6.46.5.2.

NEURAL NETWORKS

Igor Vajda, Academy of Sciences of the Czech Republic
Jiří Grim, Academy of Sciences of the Czech Republic

Keywords
the neuron, cerebral cortex, artificial intelligence, perceptron, multilayer perceptron, learning, back-propagation algorithm, self-organizing maps, probabilistic approach

Contents

Glossary

Automaton is a mathematical machine with an input, an output and a memory. The output depends on the input and current state of memory. The memory can be modified depending on its current state and the input. automata can "learn" in the sense that they are able to modify their memory.
Neuron has different meanings in the neurophysiology and in in the neural network science. The latter is a more or less realistic abstract model of the former.
Neuron of the neurophysiology as the basic constituent of biological neural systems is a biological cell responsible for information transmission and information processing in higher organisms.
Neuron of the neural network science is an automaton with an input consisting of one or more discrete or continuous components, an output consisting of one discrete or continuous variable and a memory containing the same number of components as the input. The output is either the product of the input and memory or an appropriate function of this product.
Neural network is the basic concept of the neural network science. It is a system of interconnected neurons usually arranged into layers. A typical neural network is able to react to stimuli coming from outside by producing one or more outputs. Learning of the neural network is a process aiming at optimization of the outputs, at achieving outputs which are "intelligent" in some sense. This depends on the purpose which the neural network is to serve. Since there is a wide variety of purposes and approaches, there exists a wide variety of types of neural networks, e.g. multilayer perceptrons, radial basis functions networks, probabilistic neural networks, self-organizing Kohonen networks etc.

Summary
The method of selective coloring of neurons discovered by Golgi in 1875 enabled Ramon y Cajal to show the structure of neural tissue as a complex network of specific cells - neurons. The neurons occurring in a wide variety of shapes and sizes have a large number of inputs (synapses) at their receptive zones (dendrites) and a single output line (axon) branched into multiple synaptic endings. Microelectrode was the starting point of ideas which led McCulloch and Pitts to the binary threshold model of neuron's electrical activity. In 1949 Hebb proposed a physiological rule for synaptic plasticity as a basic adaptive principle of neural assemblies. However, despite of the extensive and detailed knowledge accumulated in the last century, the learning principles of biological neural assemblies still continue to be a strong motivation for further search of suitable neural network models. Around 1960 Rosenblatt proposed perceptron - the most popular artificial neural network and succeeded to prove the convergence of a heuristic learning rule. The optimism initiated by the perceptron convergence theorem was damped by a book of Minsky and Papert in 1969 who demonstrated fundamental limits of perceptron-like networks. A new interest in neural networks come with new approaches and learning algorithms. One of the most influential methods called self-organizing maps has been proposed by Kohonen and others to demonstrate the principles of topologically ordered maps in the brain. The widely used back-propagation algorithm was designed to solve the difficult optimization problem in multilayer networks. The present approaches are frequently used as a tool to solve practical problems without any biological analogies.

1. Introduction. Nervous Systems and Neurons

Advances of the modern neural sciences can be traced back by new research techniques. The first powerful tool invented Camillo Golgi around 1880 who proposed a selective staining of nervefibres by bichromat silver reaction. The coloring technique of Golgi was ingeniously applied by Ramon y Cajal who analysed extensively the human central nervous system. He succeeded to prove that the network of nerve fibers is not a continuous tissue but that it is actually composed of a vast number of distinct interconnected cellular units, the neurons. In 1906 Golgi and Ramon y Cajal shared the Nobel prize in medicine for their discoveries.

The detailed investigation of the structure of neural cells was enabled by the invention of electron microscope around 1940. It was found that all neurons are constructed from the same basic parts. From the bulbous cell body called soma project several root-like extensions, the dendrites, as well as a single long tubular fiber called axon. At its end axon branches into strands and substrands with button-like endings called synapses. The axon of a typical neuron makes a few thousand of synapses. The size of the soma of a typical neuron is about $10-80$ μm, while dendrites and axons have a diameter of several μm. The total length of neurons shows great variations from 0.01 mm to 1 m depending on their location.

The axon's purpose is the transmission of the generated neural activity to other neurons via synapses. The synapses are located either directly at the soma or at the dendrites of the subsequent neurons, i. e. the dendrites serve as receptors of incoming signals. At the

synapse the two neurons are separated by a tiny gap only about 0.2 μm wide. In relation to the synapse the structures are called presynaptic and postsynaptic, e. g. presynaptic neuron, postsynaptic membrane etc.

Nervous signals are of electrical nature. In the state of inactivity the interior of the neuron, the protoplasm, is negatively charged against the surrounding neural liquid. This resting potential of about -70 mV is caused by a selective permeability of the cell membrane which is impenetrable for Na^+ ions. The resulting deficiency of positive ions in the protoplasm is responsible for its negative charge.

Signals arriving from the synaptic connections may result in a transient weakening, or depolarization, of the resting potential. When the polarization is reduced to a critical level of approximately -60 mV, the membrane suddenly loses its impermeability against Na^+ ions which then enter into the protoplasm and neutralize the negative internal potential of the neuron. The quick discharge, called the spike or action potential, is followed by a gradual recovery of the membrane properties and by the corresponding regeneration of the negative resting potential. During this recovery period of several miliseconds the neuron remains incapable of further excitation. Let us remark that in 1963 Sir John Eccles, Alan Lloyd Hodgkin and Andrew Huxley were jointly awarded the Nobel prize in medicine for their detailed studies of the electrical signal transmission in the nervous system.

The discharge, which initially occurs in the cell body, then propagates along the axon to the synapses without decay since the depolarization of each new segment of the axon is always complete. This all-or-nothing rule resembling the properties of the binary electronic circuits was a strong motivation for the first simple model of a neuron as a binary threshold unit – proposed by McCulloch and Pitts in 1943. However the intensity of a nervous signal can be coded by the output frequency of spikes in the range from 1 to about 100 Hz because the interval between two electrical spikes can take any value longer than the regeneration period.

When the spike signal arrives at the synapse, special substances called neurotransmitters (e. g. acetylcholine) are released into the synaptic cleft from tiny vesicles contained in the endplate. The transmitter molecules reach the postsynaptic membrane within about 0.5 ms and modify its conductance for certain ions (Na^+, K^+, Cl^-, etc.). The arising flows of ions change the local postsynaptic potential. In case of depolarization the synapse is termed excitatory, in case of increased polarization the synapse is called inhibitory, since it counteracts excitation of the neuron. According to the so called Dale's law all the synaptic endings of an axon are either of an excitatory or an inhibitory nature. There are also significant structural differences between those two types of synapses. In 1936 Sir Henry Dale shared the Nobel prize in medicine with Otto Loewi, who discovered the chemical transmission of nerve signals at the synapse.

Just as each axon sends synapses to the dendrites and bodies of a number of downstream (efferent) neurons, so is each neuron connected to many upstream (afferent) neurons which transmit their signals to it. The soma of a neuron acts as a kind "summing" device which adds the local depolarizing (or polarizing) effects of different synapses. The depolarizing effect of a synapse decays with a characteristic time of $5-10$ ms and therefore the excitatory effects of successive spikes may accumulate at the same synapse. A high rate of repetition of spikes can therefore express a large intensity of the signal. When the total magnitude of the depolarization exceeds the threshold of about 10 mV, i. e. the potential in the cell body achieves the critical level of -60 mV, the neuron "fires" and generates an electrical spike.

In principle, a single synapse can cause a neuron to fire, though the depolarizing contribution of the synapses is generally diminishing with the increasing distance of their location from the cell body. At the other hand a synapse placed at a thin dendritic branch is more likely to trigger a spreading depolarization vave with the resulting spike. The influence of a given synapse may depend on many different aspects in a complex way but the following three factors seem to play a dominant role: the inherent strength of depolarizing effect of the synapse, its location with respect to the cell body and the repetition rate of the arriving spikes.

