

OPTIMUM STRUCTURE FOR A RADIAL BASIS FUNCTIONS BASED EQUALIZER

Adrian G. Bors and Moncef Gabbouj
Signal Processing Laboratory
Tampere University of Technology
Tampere , Finland

Abstract : A solution of a nonlinear equalizer based on the radial basis functions neural network is given in this paper. We consider a bipolar signal which passes through a dispersive channel and is corrupted by additive noise. When the distortion caused by the channel is nonlinear, the classical methods fail. The task of signal recovery is viewed as a pattern recognition problem, where each transmitted signal has been assigned a class and the equalizer's task is to find an optimum boundary between the classes. We give a solution for the minimum necessary number of hidden units for a radial basis functions (RBF) neural network implementation for the equalizer. Simulation results, where the RBF neural network is used as an equalizer are provided.

1. INTRODUCTION

In data transmission, the signal is distorted by the channel and by the noise. The equalizer has to reconstruct the transmitted signal sequence as accurately as possible, based on a finite number of channel observations. In this paper, we consider the case when the channel introduces nonlinear distortions, and the received symbols are mixed in such a way that usual equalization methods are very difficult to distinguish them.

An artificial neural network (ANN) can be used to model an unknown input-output relation, and it is very suitable to model a nonlinear function. A neural network can be regarded as a system which produces an output y , when x is the input, based on a conditional probability $p(y|x, \theta)$, where θ is the parameter vector specifying the network. This it is found after an efficient training with a selected set of observations. The neural networks capability of learning makes them suitable to implement adaptive equalizer structures as previously explored in some studies [2-4].

Radial basis functions neural network is a feed forward ANN which has locally tuned units. Overlapping classes in which the elements of one class are spread through the elements of the other classes, can be described through a combination of such locally tuned units.

The consistency of the implemented function by the

neural network with the desired function, which it has to approximate, depends on the complexity of the network given by the number of adaptable parameters of the network. Most of the neural networks developed to date have focused on training the parameters of a fixed network configuration selected a priori.

The paper is organized as follows. The equalization task is provided in section 2, the RBF neural network model in section 3, the training algorithm for a RBF neural network in section 4, and simulation results are given in section 5.

2. THE NONLINEAR EQUALIZATION

We consider the case of a bipolar signal in which the transmitted signal can take only the values 1 or -1 with an equal probability. At the receiver, the decision is made symbol by symbol in which, is given to a received signal one of the possible values: 1 or -1 assumed to be independent one of another. Channel equalization can be viewed as a classification problem where each received symbol is a pattern and the transmitted symbol is its corresponding class. A pattern recognition problem can be split in two stages; the first stage involves an appropriate selection of the characteristic features for the signals; the second is the classification of the patterns in the features space.

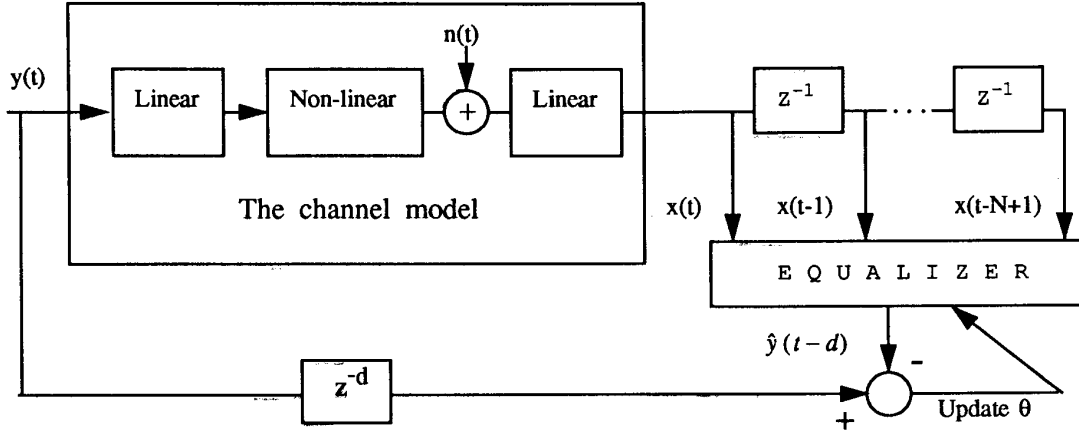


Fig. 1 The adaptive equalizer, where θ is the parameter specifying the equalizer structure.

