INTRODUCTION TO NEURAL NETWORKS AND THEIR USE IN REMOTE SENSING

Markus Törmä
Helsinki University of Technology
Institute of Photogrammetry and Remote sensing
Espoo, Finland

Abstract

The neural networks are discussed as an alternative to conventional programmed computing. First, the concept of neural network and their various charasteristics are introduced. Then different neural networks are represented with their applications in remote sensing.

1. INTRODUCTION

Programmed computing has dominated information processing for almost 50 years. Solving a problem using programmed computing means making an algorithm with certain rules and coding these using some computer language. So that our algorithm would perform well, we must know everything about the problem. For some problems neurocomputing using neural networks makes possible to develop information processing system for which the algorithms or rules are not known. This paper provides an introduction to neural networks and their use in remote sensing.

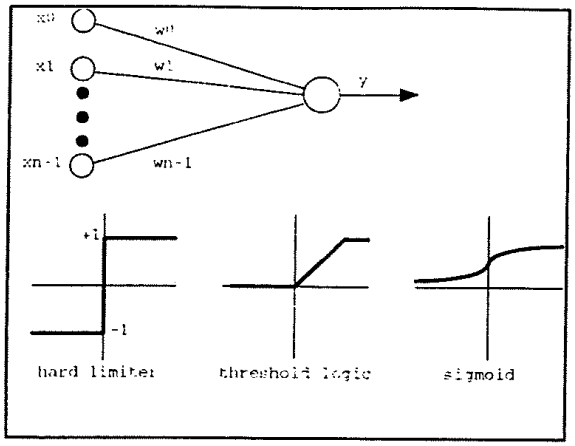
2. NEURAL NETWORK

A neural network is a parallel, distributed signal or information processing system consisting of simple processing elements. Those processing elements can possess a local memory and carry out localized information processing operations. Processing elements are highly connected via unidirectional signal channels called connections. The connections are usually weighted and those weights are adapted during training of the network. A neural network can be made using special hardware technology or conventional programming languages. Neural networks try to imitate biological nervous system and its properties like memory and learning /1//2/.

Figure 1.

Processing element, which sums N weighted inputs and passes the result through transfer function. At the bottom of the figure three most common transfer functions are found. The output of the processing element is evaluated by using formula:

$$y = f\left(\sum_{i=0}^{N-1} w_i x_i - \theta \right).$$



The simplest neural network has only one processing element. In this example, a processing element sums N weighted inputs and passes the result through transfer function as shown in Figure 1. The processing element is characterized by threshold θ and by the type of the transfer function. More complex processing elements may include temporal integration or other types of time dependencies and more complex mathematical operations than summation. More complex neural networks may include many processing elements or many layers of processing elements (Figure 2) /2/.

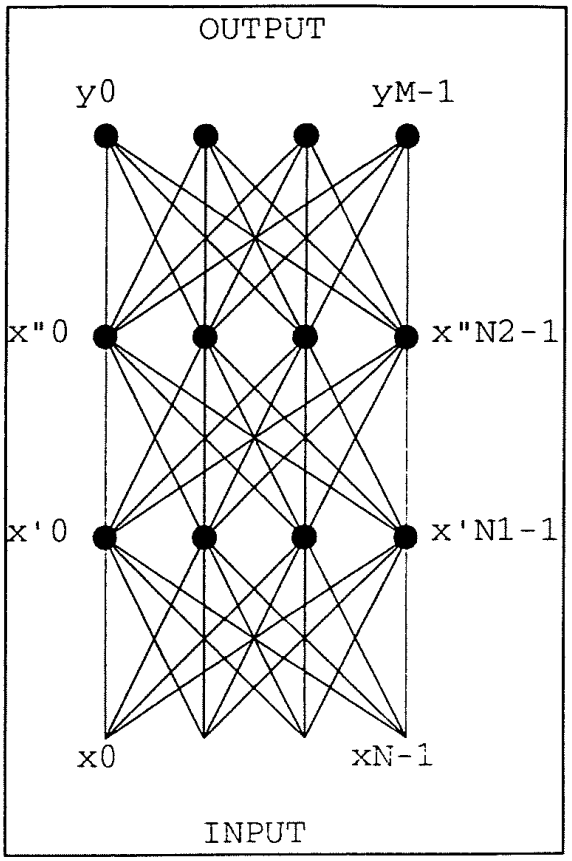
Neural networks are specified by the net topology, processing element characteristics like transfer functions, thresholds, and mathematical operations, as well as training and learning rules. These rules specify an initial set of weights and indicate how weights should be adapted during use to improve performance.