There is a great deal of evidence that the inherent strength of a synapse is not fixed but that it is changing in time in dependence on the activity of both the presynaptic- and postsynaptic neuron. According to a hypothesis originally postulated by Donald Hebb it is assumed that:

When an axon of cell A is near enough to excite a cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic changes take place in one or both cells such that A's efficiency as one of the cells firing B, is increased. (cf. Hebb (1949), p. 62, see also Adaptive Systems.)

In other words, an active synapse which repeatedly triggers the activation of its post-synaptic neuron will grow in strength while others will gradually weaken. The mechanisms of *synaptic plasticity* more-or-less exactly corresponding to the above Hebb's rule appear to play a dominant role in the complex process of learning.

It should be emphasized that real neurons and neural systems involve many complications which have been ignored in the above selective and purpose-oriented description. To illustrate our simplifications let us remark that

- the structural organization of human brain is very complex, only partly recognized and still less understood,
- at different locations in the brain there are usually different types of neurons with specific properties (e. g. receptor neuron, motor neuron, interneuron etc),
- there are qualitatively different types of synapses (e. g. axo-somatic, axo-dendritic, axo-axonal, dendro-dendritic) with unknown properties and unclear meaning,
- the speed of propagation of the discharge signal along the axon is increased by electrically insulating myelin sheath-segments,
- more than 20 chemically different neurotransmitters were identified in the last decades,
- the excitatory and inhibitory effects of synapses are added in the neuron in a complex way, nonlinearly in space and time with many unclear details,
- the mechanism of release of the neurotransmitter in the process of synaptical transmission is widely accepted but the discussion about some important details is not closed completely,
- there are many internal structural elements of neurons the role of which is not fully recognized,

- there are mechanisms of a short-term synaptical plasticity (so called potentiation) with unknown properties and unclear background and meaning.

In spite of the fact that neuron is not very accurate and rather unreliable functional unit the global performance of the nervous system is surprisingly perfect. The brain is robust and fault tolerant. Many neurons die daily without any significant consequence for its performance. It is flexible. It can easily adjust to a new environment by "learning". It can deal with information that is fuzzy, probabilistic, noisy or inconsistent. It is able to perform "creatively" in a way not previously trained. It is highly parallel, small, compact and dissipates very little power. The brain outperforms standard computers except only in tasks based on simple arithmetic. A good example is the processing of visual information: a one-year-old baby is much better and faster at recognizing objects or faces than even the most advanced AI system running on the fastest supercomputer (see Biological and Computational Intelligence).

A fundamental question arises about the basic functional principles of the brain which are robust and reliable enough to guarantee its outstanding properties (see also Adaptive Systems, Biological and Computational Intelligence, Cybernetics and Artificial Intelligence). Since the fifties it became increasingly obvious that the functioning of the brain cannot be explained without understanding the functional principles of the interconnected neural assemblies - the neural networks. Despite a great effort and many interesting results achieved in the last decades this problem remains a strong motivation for further research of different neural network models and learning algorithms.

2. Perceptrons and More General Models of Neurons

An important task of the nervous system of all living beings is to provide reactions to external stimuli. On the quality of these reactions depends the chance to survive in an often hostile environment. Fixed preprogrammed reactions hardly provide such a chance when the environment is subject to changes. It is therefore not surprising that the nervous system developed in the process of evolution a peculiar but characteristic property, namely the ability to learn adequate reactions (see Evolutionary Complex Systems and Adaptive Systems).

Each reaction is a result of information processing characterized by a one-way flow of information: from input sensors receiving external *stimuli* to neurons specifying the *reaction*. In typical complex systems the input information is processed by several distinct layers of neurons before it is fed to the output reactions – specifying layer. The networks in which the information does not circulate but flows in one direction are called *feed-forward neural networks*. In such networks the synaptic connections between the neurons of different layers are usually asymmetric (in the sense that they have much more inputs than outputs). The maximally asymmetric are the *unidirectional networks* where each neuron has several inputs but only one output. The architecture of these networks reduces to one or more trees where the leafs are sensoral inputs and the roots represent outputs (the reactions; the outputs may further feed into specialized subnetworks which materialize the reaction, e. g. to motor networks).

Let us start with the unidirectional networks where each neuron has only one output, and consider one such neuron with $K > 1$ inputs. The input values $x = (x_1, \ldots, x_K)$

may be binary numbers (e. g. the neuron detects whether an input amplitude exceeds zero or not, for simplicity we put $x_k = 1$ or $x_k = -1$ for the k-th signal, $1 \le k \le K$) or real numbers (e. g. the neuron detects the amplitudes x_k themselves, $1 \le k \le K$). The output of the neuron is a function $F(x, w)$ of the input vector x and synaptic weights $w = (w_1, \ldots, w_K)$ associating the inputs $1, \ldots, K$ with the output. These weights represent a memory of the neuron and they can be modified by using a feedback information about the gains or losses caused by the outputs $F(x, w)$ influenced by these weights. This modification is called *learning* – the main object of study in neural networks.

The neuron output is usually of the form

$$F(x, w) = f\left(\Sigma_k w_k\, x_k\right), \tag{1}$$

where $f(h)$ is a function of the *synaptic potential*

$$h = \Sigma_k w_k\, x_k$$

caused by the action of the input x (stimuli if the neuron is from the input layer, or outputs of the K connected neurons of the previous layer, respectively). This potential is a linear function of the inputs while the *transfer function* $f(h)$ is usually a nonlinear function of the potential. The standard choice is

$$f(h) = \tanh(\beta h) = \frac{1 - \exp\{-2\beta h\}}{1 + \exp\{-2\beta h\}} \tag{2}$$

for some $\beta > 0$ (so-called *sigmoidal function*; it is skew-symmetric about $h = 0$, increasing from -1 at $h = -\infty$ to 1 at $h = \infty$, with $f(0) = 0$ and the derivative $f'(0) = \beta$), or the limit of this function for $\beta \to \infty$,

$$f(h) = \mathrm{sgn}(h) = \left\langle \begin{array}{ll} 1 & \text{if} \quad h \ge 0 \\ -1 & \text{if} \quad h < 0. \end{array} \right. \tag{3}$$

The continuous transfer function (2) is used when the inputs are continuous and the jump function (3) is used when the inputs are binary. Note that the *binary neuron* is preferred in the neurophysiology since in this case the inputs as well as output (3) possess the features of the model of Mc Culloch and Pitts mentioned in Part 1. For various computer information processing applications the *continuous neuron* with the differentiable output (2) is more convenient.

The most distinguished feature of neurons is their ability to learn, i. e. to adapt their memory (synaptic weights) to the state of environment. Suppose that

$$x^t = (x_1^t, \ldots, x_k^t)$$

are inputs of the neuron at time instants $t = 1, 2, \ldots, T$, and that there exist a function $\varphi(x)$ specifying the best possible reactions of the neuron to arbitrary inputs x (*correct outputs*). To optimize weights w at a fixed time t means to minimize the squared deviation

$$D(w) = \frac{1}{2}[f(h(w)) - \varphi]^2,$$

where $f(h)$ is given by (2) and

$$h(w) \equiv h^t(w) = \Sigma_k w_k\, x_k^t, \qquad \varphi \equiv \varphi^t = \varphi(x^t).$$

After standard application of the gradient method and using the derivative

$$\psi^t(w) = 1 - f^2(h^t(w))$$

of the function (2) at $h = h(w)$ one obtains that

$$w^{t+1} = w^t - \varepsilon\beta[f(h^t(w^t)) - \varphi^t]\,\psi^t(w^t)\,x^t. \tag{4}$$

This is an improvement of weights w^t provided ε is small enough (otherwise one can overshoot the global or local minimum of $D(w)$).

