weights are assigned only to connections in the second layer.
- The lack of weights in the first layer results from the way the radial neuron operates.

Radial neural network

- A radial neuron is a neuron whose activation function is the radial function.
- This function takes a vector as an argument and does not work on the principle of an adder as in one-way networks.
- Examples of radial functions are:
  - $\varphi(r) = e^{\left(-\frac{r^2}{2\delta^2}\right)}$
  - $\varphi(r) = \frac{1}{\sqrt{r^2+\delta^2}}$
  - $\varphi(r) = \sqrt{r^2 + \delta^2}$
  - $\varphi(r) = r,$

$$r = \| x - c \|, \ \delta > 0.$$