2.1 Applications

Neural networks can be used for different tasks. First, they can define which class respond best to some input. This case is normal classification problem. Secondly, neural network can be used as an associative memory. In that case class exemplar is desired and the input pattern is used to determine which exemplar to produce. Associative memory is very useful, when only part of input pattern is known and complete pattern is required. Third task is to divide inputs into different groups or to make vector quantization. In that case neural network is used to form clusters or compress inputs /2/.

Figure 2.

Complex neural network, which has three layers, N input elements, N1 processing elements in first hidden layer, N2 processing elements in second hidden layer and M processing elements in output layer.



2.2 How neural networks are trained

The most usual training method for neural network is called supervised training. For each input to the network, the correct output is supplied. If the output of the network is not the desired correct output, weights of network are adapted so that they can produce correct output next time. In the case of graded or reinforcement training , the network is given inputs, but is not supplied with desired outputs. Instead, it is occasionally given a "grade" or "performance score" that tells it how well it has done overall since the last time was graded. In self-organization or unsupervised training, the network is given only inputs. From these inputs, it is expected to organize itself into some useful configuration /1/.

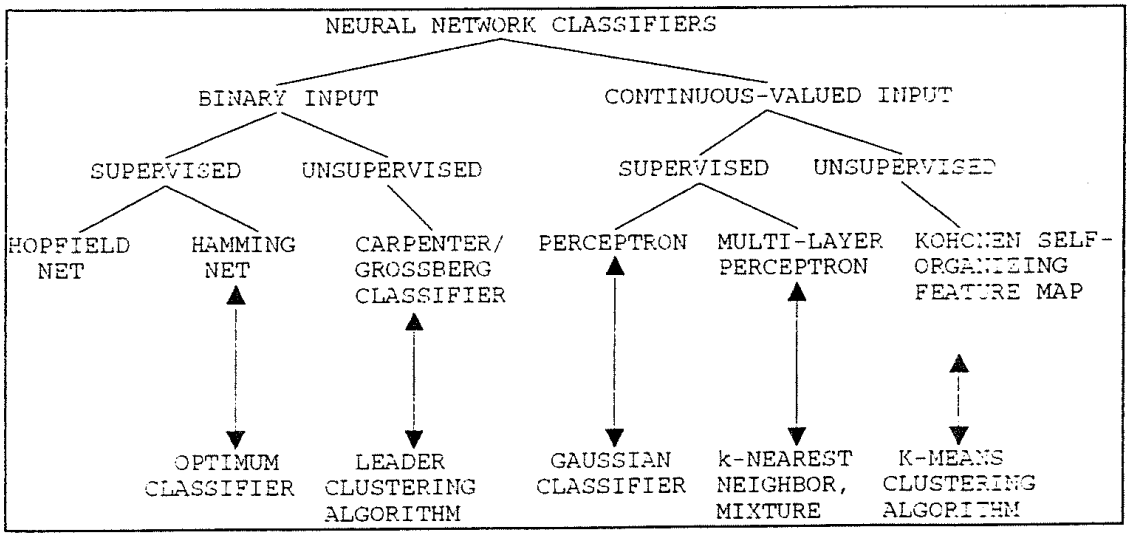
2.3 One way to separate different neural networks

A taxonomy of six neural networks that can be used for classification in presented in Figure 3. This taxonomy is first divided between networks with binary or continuous valued inputs. Below this, networks are divided between those trained with or without supervision. Nets trained with supervision such as the Hopfield network and perceptron are usually used as associative memories or classifiers. Networks trained without supervision such as the Kohonen self-organizing feature map, are used as vector quantizers or to form clusters.

The algorithms listed at the bottom of Figure 3 are those classical algorithms which are most similar or perform the same function as the corresponding neural network /2/.

Figure 3.

A taxonomy of six neural networks which can be used for classification. Classical algorithms which are most similar or perform the same function as the corresponding neural network are presented at the bottom of figure.