If $D(w)$ is replaced by the sum of squared deviations

$$D_t(w) = \frac{1}{2}\Sigma_{s\le t}[f(h^s(w)) - \varphi^s]^2$$

then the same argument as used above leads to the conclusion that

$$w^+ = w - \varepsilon\beta\Sigma_{s\le t}[f(h^s(w)) - \varphi^s]\,\psi^s(w)\,x^s \tag{5}$$

satisfies for $0 < \varepsilon \ll 1$ the inequality $D_t(w^+) < D_t(w)$. This justifies the recursive formula

$$w^{t+1} = w^t + \varepsilon_t\Sigma_{s\le t}[f(h^s(w^t)) - \varphi^s]\,\psi^s(w^t)\,x^s \tag{6}$$

for improvement of weights at time t where ε_t decreases to 0 as $t \to \infty$. This formula is known in the literature as the *perceptron learning rule*.

If the inputs $x^1, x^2, \ldots$ are random, distributed by a probability law P and mutually independent (or ergodic), then for large t and arbitrary w

$$\frac{2}{t}\,D_t(w) \doteq E\left[f\left(\Sigma_k w_k\,x_k\right) - \varphi(x)\right]^2 \triangleq e(w) \quad \text{for } x \sim P$$

is the neuron *mean square error* (MSE). Since f, φ and ψ are bounded functions, the perceptron learning rule selects weights from a bounded set when the support of the law P is bounded and $\varepsilon_t = O(1/t)$ as $t \to \infty$. This together with the fact that $e(w^{t+1}) < e(w^t)$ implies that w^t (or a subsequence of w^t) has a limit w^∞ as $t \to \infty$ which minimizes (locally or globally) the MSE, i. e.

$$e(w^\infty) = \inf_w e(w).$$

The learning rule (6) is also known under the name *synaptic plasticity law*. It enables the neuron to learn optimal reactions $f\left(\Sigma_k w_k^\infty\,x_k\right)$ in an arbitrary environment (represented in the present context by the probability law P). If the environment (i. e. the law P) changes and then remains for a relatively long period stable then the learning rule (6) produces weights w^t tending to a new limit w^∞ which is optimal in the new environment.

The error reduction property of the learning rules (5) and (6) is justified only for neurons with differentiable transfer functions $f(h)$. It is thus not justified for the binary neurons where $f(h)$ is given by (3). The nondifferentiable transfer function (3) is a limit of the differentiable function (2) for $\beta \to \infty$. If $\psi^s(w) \doteq c \neq 0$ for a fixed w and large β (which is satisfied by w from the neighborhood of $w^0 = (0, 0, \ldots, 0)$ and $c = 2$) then one can argue from (5) with $\varepsilon = 1/(\beta c)$ that $D_t(w^+) < D_t(w)$ for

$$w^+ = w - \Sigma_{s\le t}[f(h^s(w^s)) - \varphi^s]\,x^s.$$

Replace in this formula $f(h^s)$ of (2) by $f(h^s)$ of (3), which can be justified by the assumption that β is large. If $w \neq w^0$ then we replace w by its close neighbor w^0. As a result we obtain that the sequence $w^+ = w^t$ satisfies the recursive formula

$$\begin{aligned} w^{t+1} &= w^t - [f(h^t(w^t)) - \varphi^t]\, x^t \\ &= \begin{cases} w^t & \text{if} \quad f\left(\Sigma_k w_k^t\, x_k^t\right) = \varphi(x^t) \\ w^t + \text{sign}(\varphi(x^t))\, x^t & \text{if} \quad f\left(\Sigma_k w_k^t\, x_k^t\right) \neq \varphi(x^t). \end{cases} \end{aligned} \tag{7}$$

This formula is the well known *binary perceptron learning rule*.

It is seen from the derivation presented above that now the sequence w^t can pretend neither the general asymptotic optimality nor the convergence observed in the continuous perceptrons. The only general pretension is that $e(w^t) < e(w)$ for $t > 1$ and w from the close neighborhood of $w^0 = (0, 0, \ldots, 0)$. Next we show that, nevertheless, in an important class of situations the learning rule (7) is a synaptic plasticity law guaranteeing both the asymptotic optimality and the convergence. But before going into these special details, let us mention the following general terminological remarks.

Remarks. (i) The system with inputs x, synaptic memory w, output $f\left(\Sigma_k w_k\, x_k\right)$ and learning rules (6) or (7), called neuron, is of course not a cell of nervous system of a living organism studied by Golgi and Ramon y Cajal, also called neuron. The term *neuron* in the neural networks literature usually means a model of a class of the real world neurons. The particular model mentioned at the beginning of this remark is known under the name *perceptron*, following Rosenblatt who introduced both, the model and the name. Perceptron (and, more generally, any model of neuron, see the definition of general neural network in Part 3 below) is a particular case of an abstract mathematical machine called *automaton* which receives inputs x^t from an environment, produces outputs $y_t = F_t(x^t, w^t)$ depending on time t, inputs x^t and a memory w^t, and recursively innovates the memory by a rule $w^{t+1} = G_t(x^t, w^t)$. Functions $F(x^t, w^t) = f(h^t)$ figuring in majority of all models of neurons are given in (1) and the special perceptron examples of functions $G_t(x^t, w^t)$ or $G(x^t, w^t)$ are given in (6) and (7).

(ii) Let us point out that if $x = (x_1, \ldots, x_K)$ is extended by a dummy component $x_{K+1} = 1$ and $w = (w_1, \ldots, w_k)$ is extended by a new variable w_{K+1} then $h = \Sigma_k w_k\, x_k$ figuring in (1) can be written as

$$h = \sum_{k=1}^{K} w_k\, x_k - \theta \quad \text{where} \quad \theta = -w_{K+1}$$

so that the perceptron output is of the form

$$f\left(\Sigma_k w_k\, x_k\right) = f\left(\sum_{k=1}^{K} w_k\, x_k - \theta\right). \tag{8}$$

Here the right-hand sum is the synaptic potential and θ is a threshold for this potential. In this extended form the perceptron is more flexible and better formalizes the McCulloch and Pitt's model mentioned in Part 1.

(iii) Finally, let us remark that the learning rules (6) and (7) are nothing but formalized variants of the *Hebb's rule* formulated in Part 1: The weights w^{t+1} "grow" in positive or

negative direction depending on the reaction achieved by w^t and the intensity of the "growth process" depends on the inadequacy of this reaction. E. g., in (7) there is no "growth" (i. e. $w^{t+1} = w^t$) if the reaction produced by w^t is correct.

Now we are prepared to analyze the asymptotic optimality of binary perceptrons. These properties were systematically studied in the early 60'ies by Rosenblatt, Novikoff, Nillson and some other authors. An important result of this type is due to Rosenblatt. He proved for the perceptrons *extended in the sense of* (8) that if

$$\varphi(x) = f(H(x)) = \mathrm{sgn}(H(x)) \quad \text{(cf. (3))}, \tag{9}$$

where $H(x_1, x_2, \ldots, x_{K+1})$ is linear in all its variables, i. e. $H(x) = 0$ is a hyperplane in R^k then, after finitely many steps, the learning rule (7) enables correctly represent input vectors from any finite set $\{x^1, x^2, \ldots, x^T\}$ (if more than T learning steps are needed then it is assumed that $x^{T+1} = x^1,\ x^{T+2} = x^2$ etc). More precisely, for arbitrary initial weights w^1 there exists $1 \leq t_1 < \infty$ such that after t_1 learning steps the perceptron achieves weights $w = w^{t_1}$ with the property

$$f\left(\Sigma_k w_k\, x_k\right) = \varphi(x) \quad \text{for all} \quad x \in \{x^1, \ldots, x^T\}.$$

This means that either $w^{t_1} = w$, i. e. the unknown weights w are correctly established by the learning, or w^t differs from w but classifies the vectors $x^t,\ 1 \leq t \leq T$, in precisely the same way as does w.

This result is often called simply a *perceptron convergence theorem*. Let us emphasize that it guarantees an asymptotic optimality (the errorless outputs) for all binary perceptrons in the environments where the correct reactions function φ is pseudolinear in the sense specified in (9).