2.4 Other features that can be used to separate different neural networks

2.4.1 Transfer functions

One important feature of neural networks is that they are usually nonlinear and thus they can successfully make nonlinear transforms. Basic reason for this nonlinearity is that transfer functions of processing elements are nonlinear functions. Figure 1 illustrates three common types of transfer functions; hard limiter, threshold logic and sigmoid transfer functions /2/.

2.4.2 Learning laws

If the connections of the neural network are weighted, the weights are usually adapted during the use of network. In this case, the set of all possible weight vectors determines the set of all possible information processing configuration for network (assuming, the weights are the only adaptive elements in the network).

For neural networks using weight training, learning law should be developed so that it will efficiently guide the weight vector to a location that yields the desired network performance. So, most learning laws are formulated with a specific goal in mind. A commonly encountered type of goal is to move weight vector to a position that yields a network that minimizes or maximizes some particular global cost or performance function, such as mean squared error for example. Learning laws that do not attempt to optimize cost functions typically have a goal that can be expressed in behavioral or mathematical terms, such as learning the average of an input signal over a period of time /1/.

2.4.3 Different types of neural networks

Two main classes of networks are associative and mapping networks. Additional classes of networks are spatiotemporal, stochastic and hierarchical networks.

Associative networks are simple data tranformation structures. They are networks with single functional layer that associates one set of vectors with

another set of vectors. More precisely, associative networks are designed to map vectors $x_1,...,x_L$ into vectors $y_1,...,y_L$. Usually vectors x belong to $\mathbb{R}^n$ and vectors y belong to $\mathbb{R}^m$. If we think of the action of the network as being described by an function F, then our goal is to have $F(x_i) = y_i$, for i = 1,2,...,L. Also, there can never be two different y vectors assigned to the same x vector. Examples of associative networks are Hopfield and Linear Associator Networks.

Mapping networks are multi-layer data transformation structures. They can have many layers and many processing elements. Learning laws and processing element characteristics depend on application. Mapping networks try to approximate transform f: $A \subset \mathbb{R}^n \rightarrow \mathbb{R}^m$, from a bounded subset A of n-dimensional Euclidean space to to a bounded subset f[A] of m-dimensional Euclidean space. Network is trained using examples $(x_1,y_1),...,(x_K,y_K)$ where x_i is input and y_i is desired output, so that the network can perform function $y_i = f(x_i)$. Examples of mapping networks are multi-layer perceptron with back-propagation and Kohonen self-organizing feature map /1/.

2.5 Advantages of neural network classifiers

The neural networks are useful in classification, because formal models do not have to be constructed to capture the complexity and variability of the objects to be detected. Also, the ability to learn is not dependent on assumptions about underlying statistical distributions. The neural networks are relatively tolerant of missing data and noise within data. Most neural networks use nonlinear adaptation of weight vector, which allows the network to perform more realistically on nonlinear data. They appear to be a very promising approach to problems that involve the association of elements in a set with elements in another set (multisource classification). When the form and parameters of the underlying distributions can be accurately estimated by the training data, statistical classifiers may be preferable to neural network classifiers which usually require long training time and may converge to wrong solution /3//4//5//6/.

3. NEURAL NETWORKS SUITABLE FOR REMOTE SENSING

Neural networks which are suitable for remote sensing are usually mapping networks, where the main application is classification. Associative networks can be used in expert systems.

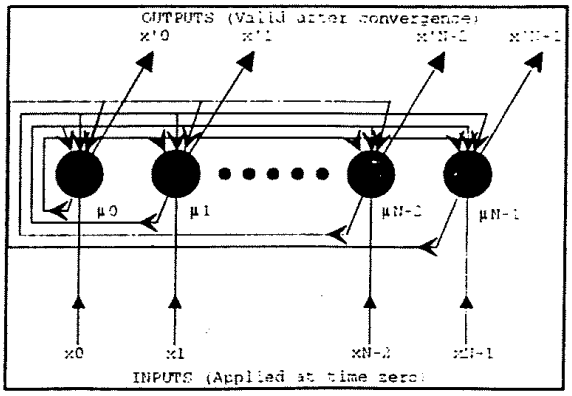
3.1 Hopfield network

The Hopfield network is usually used with binary inputs. This network can be used as an associative memory or to solve optimization problems. An assosiative memory is very useful, when part of an input is known and the complete pattern is required.

The Hopfield network is shown in Figure 4. This version has N processing elemets containing hard limiting transfer functions and binary (-1 or +1) inputs and outputs. The output of each processing element is fed back to all other processing elements via weights. Weights are set using the given recipe /2/ from exemplar patterns for all classes. Then an unknown pattern is imposed on the network at time zero by forcing the output of the network to match the unknown pattern. After this initialization, the network iterates in discrete time steps. The network is considered to have converged when outputs no longer change on successive iterations. The pattern specified by the processing element outputs after convergence is the output of the network. More detailed algorithm with formulas is in /2/.

When the Hopfield network is used as an associative memory, the network output after convergence is used directly as the complete restored memory. When the network is used as a classifier, the output after convergence must be compared to exemplars to determine if it matches an exemplar exactly. If it does, the output is that class whose exemplar matched the output pattern. If it does not, then "no match" result occurs.

Figure 4.



Hopfield network, which can be used as associative memory. An unknown pattern x is imposed on the network at time zero and the network iterates until convergence. The pattern x' specified by the processing element outputs is the output of the network. μ0,μ1,..., μN-1 are outputs of the processing elements during iteration.

3.1.1 Applications

Hopfield network has been used to solve problems in stereo vision, image segmentation, and expert systems.

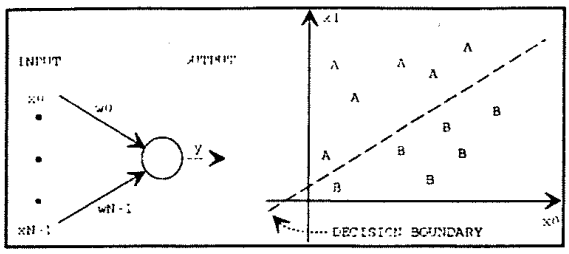
Hopfield network can be used to optimize the correspondence problem for a set of features extracted from a pair of stereo images. Each processing element in the network represents a possible match between a feature in the left image and one in the right image. Correspondence is achieved by initializing each processing element that represents a possible match and then allowing the network to settle down into a stable state. Matching points have been achieved (feature detection) using the Moravec operator /7/.

Stereo vision system can be based on neural networks only. In this approach there are four networks for different tasks. Feature detection has been done by back-propagation network. Hopfield network has been used in pattern matching, stereo fusion, and decision. The task of pattern matching network is to find corresponding points between left and right images. The stereo fusion network match other points between left and right images. The compound decision network is employed to complete cooperative decision between pattern matching and stereo fusion networks to enforce the reliability of the matching result /8/.

Modified analog Hopfield network has been used in image segmentation. In this approach the pixels of the image are to be classified into one of the M segments. For an image with NxN pixels to be classified to M classes, NxNxM processing elements are needed. Although the amount of prosessing elements is quite huge, by using analog implementation it is possible to do fast. Using this method it is possible to segment images with very large noise /9/.

In /10/ Kozato et al introduce expert system, which uses neural networks to help rule-based problem solving. The system is divided into four layers: the Hopfield network, the Proposition collector units, the Proposition connector units and input/output ports. All propositions necessary for a particular problem are mapped onto the Hopfield network and the rules are represented in a network consisting of Proposition collector units, the Proposition connector units and the links.

Figure 5.



Perceptron which classifies inputs to classes A or B. Perceptron divides inputspace into two subspaces separated by hyperplane. The output of perceptron is evaluated using formula:

$$y = f\left(\sum_{i=0}^{N-1} w_i x_i - \theta \right).$$

3.2 Perceptron

Perceptron is a very simple neural network which can be used to recognize patterns (Figure 5). The goal of the perceptron is to find hyperplane which separates two classes. The weight vector of perceptron defines the hyperplane. Classes have to be linearly separable for perceptron to separate them. If classes are non-linearly separable, perceptron does not converge. It is possible to overcome this inconvenience with some modification.

Weight vector is adapted iteratively using supervised learning. Every time that perceptron misclassifies, weight vector is adapted so that it minimizes the classification error. Processing element sums weighted inputs and passes the result through the transfer function. Usually the transfer function is a hard limiting transfer function. There are different kinds of learning laws, but the Widrow/Hoff learning law with its variants is the most common.