In the next examples we illustrate the perceptron convergence theorem by letting simple perceptrons with binary input vectors $x = (x_1, x_2)$ to learn arbitrary Boolean functions $\varphi(x_1, x_2)$. Such functions characterize the truth values of composite propositions of the type $X = X_1\, \mathrm{R}\, X_2$ where R is a logical relation between two propositions, provided x_1, x_2 are the truth values of X_1, X_2. Well known examples such relations are OR, AND and IMPLIES. For our purposes the "true" will be represented by the value 1 and "false" by -1.

Example 1: Proposition $X_1\, \mathrm{OR}\, X_2$**.** The four possible values $x^t = (x_1^t, x_2^t)$, $1 \leq t \leq 4$, of $x = (x_1, x_2)$ are presented in the first row of Table 1 and the corresponding values of $\varphi(x^t)$ in the last row.

t	1	2	3	4	5	6
x^t	-1,-1	-1,1	1,-1	1, 1	-1,-1	-1,1
w^t	0,0	1,1	1,1	1,1	1,1	1,1
h^t	0	0	0	2	-2	0
f^t	1	1	1	1	-1	1
$\varphi(x^t)$	-1	1	1	1	1	1

Table 1: Detailed record of the dynamics of the perceptron of Example 1 with the initial weights $w^1 = (0, 0)$ given in the $t = 1$ column. We see that the weights w^t remain constant, equal (1,1), for $2 \leq t \leq 5$, and that the columns for $t \geq 6$ coincide with those for $t - 4$. Therefore this perceptron learns the correct representation of the Boolean function of X_1 OR X_2 at the time $t = 2$, after just one learning step.

We define $x^5 = x^1$, $x^6 = x^2$ etc. In order to provide a possibility to study the dynamics of the perceptron under the learning rule (7), we present also the weights $w^t = (w_1^t, w_2^t)$ and the values of

$$h^t = \Sigma_1^2 w_k^t \, x^t, \quad f^t = f(h^t) = \text{sign}(h^t) \quad \text{and } \varphi(x^t).$$

The function $\varphi(x_1, x_2) = x_1 \vee x_2$ obviously satisfies the condition (9). The results are commented in the text to the table.

Similar result as in Example 1 one obtains also for the proposition X_1 AND X_2. The only difference is that here $\varphi(x_1, x_2) = x_1 \wedge x_2$ does not satisfy (9) when the hyperplane $H(x) = 0$ is passing through the origin. Thus we need to introduce the dummy variable $x_3^t \equiv 1$ (add the constant coordinate 1 in the x_t-row of Table 1 and one more 0 in the first position of the w_t-row, and then calculate in accordance with (7) the weights $w^t = (w_1^t, w_2^t, w_3^t)$ in the remaining positions of this row). In this manner one obtains that the Boolean function of this proposition is again correctly represented at $t \geq 2$, after one learning step.

Example 2: Proposition X_1 IMPLIES X_2**.** Here we use the same inputs x_1^t, x_2^t as above, including the dummy component $x_3^t \equiv 1$. Consequently we use the extended 3-D weights $w^t = (w_1^t, w_2^t, w_3^t)$, which means that the summation for h^t in Example 1 extends from 1 to 3. The results are in Table .

t	1	2	3	4	5	6	7
x^t	-1,-1,-1	-1,1,1	1,-1,1	1,1,1	-1,-1,1	-1,1,1	1,-1,1
w^t	0,0,0	0,0,0	0,0,0	-1,1,-1	0,2,0	-1,1,1	-1,1,1
h^t	0	0	0	-1	-2	3	-1
f^t	1	1	1	-1	-1	1	-1
$\varphi(x^t)$	1	1	1	1	1	1	-1

Table 2: Detailed record of the dynamics of the perceptron of Example 2 with the initial weights $w^1 = (0, 0, 0)$ visible in the $t = 1$ column. After calculating the columns for $t \geq 8$, one obtains constant weights $w^t = (-1, 1, 1)$ for $6 \leq t \leq 9$, and also columns which for $t \geq 10$ coincide with those for $t - 4$. Therefore this perceptron correctly represents the Boolean function of X_1 IMPLIES X_2 in all time instants $t \geq 6$, after 5 learning steps.

Example 3: Proposition X_1 XOR X_2. The relation XOR means EXCLUSIVE–OR in the sense that $\varphi(x_1, x_2) = 1$ if and only if $x_1 \vee x_2 = 1$ and $x_1 \wedge x_2 = -1$. Thus now a table analogous to Table differs from Tables 1 and 2 in the bottom line, which now reads:

t	1	2	3	4	5	6	7
$\varphi(x^t)$	-1	1	1	-1	-1	1	1

After evaluating such a new table with $w^1 = (0, 0, 0)$ one obtains that all w^t, $1 \leq t \leq 4$, are mutually different and the columns for $t \geq 5$ coincide with those for $t - 4$. Thus the weights w^t vary periodically for $t \geq 1$ with the period $T = 4$. This corresponds very well to the fact that the condition (9) of the perceptron convergence theorem is not satisfied by the present $\varphi(x_1, x_2)$: the points $x = (1, -1)$ and $x = (-1, 1)$ with $\varphi(x) = 1$ cannot be separated in R^2 by a hyperplane from the remaining two points $x = (1, 1)$ and $x = (-1, -1)$ for which $\varphi(x) = -1$.

3. Multilayered Perceptrons and General Neural Networks

In Example 3 we argued that the XOR problem cannot be solved by simple perceptrons since the input stimuli x^t with the property $\varphi(x^t) = 1$ are not linearly separable from those with $\varphi(x^t) = -1$. In the next example we show that this problem can be solved by a unidirectional network consisting of three perceptrons.

Example 4: Proposition X_1 XOR X_2 revisited. Consider $x^t = (x_1^t, x_2^t, 1)$ where x_1^t, x_2^t are the same truth values of X_1, X_2 as in Examples 1–3. Let the input stimuli x^t feed at discrete time instants $t = 1, 2, \ldots$ simultaneously two parallel *input perceptrons* indexed by $j = 1, 2$, with synaptic weights

$$\overline{w}_j^t = (\overline{w}_{jk}^t : 1 \leq k \leq 3).$$

It holds

$$\overline{h}_j^t \triangleq \Sigma_1^3 \overline{w}_{jk}^t \, x_k^t = \Sigma_1^2 \overline{w}_{jk}^t \, x_k^t - \overline{\theta}_j^t \quad \text{cf. (8)}, \tag{10}$$

where $\overline{\theta}_j^t = -\overline{w}_{j3}^t$ are thresholds for the synaptic potentials $\overline{h}_j^t$ leading to the outputs

$$\overline{x}_j^t = f(\overline{h}_j^t) = \operatorname{sgn}\left(\Sigma_1^3 \overline{w}_{jk} \, x_k^t\right), \quad j = 1, 2. \tag{11}$$

Outputs $\overline{x}^t = (\overline{x}_1^t, \overline{x}_2^t, 1)$ of the input perceptrons (extended by the dummy $x_3^t \equiv 1$) are assumed to feed inputs of an *output perceptron* with synaptic weights $w^t = (w_k^t : 1 \leq k \leq 3)$ and outputs

$$f^t = f(h^t) = \operatorname{sgn}\left(\Sigma_1^3 w_k^t \, \overline{x}_k^t\right). \tag{12}$$

Thus we have a feed-forward network of three perceptrons organized in two layers: two perceptrons in the input layer and one in the output layer. This network is fed from outside by a sequence of stimuli x^t describing the truth values of propositions X_1, X_2 and responds to these stimuli by the outputs f^t defined by (12). These responses are correct if $f^t = \varphi(x^t)$ and incorrect if $f^t \neq \varphi(x^t)$.

The state of the network is completely specified any time t by the quadruplet $(x^t, \overline{w}_1^t, \overline{w}_2^t, w^t)$. To achieve this end we need to specify the memories $(\overline{w}_1^t, \overline{w}_2^t, w^t)$. This can be done by considering arbitrary initial

$$(\overline{w}_1^1, \overline{w}_2^1, w^1)$$

and subsequently applying the recursive binary perceptron learning rule (7).