Because of the simplicity of perceptron its decision regions are very simple. To overcome this simplicity, we can build a multi-layer perceptron. It has many single perceptrons in one layer and many layers. This kind of structure can form arbitrary shaped decision regions. Main difficulty has been that we haven't had an efficient training algorithm for them. Error back-propagation was the first successful training algorithm for the multi-layer perceptron /1//2/.

3.3 Back-propagation

Back-propagation network is the most used network which uses supervised learning.

Theoretical basis for back-propagation is Kolmogorov's theorem /11/. In this theorem it is said that any continuous function $f:[0,1]^n \rightarrow \mathbb{R}^m$ can be implemented exactly by a three-layer feedforward neural network. There should be n processing elements in first input layer, 2n+1 processing elements in the middle layer and m processing elements in output layer. But main disadvantage with this theorem is that it promises only that this kind of transform is possible. Theorem does not tell which kind of functions we should use.

The information processing operation that back-propagation network are intented to carry out is the approximation of a bounded mapping or function $f:A \subset \mathbb{R}^n \rightarrow \mathbb{R}_m$, from a compact subset A of n-dimensional Euclidean space to a bounded subset f[A] of m-dimensional Euclidean space. This is made by training on examples $(x_1,y_1),...,(x_k,y_k),...$ of the mapping, where $y_k = f(x_k)$.

The principle of the training algorithm is to move the weight vector so that error function is minimized. Error function is minimized by using gradient method. With this principle we get a learning law which is called the generalized delta rule. The transfer functions of the processing elements are sigmoid transfer functions. Because of transfer function, the decision regions are formed by curves.

Although Kolmogorov's theorem proves "three layers are enough", it is often essential to have more layers. This is because for many problems an approximation with three layers would require an impractically large number of hidden units, whereas an adequate solution can be obtained with a tractable network size by using more than three layers. Difficulties with the back-propagation network are that training algorithm can find a local minimum of error surface instead of the global minimum and the algorithm is very slow because it moves with little jumps on the error surface /1//2/.

3.3.1 Applications

Back-propagation network has been used in different classification, image compression, and feature extraction applications.

Back-propagation network has been used to classify satellite images into different classes. In /12/ the input data was three Landsat-TM images and a digital elevation model. Training set was near 8000 pixels and planning set was 800 pixels. Rest of the pixels from training set were used to testing. Authors made experiments with different combination of images and compared the

result to the maximum likelihood classifier. They noticed that in all cases back-propagation network performed better than statistical classifier.

In /13/ Tsang et al have used back-propagation for the inversion of snow parameters from passive microwave remote sensing measurements. Those parameters were mean-grain size of ice particles in snow, snow density, and snow temperatire. The absolute percentage errors for mean-grain size of ice particles and snow density were less than 10% and the absolute error for snow temperature was less than 3°K.

Method called Cottrell/Munro/Zipser compression can be used to compress digital images. In this case, back-propagation network carries out principle component analysis to reduce dimensionality of input dataset. Besides compressing images, the method can be used in feature extraction /1/.

3.4 Kohonen self-organizing feature map

The processing elements of the network are made competitive in a self-organizing process and winning processing element whose weights are updated is chosen by some criteria. The basic idea is that the weight vectors of the processing elements approximate the probability density function of the input vectors.

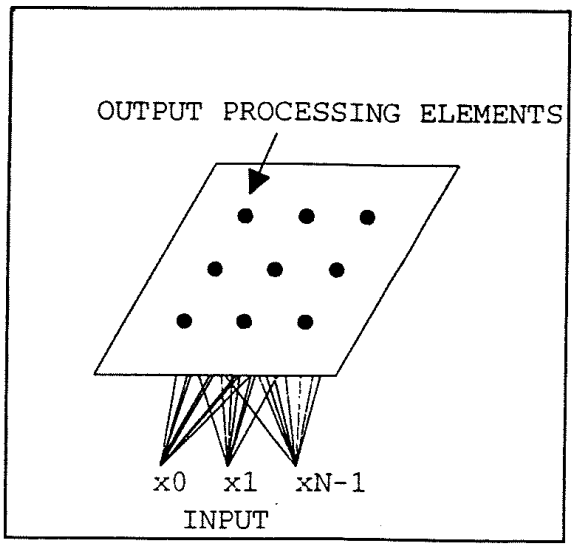
Instead of learning a mapping f: $A \subset \mathbb{R}^n \rightarrow \mathbb{R}^m$ like in supervised learning (for instance back-propagation), the self-organizing map learns a continuous topological mapping f: $B \subset \mathbb{R}^n \rightarrow C \subset \mathbb{R}^m$ using unsupervised learning. This is nonlinear mapping from n-dimensional space of input vectors to m-dimensional space of weight vectors. Weight vectors are arranged so that processing elements, which are topologically near each other, are sensitive input vectors which are physically similar. So the weight vectors are arranged in a natural way.

Map contains M processing elements and every processing element receives inputs $x_1,...,x_n$ from N input elements (Figure 6). Every input x_i has its own weight w_{ij}. Dimensionality of the map depends on application, usually it is a two-dimensional square.

Winning processing element is found by computing the intensity I_j of the processing element, $I_j = D(w_j,x)$ where $w_j = (w_{1j},...,w_{nj})$, $x = (x_1,...,x_n)^T$ and D is distance function. Usually the distance function used is Euclidean distance. Winning processing element is the processing element with smallest intensity.

Figure 6.

Two-dimensional self-organizing feature map. Every input element has been connected to every output processing element via weighted connection.



Weight vectors are adapted by using Kohonen learning law. This learning law moves the weight vector toward the input vector. As result of a learning process, probability density function is reflected to weight vectors /1//14/.

3.4.1 Applications

When self-organizing map is used for classification, map is first trained by unsupervised training. After that supervised training is made or classified training set is fed to map, one vector at time. At this stage all or some processing elements of the map are classified, depending on how large our training set is. When testset is classified, the nearest classified processing element is searched for each test vector /15/.

The self-organizing map can be used for clustering data same way, but without supervised training. In this application the map approximates the probability density function. This is made by feeding all dataset to map and counting how many times each processing element responses to input vectors. After that the map is clustered by searching peaks and valleys from the map /15/.

Shape recognition can be done by extracting features from different shapes and clustering these features by self-organizing map /16/. Another way to do shape recognition is to represent shapes using parametrized representation and to use

self-organizing map determining these parameters. This method is similar to a Hough-transform, but its advantages are higher accuracy, smaller storage and high computational speed /17/.

Identification of stochastic textures can be done by self-organizing map. The idea is to utilize co-occurrence matrices at different resolution levels and to let the self-organizing map take care of the clustering problem /18/. Another way to do it is to use autoregressive models. In this case, one processing element of map learns to detect one typical texture model /19/.

3.5 Learning vector quantization

Learning vector quantization (LVQ) is a supervised, non-parametric classification algorithm. They can be used to improve the performance of Kohonen self-organizing map. Main idea is to approximate optimal decision regions of classes without computing probability density function. LVQ-methods divide the examined region into smaller regions and one reference vector responds to one small region. These reference vectors approximate the probability density function of classes. After training, classified vector is classified by searching the nearest reference vector.

The idea behind LVQ1-algorithm is to move the reference vector toward those input vectors which are classified correctly. If the classification is not correct, reference vector is moved away.

In LVQ2-algorithm reference vector is adapted if the input vector is near the decision boundary of classes. So, we have to define a "window" which is on both sides from medium of reference vectors (Figure 7). Reference vectors are adapted if input vector is in this "window" and the nearest reference vector belong to a wrong class and the second nearest vector belong to a correct class.

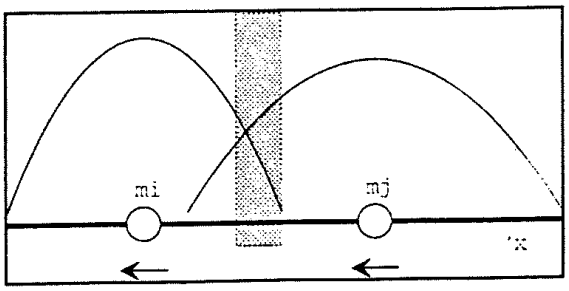


Figure 7.

"Window" which is used with LVQ2- and LVQ3-algorithms. Curves represent class distributions of x samples.