To be precise, assume that $\overline{w}_j^{t+1}$ is defined by (7) for

$$w^t = \overline{w}_j^t \quad \text{and} \quad \varphi(x^t) = \overline{\varphi}_j(x^t),$$

where $\overline{\varphi}_j$ is defined by means of the outputs (11) and function φ as follows

$$\overline{\varphi}_j(x^t) \begin{cases} = \overline{x}_j^t & \text{if} \quad f^t = \varphi(x^t) \\ \neq \overline{x}_j^t & \text{if} \quad f^t \neq \varphi(x^t). \end{cases}$$

We see that the outputs of both the input perceptrons are considered to be incorrect (i. e. $\overline{\varphi}_j(x^t) \neq \overline{x}_j^t$ for $j = 1, 2$) if the whole network output f^t is incorrect. This is a simple example of learning by an *error back-propagation*. Let us notice that the rule $\overline{\varphi}_j(x^t) \neq \overline{x}_j^t$ uniquely determines $\overline{\varphi}_j(x^t)$ in the sense that $\overline{\varphi}_j(x^t) = 1$ if $\overline{x}_j^t = -1$ and $\overline{\varphi}_j(x^t) = -1$ if $\overline{x}_j^t = 1$.

As to the output perceptron, it is assumed to be learned by (7) for the truth value function φ under consideration and $x^t = \overline{x}^t$. The dynamics of the network under consideration for one possible specification of initial weights $\overline{w}_1^t$, $\overline{w}_2^t$ and w^t is in Table 3.

t	1	2	3	4	5	6
x^t	-1,-1,1	-1,1,1	1,-1,1	1,1,1	-1,-1,1	-1,1,1
$\overline{w}_1^t$	2,0,0	2,0,0	1,1,1	1,1,1	1,1,1	1,1,1
$\overline{h}_1^t$	-2	-2	1	3	-1	1
$\overline{x}_1^t$	-1	-1	1	1	-1	1
$\overline{\varphi}_1^t$	-1	1	1	1	-1	-1
$\overline{w}_2^t$	0,2,0	0,2,0	1,1,-1	1,1,-1	1,1,-1	1,1,-1
$\overline{h}_2^t$	-1	2	-1	1	-3	-1
$\overline{x}_2^t$	-1	1	-1	1	-1	-1
$\overline{\varphi}_2^t$	-1	-1	-1	1	-1	-1
$\overline{x}^t$	-1,-1,1	-1,1,1	1,-1,1	1,1,1	-1,-1,1	1,-1,1
w^t	-3,-2,-3	3,-2,-3	2,-1,-2	2,-1,-2	2,-1,-2	2,-1,-2
h^t	-4	-8	1	-1	-1	1
f^t	-1	-1	1	-1	-1	1
$\varphi(x^t)$	-1	1	1	-1	-1	1

Table 3: Detailed record of the dynamics of the network of 3 perceptrons of Example 4, with the error back-propagation learning, in time $t = 1, 2, \ldots$, for the initial weights $\overline{w}_1^1$, $\overline{w}_2^1$, w^1 given in the first column. The table is divided horizontally into three blocks which separately describe the dynamics of individual perceptrons in the same way as used in Tables 1 and 2. The row x^t contains common inputs for the first two perceptrons, and the row $\overline{x}^t$ the inputs for the last perceptron. This network is learned to solve the XOR problem at $t = 3$, after two learning steps.

We see from Table that $f^6 = \varphi^6$, so that $\overline{w}_i^7 = \overline{w}^6$ and $w^7 = w^6$. But $(\overline{w}_i^6, w^6) = (\overline{w}_i^3, w^3)$ and also $x^7 = x^3$. Therefore the quadruplets $(x^t, \overline{w}_1^t, \overline{w}_2^t, w^t)$ with $t = 7$ and $t = 3$ coincide. This coincidence can be extended by induction to the quadruplets with any t and $t + 4$. Since the weights $\overline{w}_1^t$, $\overline{w}_2^t$, w^t are constant for all $3 \leq t \leq 6$, this means that they remain constant for all $t \geq 3$. This means that the network was learned to solve the XOR problem correctly after two learning steps. Note, however, that the learning under consideration does not guarantee for arbitrary initial weights to find such a solution.

The network of Example 4 with perceptrons arranged in two layers easily generalizes to networks with arbitrary (binary or continuous) perceptrons arranged in three layers. The first layer contains $J \geq 2$ perceptrons with common inputs $x = (x_1, \ldots, x_k)$ and outputs

$$\overline{x}_j = f(\overline{h}_j), \quad \overline{h}_j = \Sigma_k \overline{w}_{jk}\, x_k - \overline{\theta}_j, \quad j = 1, \ldots, J.$$

Vector $\overline{x} = (\overline{x}_1, \ldots, \overline{x}_J)$ of the outputs of these perceptrons is a common input of $I \geq 2$

perceptrons situated in the second layer. The outputs of the second layer perceptrons are

$$\overline{\overline{x}}_i = f(\overline{\overline{h}}_j), \quad \overline{\overline{h}}_j = \Sigma_j \overline{\overline{w}}_{ij}\, \overline{x}_j - \overline{\overline{\theta}}_j.$$

Finally, vector $\overline{\overline{x}} = (\overline{\overline{x}}_1, \ldots, \overline{\overline{x}}_I)$ is a common input of $L \geq 1$ perceptrons of the third layer where the outputs are

$$f(h_\ell), \quad h_\ell = \Sigma_i w_{\ell i}\, \overline{\overline{x}}_i - \theta_\ell, \quad \ell = 1, \ldots, L. \tag{13}$$

This network is an example of *3-layered perceptron* where the first layer is called an *input layer*, the second layer is a *hidden layer* and the third layer is an *output layer*.

By inserting between the input and output layer more than one hidden layer we obtain a neural network called *multilayered perceptron*. If in this network the neurons of the perceptron type, with single outputs $f\left(\sum_\gamma w_{\beta\gamma}\, x_\gamma - \theta_\beta\right)$, are replaced by neurons with multiple outputs $f_\alpha\left(\sum_\gamma w_{\alpha\beta\gamma} - \theta_{\alpha\beta}\right)$, then we speak about *cybernetic networks*.

A *general neural network* can be defined rigorously as a collection of automata indexed by $1 \leq i < \infty$ which are partially (or completely) connected by oriented links, where: (i) a real-valued weight w_{ik} is associated with each link $i \leftarrow k$, (ii) a state variable x_i and a real-valued bias θ_i are associated with the i-th automaton, and (iii) the output to all outgoing links $i \rightarrow j$ is a real-valued function $f_i(x_k, w_{ik}, \theta_i : k \neq i)$, where $k \neq i$ runs over all incoming links $i \leftarrow k$. The set $\{x_k : k \neq i\}$ represents the input of the i-th automaton, $\{w_{ik},\, \theta_i : k \neq i\}$ represents its memory and $f_i(x_k, w_{ik}, \theta_i : k \neq i)$ the output. Usually x_k are real numbers and

$$f_i(x_k, w_{ik}, \theta_i : k \neq i) = f\left(\Sigma_k w_{ik}\, x_k - \theta_i\right).$$

A *feed-forward network* is one whose links form no closed paths.

Automata are abstract mathematical machines with a memory which can "learn" in the sense that they are able to modify their memory (see Cybernetics and Artificial Intelligence and Cybernetics and the Integration of Knowledges). The automata figuring in the neural networks are called simply *neurons*. Perceptrons investigated in Part 2 are examples of neurons in this sense of word. Multilayered perceptrons or cybernetic networks mentioned above are examples of general neural networks with the feed-forward property. In the rest of this part we restrict ourselves to multilayered perceptrons. Other important types of general neural networks are discussed in the following parts.