There are two disadvantages in the LVQ2-algorithm. The distance between two reference vectors most likely decreases and if the reference vector is in an optimal place, it moves away. These disadvantages can be eliminate if reference vectors are always adapted when the other reference vector belongs to a correct class and another reference vector belongs to a wrong class. Also if both reference vectors and the input vector belong to a correct class, both reference vectors are adapted /14/.

3.5.1 Applications

Usually LVQ-algorithm are used to make the classification result of the self-organizing map better /15/. It is possible to get reference vectors using other ways than weight vectors of self-organizing map /20/.

4. CONCLUSIONS

Neural networks are an interesting field of technology and it is rapidly developing and changing. Continuously new ideas concerning theory and applications are presented. Main disadvantage of neural networks is the lack of covering mathematical analysis. Also they try to behave like biological nervous system, but nobody knows how a biological nervous system (human brain) exactly works.

In practice, neural networks have shown their power. Their main charasteristics are parallel, distributed processing and simple processing elements. All these charasteristics mean, if they all can be done, high computational speed. Usually this means special processors or analog computing, but some in cases neural networks can be very useful when they are implemented using digital computer (self-organizing map). Also neural networks provide a greater fault tolerance than conventional computers because there are many simple processing elements and each processing element has its own local connections. Damage in some processing element or connection does not affect the overall performance noticeably. Using neural networks it is possible to build real-time information processing system.

REFERENCES

/1/ R. Hecht-Nielsen
Neurocomputing
Addison-Wesley, 1989

/2/ R.P. Lippmann
An Introduction to Computing with Neural Nets
IEEE ASSP Magazine, April 1987, pp. 4-22

/3/ J.A. Benediktsson, P.H. Swain, O.K. Ersoy
Neural Network Approaches Versus Statistical Methods in Classification of Multisource Remote Sensing Data
IEEE Transactions on Geoscience and Remote Sensing
Vol. 28, No. 4, July 1990, pp. 540-552

/4/ J.A. Parikh, J.S. DaPonte, M. Damodaran, A. Karageorgiou, P. Podaras
Comparison of Backpropagation Neural Networks and Statistical Techniques for Analysis of Geological Features in Landsat Imagery
Applications of Artificial Neural Networks II
SPIE Vol. 1469, 1991, pp. 526-538

/5/ J.R. Brown, D. Bergondy, S. Archer
Comparison of Neural Network Classifiers to Quadratic Classifiers for Sensor Fusion
Applications of Artificial Neural Networks II
SPIE Vol. 1469, 1991, pp. 539-543

/6/ G.F. Hepner, T. Logan, N. Ritter, N. Bryant
Artificial Neural Network Classification Using a Minimal Training Set: Comparison to Conventional Supervised Classification
Photogrammetric Engineering & Remote Sensing
Vol. LVI, No. 4, April 1990, pp. 469-473

/7/ N.M. Nasrabadi, C.Y. Choo
Hopfield Network for Stereo Vision Correspondence
IEEE Transactions of Neural Networks
Vol. 3, No. 1, January 1992, pp. 5-13

/8/ G. Loung, Z. Tan
Stereo Matching Using Artificial Neural Networks
ISPRS 1992, Vol. XXIX, Part B3, Comm. III, pp. 417-421

/9/ C.-L. Huang
Parallel Image Segmentation Using Modofied Hopfield Model
Pattern Recognition Letters
Vol. 13, No. 5, May 1992, pp. 345-354

/10/ F. Kozato, Ph. DeWilde
How Neural Networks Help Rule-Based Problem Solving
Artificial Neural Networks (ICANN-91)
Vol. 1, pp. 465-470

/11/ A.N. Kolmogorov
On the Representation of Continuous Functions of Many Variables by Superposition of Continuous Functions of One Variable and Addition
Dokl. Akad. Nauk USSR, 114, 1957, pp. 953-956

/12/ G. Pasquariello, P. Blonda
Multitemporal Remote Sensing Data Classification Using Neural Network
ISPRS 1992, Vol. XXIX, Part B3, Comm. III, pp. 922-929

/13/ L. Tsang, Z. Chen, S. Oh, R.J. Marks
Inversion of Snow Parameters from Passive Microwave Remote Sensing Measurements by a Neural Network Trained with a Multiple Scattering Model
IGARSS 1991, Vol. 4, pp. 1965-1967