The basic problem of multilayered perceptrons is to find a learning which for given inputs x^t minimizes the average squared deviation

$$D = \frac{1}{T} \sum_{t=1}^{T} \sum_{\ell=1}^{L} [f(h_\ell^t) - \varphi_\ell(x^t)]^2 \tag{14}$$

where $f(h_\ell^t)$, $1 \leq \ell \leq L$, are the network outputs (cf. e.g. (13) above) and $\varphi_\ell(x^t)$, $1 \leq \ell \leq L$, are desired correct outputs. Since h_ℓ^t depends on several layers of weights and thresholds, it is hard to generalize to this case the perceptron learning rule of (6), and even less pleasing is the numerical implementation of such a generalization.

A numerically simple recursive minimization of the deviation D of (14) is achieved by the so-called *error back-propagation rule*. One variant of such a rule for binary perceptrons has already been presented in Example 4. As mentioned there, that variant

asymptotically minimizes the squared deviation only in special cases. However, for continuous perceptrons with differentiable transfer functions (2) such a rule leads to (local or global) minima of D. Roughly speaking, the error back-propagation rule consists of a subsequent application of a rule similar to the perceptron rule (6) to the individual network layers, starting with the output layer and finishing with the input layer (i. e. the application is directed "backward" trough the network, against the orientation of the synaptic links). The iterative character of the algorithm of error back-propagation is well suited for hardware as well as software computer realizations.

Another important problem of multilayered perceptrons is what functions can be represented (i. e. learned and practically realized) by these networks. We shall discuss separately the binary and continuous perceptron networks. Functions of binary variables are known as Boolean (or logical) functions. Examples 1 and 2 introduced simple Boolean functions of two variables which can be represented by one perceptron. Example 3 introduced a more complicated example where one perceptron is not enough. As shown in Example 4, to this end is needed a 2-layered network of 3 perceptrons. In general, any Boolean function of n variables can be represented by a network of binary perceptrons with one hidden layer and a single output. As to the functions of n continuous variables, all of them which are continuous and absolutely bounded by 1 can be represented by a network of continuous perceptrons with one hidden layer and a single output. These results mean that perceptron networks can in principle learn discrete as well as continuous laws of nature from sufficiently representative empirical data. Due to this universal flexibility of perceptron networks, they found important practical applications are in pattern recognition, statistical decisions, control etc. (see Cybernetics and Artificial Intelligence and Biological and Computational Intelligence).

4. Radial Basis Function Networks

This and the next three parts describe important particular types of the general neural networks defined in Part 3. Notation used below is affine but not identical with that used in the previous parts. E.g. the dimension of input vectors x, denoted above by K, is denoted by N etc. This change of notation is motivated by our attempt to be conform with the majority of neural network literature, where the notation is not fully consistent and varies from one type of the network to another. We believe that the notational conventions used here will be helpful for the readers entering for the first time the field of neural networks.

The design of radial basis function (RBF) neural network can be understood as a curve fitting (approximation) problem in a high-dimensional space (see also Biological and Computational Intelligence). In other words the goal of learning is to find a surface in a multidimensional space that provides the best fit to the training data. The idea of approximation by radial basis functions can be traced back to the method of potential functions. From the statistical point of view, there is a close relation to the kernel-type probability density estimation introduced by Parzen and to the approximation of density functions by finite mixtures (for detailed references see Haykin).

A *radial basis function network* (RBFN) consists of several layers. The input layer is

a collection of N data sources (input neurons) with real outputs,

$$x = (x_1, x_2, \ldots, x_N) \in \mathcal{X}, \quad \mathcal{X} \subset I\!R^N. \tag{15}$$

The second (hidden) layer consists of M nonnegative radial basis functions

$$F(x|m),\ x \in \mathcal{X},\ \ m \in \mathcal{M},\ \ \mathcal{M} = \{1, 2, \ldots, M\}, \tag{16}$$

representing special neurons. "Radial" means that the network basis functions $F(\cdot|m), m \in \mathcal{M}$ are radially symmetric about some points $c_m \in \mathcal{X}$, e. g. $\mathcal{X} = I\!R^N$ and

$$F(x|m) = F\left(\frac{x - c_m}{\sigma_m}\right), \quad m \in \mathcal{M}, \tag{17}$$

where $\sigma_m > 0$ and $F : I\!R^N \to I\!R$ is symmetric about $0 \in I\!R^N$. The third linear layer performs a linear transformation of the hidden layer outputs. This transformation is specified by a $(J \times M)$–matrix

$$W = (w_{jm})_{j \in \mathcal{J}, m \in \mathcal{M}}$$

of nonnegative weights. The outputs are thus weighted sums of RBF's,

$$y_j(x) = \sum_{m \in \mathcal{M}} w_{jm} F(x|m),\ x \in \mathcal{X}, \quad j \in \mathcal{J}, \quad \mathcal{J} = \{1, 2, \ldots, J\}. \tag{18}$$

In most cases only the linear layer (i.e. the weight matrix W) is trained in a *supervised way* (i.e. using correct outputs), whereas the training of the hidden layer is usually unsupervised. For this and other reasons the underlying neural structures are also referred to as *hybrid learning networks, counterpropagation networks* or *hierarchical feature map classifiers.*

Usually the RBF's $F(x|m)$ are symmetric multivariate normal on $\mathcal{X} = I\!R^N$ with means μ_m, diagonal covariance matrices, common variances σ^2 and with a normed output.

$$y_j(x) = \frac{w_j F(x|\mu_j, \sigma)}{\sum_{m \in \mathcal{M}} w_m F(x|\mu_m, \sigma)} \quad x \in I\!R^N, \quad j \in \mathcal{J}, \tag{19}$$

where w_m are nonnegative weights. This formula may also be viewed as a generalization of the *winner-take-all rule* since the maximal output value suppresses the other outputs by simple norming.

The learning algorithms of RBF networks usually apply different techniques and "time-scales" to different layers as they perform different tasks: the hidden layer is modified slowly in accordance with some nonsupervised nonlinear optimization strategy whereas the output layer's weights are adjusted rapidly through a supervised linear optimization technique (see also Adaptive Systems).

At the hidden layer the simplest approach is to assume fixed RBF defining the hidden units whereby the location of the centers may be chosen randomly. An alternative way is to use the standard k-nearest-neighbor rule. Another possibility is to fix the form of RBF (e. g. Gaussian) and optimize their location by placing the centers only in the regions containing significant data. For the supervised learning of weights of the output layer we may use an error-correction learning rule such as the *least-mean-square* (LMS) algorithm.

In the most general form we may optimize all the free parameters of the RBF network e. g. by applying gradient descent to the LMS criterion.

5. Probabilistic Neural Networks

The probabilistic approach to neural networks naturally evolves from the general framework of statistical classification. The basic idea of probabilistic neural networks (PNN) is to approximate the class-conditional probability distributions by means of a distribution mixture whereby the components of mixtures correspond to formal neurons.

Let us recall that the standard approaches to neural networks usually start with some neural network model which is then optimized with respect to some parameters (see also Biological and Computational Intelligence). In case of PNN we consider first a general classification problem and the model of neural network is obtained by interpretation of the final statistically correct solution.

There is a similarity between PNN and the RBF neural networks (cf. previous Section). However, the RBF's are usually optimized for the sake of a multivariate interpolation or approximation of some output variables whereas the purpose of estimating distribution mixtures is the Bayesian classification of observations. Also the simplifying assumption of radial symmetry is not necessary in case of mixture components since the EM algorithm as an optimization tool is usually applicable in full generality.

In particular we assume that there is a finite set of classes Ω with a priori probabilities

$$p(\omega), \omega \in \Omega, \quad \Omega = \{\omega_1, \omega_2, \ldots, \omega_C\}$$

and random observations X from a space $\mathcal{X}$ are characterized by class–conditional probability density functions

$$P(x|\omega), \quad x = (x_1, x_2, \ldots, x_N) \in \mathcal{X}, \quad \omega \in \Omega.$$

All statistical information about the set of classes Ω given some observation $x \in \mathcal{X}$ is expressed by the Bayes formula for a posteriori probabilities

$$p(\omega|x) = \frac{P(x|\omega)p(\omega)}{P(x)}, \quad \omega \in \Omega, \quad P(x) = \sum_{\omega \in \Omega} P(x|\omega)p(\omega) \tag{20}$$

where $P(x)$ is the unconditional joint probability density of x. We assume that the posterior distribution $p(\omega|x)$ is the final solution of the statistical decision problem $\{\mathcal{X}, P(\cdot|\omega)p(\omega), \omega \in \Omega\}$. Thus, to solve the problem, we have to estimate the unknown densities, as a rule.

Applying method of mixtures we approximate the conditional densities $P(x|\omega)$ by finite mixtures but, unlike usual approaches, we assume that the components belong to a common pool. In particular, we assume that there is a finite set $\mathcal{F} = \{F(\cdot|m), m \in \mathcal{M}\}$ of probability density functions on $\mathcal{X}$ such that each conditional density $P(x|\omega)$ may be expressed as a convex combination of densities from $\mathcal{F}$:

$$P(x|\omega) = \sum_{m \in \mathcal{M}} F(x|m) f(m|\omega), \quad \mathcal{M} = \{1, 2, \ldots, M\}, \quad \omega \in \Omega, \quad x \in \mathcal{X}. \tag{21}$$

Here $f(m|\omega) \geq 0$ are some conditional probabilistic weights and the components $F(x|m)$ may be shared by all class-conditional densities $P(x|\omega)$.

By using substitution (21) we can express the joint probability density $P(x)$ in the form

$$P(x) = \sum_{m \in \mathcal{M}} F(x|m) f(m), \quad x \in \mathcal{X}, \qquad f(m) = \sum_{\omega \in \Omega} f(m|\omega) p(\omega). \tag{22}$$

As it can be seen, the set of shared component densities $F(x|m)$ naturally introduces an additional "descriptive" decision problem $\{\mathcal{X}, F(\cdot|m) f(m), m \in \mathcal{M}\}$ with a priori probabilities $f(m)$ whereby each component in the mixture (22) may correspond e. g. to an elementary situation on the input. Given an observation $x \in \mathcal{X}$, the a posteriori probabilities of components

$$f(m|x) = \frac{F(x|m) f(m)}{P(x)}, \quad m \in \mathcal{M}, \quad x \in \mathcal{X} \tag{23}$$

may be interpreted as a measure of presence of different elementary situations on input.

Formally, each neuron of a given layer realizes a coordinate function of a vector transform T mapping the input space $\mathcal{X}$ into the space of output variables $\mathcal{Y}$. We denote

$$\mathrm{T} : \mathcal{X} \to \mathcal{Y}, \quad \mathcal{Y} \subset I\!R^M, \qquad y = \mathrm{T}(x) = (\mathrm{T}_1(x), \mathrm{T}_2(x), \ldots, \mathrm{T}_M(x)) \in \mathcal{Y}. \tag{24}$$

It has been shown that the transform defined by Eqs.

$$y_m = \mathrm{T}_m(x) = \log f(m|x), \quad x \in \mathcal{X}, \quad m \in \mathcal{M} \tag{25}$$

belongs to a class of information preserving transforms minimizing the entropy of the output space $\mathcal{Y}$. In other words, the transform (24), (25) preserves the statistical Shannon information about the descriptive decision problem $\{\mathcal{X}, F(.|m) f(m), m \in \mathcal{M}\}$ and, simultaneously, minimizes the Shannon entropy $H(\mathcal{Y})$ of the transformed distribution on the output space $\mathcal{Y}$. It is easy to verify that analogous assertions hold for the original decision problem $\{\mathcal{X}, P(.|\omega) p(\omega), \omega \in \Omega\}$, too.

Note that the information preserving transform (24),(25) actually "unifies" the points $x \in \mathcal{X}$ with identical posterior distributions $f(m|x)$ and therefore the arising partition of the input space $\mathcal{X}$ doesn't cause any information loss. Instead of logarithm we could use any bijective function but the logarithmic coordinate function is important as it makes the contributions from different input variables additive in some important cases.

The principle of information preserving transform can be used for a sequential design of multilayer neural networks by transforming the descriptive decision problem repeatedly along with the training data.

Let us recall that one of the most natural features of neural networks is the possibility to connect any particular neuron with arbitrary subset of input variables. Unfortunately, in probabilistic neural networks this simple possibility is usually not compatible with a statistically correct decision-making. If we assume that each layer of a neural network is described by a mixture of component densities corresponding to neurons, then all the components must be defined on the same input space to satisfy the norming property

$$\int_{\mathcal{X}} P(x) \, \mathrm{d}x = \sum_{m \in \mathcal{M}} f(m) \int_{\mathcal{X}} F(x|m) \, \mathrm{d}x = 1.$$

Any component $F(x|m)$ defined on a subspace of $\mathcal{X}$ would cause the above integral to be infinite. Thus, all neurons must be connected with all input variables and, in this

sense, the necessity of a complete interconnection of neurons with all inputs is a direct consequence of the probabilistic description.

The undesirable complete interconnection property can be avoided by a distribution mixture including structural parameters. Making substitution

$$F(x|m) = F(x|0)G(x|m,\phi_m), \quad m \in \mathcal{M} \tag{26}$$

in Eq. (22), we obtain

$$P(x) = \sum_{m\in\mathcal{M}} F(x|0)G(x|m,\phi_m)f(m), \quad F(x|0) = \prod_{n\in\mathcal{N}} f_n(x_n|0) \tag{27}$$

where $F(x|0)$ is a nonzero "background" probability density usually defined as a product of marginals, i. e. $f_n(x_n|0) = P_n(x_n)$. The component functions $G(x|m,\phi_m)$ include additional binary structural parameters $\phi_{mn} \in \{0,1\}$:

$$G(x|m,\phi_m) = \prod_{n\in\mathcal{N}} \left[\frac{f_n(x_n|m)}{f_n(x_n|0)}\right]^{\phi_{mn}}, \quad \phi_m = (\phi_{m1},\ldots,\phi_{mN}) \in \{0,1\}^N. \tag{28}$$

By setting the parameter $\phi_{mn} = 0$ any component- specific density function $f_n(x_n|m)$ can be substituted by the respective univariate background density $f_n(x_n|0)$. We assume the number of the nonzero structural parameters ϕ_{mn} to be fixed and equal to $r, (1 \le r \le MN)$, i. e. we can write

$$F(x|m) = \prod_{n\in\mathcal{N}} f_n(x_n|m)^{\phi_{mn}} f_n(x_n|0)^{1-\phi_{mn}}, \quad \sum_{m\in\mathcal{M}}\sum_{n\in\mathcal{N}} \phi_{mn} = r. \tag{29}$$

The component functions $G(x|m,\phi_m)$ may be defined on different subspaces and simultaneously, the complexity and "structure" of the finite mixture (27) can be simplified by means of the binary parameters ϕ_{mn} set to zero.

It is an important aspect of the mixture (27) that the background probability density $F(x|0)$ cancels in Eq. (23):

$$f(m|x) = \frac{G(x|m,\phi_m)f(m)}{\sum_{j\in\mathcal{M}} G(x|j,\phi_j)f(j)} \tag{30}$$

and similarly in the Bayes formula (20)

$$p(\omega|x) = \frac{p(\omega)\sum_{m\in\mathcal{M}} G(x|m,\phi_m)f(m|\omega)}{\sum_{j\in\mathcal{M}} G(x|j,\phi_j)f(j)}. \tag{31}$$

Therefore, the a posteriori probability $p(\omega|x)$ is proportional to weighted sum of the component functions $G(x|m,\phi_m)$ which can be defined on different subspaces:

$$p(\omega|x) \approx p(\omega)\sum_{m\in\mathcal{M}} G(x|m,\phi_m)f(m|\omega). \tag{32}$$

In this way the computation of the a posteriori probabilities $p(\omega|x)$ may be confined to the relevant variables. In other words, the input connections of a single neuron can be confined to an optimal subset of input variables (neurons) by means of the binary parameters ϕ_{mn}.

The optimal choice of structural parameters ϕ_{mn} can be included into the *expectation-maximization* (EM) *algorithm*.

Example 5. Recently the structural PNN was applied to recognition of totally unconstrained handwritten numerals from the database of Concordia University in Montreal. The problem was solved in the original space of nonreduced dimension N=1024 (binary 32x32 raster). In computational experiments with randomly initialized mixture models (27) we obtained repeatedly recognition accuracy which is comparable with the results reported in literature. Unlike similar published solutions we didn't use any preceding feature extraction or feature selection which may essentially improve the classification accuracy.

6. Self–Organizing Maps of Kohonen

The idea of *self-organizing map* (SOM) proposed by Kohonen belongs to the most influential approaches to artificial neural networks. The monograph of Kohonen contains a list of about 1500 papers written on SOM since 1980, with numerous practical applications. Roughly speaking the SOM algorithm constructs a topology preserving representation of the overall probability distribution of input data. The process of self-organization is controlled by topological properties of a suitably chosen two-dimensional array (lattice) of nodes (neurons).

In particular, the self-organizing map defines a topology preserving mapping from the real input space $\mathbb{R}^N$ onto a (typically) two-dimensional array of nodes $\{r_m \in \mathbb{R}^2, m \in \mathcal{M}\}$. Every node $r_m, m \in \mathcal{M}$ of the array is associated with a reference vector

$$\mu_m = (\mu_{m1}, \mu_{m2}, \ldots, \mu_{mN}) \in \mathbb{R}^N, \quad m \in \mathcal{M} \tag{33}$$

being adapted during a competitive learning process.

It is assumed that the input vectors $x \in \mathbb{R}^N$ are received sequentially, one at a time. In the first step of the SOM algorithm the location of "response" is determined as the node $j \in \mathcal{M}$ with the reference vector μ_j having the smallest distance from the input x. Usually, the best matching node is defined in terms of Euclidean distance:

$$j = j(x) = \arg\min_{m \in \mathcal{M}} \{||x - \mu_m||\}. \tag{34}$$

In the second step the best matching reference vector μ_j of the winning node j and the reference vectors of the neighboring nodes are adapted according to Eqs.

$$\mu_m^{(t+1)} = \mu_m^{(t)} + h_{mj}(t)(x^{(t)} - \mu_m^{(t)}), \quad m \in \mathcal{M}, \quad t = 0, 1, 2, \ldots, \tag{35}$$

i. e. they are moved towards the input vector $x^{(t)}$. Here $h_{mj}(t)$ is so-called neighborhood function of the j-th node. In other words, the choice of reference vectors to be modified is controlled by the function $h_{mj}(t)$ defined externally over the two-dimensional array of nodes and parametrically dependent on the time t. The initial values of reference vectors $\mu_m^{(0)}, m \in \mathcal{M}$ can be chosen arbitrarily. If, as in our case, only a given finite set $\mathcal{S}$ of observations is available, the infinite input sequence is obtained by applying the set $\mathcal{S}$ repeatedly in cycles, i. e.

$$\{x^{(t)}\}_{t=0}^{\infty}, \quad x^{(t)} = x^{(k)} \in \mathcal{S}, \quad k = (t \ mod \ \tau) + 1$$

For convergence it is necessary that $h_{mj}(t) \to 0$ when $t \to \infty$. Also it should hold $h_{mj}(t) \to 0$ with increasing distance of the nodes $m, j \in \mathcal{M}$. A widely used form of relation is the following Gaussian function:

$$h_{mj}(t) = \alpha(t) \exp\left\{-\frac{||r_m - r_j||^2}{2\sigma^2(t)}\right\} \tag{36}$$

where $r_m, r_j \in I\!R^2$ are location vectors of the nodes $m, j \in \mathcal{M}$ and the learning-rate parameters $\alpha(t), \sigma(t)$ used to update the reference vector are monotonically decreasing functions. The success of map formation (consisting of an ordering and convergence phase) is critically dependent on the selection of the main parameters of the algorithm, namely, the learning-rate parameters and the functions. Unfortunately, there is no reliable theoretical basis for the choice of these parameters.

Let us remark that there is a great similarity between the SOM algorithm and a sequential modification of EM algorithm for m.-l. estimation of normal mixtures. As the only optimized parameters of the SOM are the reference vectors μ_m, we simplify the related mixture analogously. We shall assume fixed uniform weights $f^{(t)}(m) = 1/M$, and common unitary covariance matrix $\Sigma_m = U$ and confine ourselves only to optimization of the mean vectors μ_m. The corresponding Eq. of EM algorithm for computation of the estimates of the mean vectors has the form

$$\mu_m^{(t+1)} = \frac{1}{|\mathcal{S}| f^{(t+1)}(m)} \sum_{x \in \mathcal{S}} x q^{(t)}(m|x), \quad m \in \mathcal{M} \tag{37}$$

which can be rewritten as follows

$$\mu_m^{(t+1)} = \mu_m^{(t)} + \beta_m(x^{(t)})(x^{(t)} - \mu_m^{(t)}), \quad x^{(t)} = x^{(k)} \in \mathcal{S}, \quad k = (t \ mod \ \tau) + 1 \tag{38}$$

whereby

$$\beta_m(x^{(t)}) = \frac{1}{\sum_{i=1}^{k} f(m|x^{(i)})} \frac{F(x^{(t)}|\mu_m, U)}{\sum_{j \in \mathcal{M}} F(x^{(t)}|\mu_j, U)}. \tag{39}$$

At the first view there is an obvious similarity between the Eqs. (35) and (38) with $\beta_m(x_k)$ playing the role of corresponding function. The main difference is the periodical substitution of the updated reference vectors $\mu_m^{(t)}$ whenever $(t \ mod \ K) = 0$. Also, unlike the SOM algorithm, the corresponding functions $\beta_m(x^{(t)})$ depend on the position of $x^{(t)}$ with respect to the reference vectors $\mu_m^{(t)}$. Whereas the periodical substitution of parameters (at the end of so called epochs) is sometimes used in the SOM procedure to suppress the influence of data order, the corresponding functions defined by (39) completely eliminate any influence of the underlying two-dimensional array of nodes. In case of SOM algorithm the external topology of the twodimensional array of nodes is enforced to the corresponding reference vectors.

Let us remark that, from the point of view of estimating finite mixtures, we could try to number the obtained mixture components in such a way that their corresponding relations are close to that arising in a two-dimensional array. However, one can easily imagine that the true corresponding relations between the component densities in a multidimensional space could be too complicated to be modeled by any two-dimensional (or three-dimensional) array of nodes.

Acknowledgements

The contribution has been supported by the Grant No. A2075703 and partially by the Complex research project No. K1075601 of the Academy of Sciences of the Czech Republic.

Bibliography

Haykin S. (1993). *Neural Networks. Comprehensive Foundation.* San Mateo CA: Morgan Kaufman. [One of the best monographs on neural networks. Exhaustive and sufficiently detailed description of the present state of art in neural networks with extensive bibliography.]

Hebb D. O. (1949). *The Organization of Behavior: A Neuropsychological Theory.* New York: Wiley. [Classical work known especially for the first formulation of the fundamental principles of synaptical learning and neural assemblies.]

Kohonen T. (1997). *The Self-Organizing Maps.* New York: Springer Verlag. [One of the latest monographs on the well known and very influential topological principle of learning in neural networks proposed by Kohonen.]

Müller B., Reinhardt J. and Strichland M. T. (1995). *Neural Networks. An Introduction* (2nd edition). New York: Springer. [An excellent monograph on neural networks and especially on multilayer perceptrons and related aspects.]

Rosenblatt F. (1962). *Principles of Neurodynamics.* Washington, DC: Spartan Books. [A famous book of the well known inventor of perceptron and author of the perceptron convergence theorem.]