The equalizer takes the decision to give a pattern $x(t)$ to one of two classes. We can write the a posteriori probabilities for the received symbols:

$$P(H_1|x) = \{x(t) \in \mathcal{R}^N \mid y(t-d)=1\}$$

$$P(H_0|x) = \{x(t) \in \mathcal{R}^N \mid y(t-d)=-1\}$$

The classifier assigns the vector x to class H_1 , according to Bayes rule:

$$P(H_1|x) > P(H_0|x) \Leftrightarrow \quad (1)$$

$$\frac{p(x|H_1)P(H_1)}{p(x)} > \frac{p(x|H_0)P(H_0)}{p(x)} \quad (2)$$

where $p(x|H_0)$ and $p(x|H_1)$ are the a priori probabilities and there is no loss associated to the classes. The decision boundary is obtained for the equality in relation (2). An equalizer has to estimate these probabilities and thus to find an optimal boundary between the classes.

The estimation of the transmitted signal must be done based on the distorted data by the channel and additive noise. The equalizer uses the last N channel observations $x(t), x(t-1), \dots, x(t-N+1)$ to estimate the transmitted symbol $y(t)$ (where N is integer and is called the order of equalizer). Thus, the moments $t-i, i=0..N-1$ define a N -dimensionally feature space where the received signal can be represented as a vector. Often a delay d is introduced so at the moment t the equalizer estimates the input symbol $y(t-d)$.

The channel model can be in general represented as $x(t) = f(y(t), \dots, y(t-m); \Theta) + n(t)$ (3) where $f(\cdot)$ is a non-linear function, Θ is the channel parameter vector and $n(t)$ is additive noise. From physics reasons, if $y(t)$ is a finite value then $x(t)$ is also finite and the dynamics of this model is stable. Both the functional $f(\cdot)$ and Θ can be time varying and in this case the equalizer structure is inherently adaptive. During the training period, a known signal is transmitted and a synchronized version of this signal is

generated in the receiver to acquire information about the channel characteristics (fig. 1). The equalizer must adapt its parameters, as the decided symbol, to be as close as possible to the transmitted symbol. Thus it learns to implement the inverse of the channel distortion function.

The linear transversal equalizer [1] estimates the input symbol by:

$$\hat{y}(t-d) = \text{sgn}(w^t \bullet x) \quad (4)$$

where $w^t = [w_1 \dots w_N]$ denotes the vector of equalizer coefficients and x is the vector of channel observations. A linear structure can successfully reconstruct the input sequence if and only if the two classes are linearly separable. This is possible if the channel is minimum phase and in this situation, an FIR filter can recover successfully the transmitted symbol $y(t)$. When the channel is non-minimum phase or it contains some nonlinearities, the accuracy of linear filters suffers.

In the channel model, we take in consideration all possible distortions, from the transmitter filter, modulator, the transmission medium and receiver filters. The channel model consists of 3 distinct parts (fig.1):

$$x_1(t) = 0.5 \cdot y(t) + y(t-1) \quad (5)$$

$$x_2(t) = \frac{5 \cdot x_1(t) \cdot |x_1(t)| (1 - x_1^2(t))}{(1 + x_1^2(t))} \quad (6)$$

$$x_3(t) = x_2(t) + n(t) \quad (7)$$

$$x(t) = 0.5 \cdot x_3(t) + x_3(t-1) \quad (8)$$

where $y(t)$ is the transmitted signal, $x_i(t)$ is the signal in different sections of the channel and $n(t)$ is white Gaussian noise $E[n(t)]=0$ and $E[n^2(t)]=\sigma^2$. The first and the last channel elements are linear, but non-minimum phase and the middle part is considered nonlinear and memoryless.

3. RADIAL BASIS FUNCTIONS NEURAL NETWORK

In a problem of event estimation we must estimate the a priori probabilities (2). The estimation is based on a teaching set of patterns properly selected and can be done without loss of generality if all teaching patterns are labelled according to their class membership, the case of supervised learning. In the case of an equalizer structure, this labelling is done observing the different received signals (which form a pattern) and knowing the transmitted symbol (their correspondent class). The learning set consists of a pair of input pattern and its corresponding class:

$$\{x_i[h], h=1..N; C_i\}$$

A density function of given patterns can be quite general, and it is possible not to have a knowledge about its parametric description. Especially, in a case of a mixture patterns problem, where the patterns from one class are scattered through the patterns of another class, it is difficult to find a suitable parametric description. Such cases can be found in the interpolation of a nonlinear function and in the case of corrupted data with noise. Any bounded continuous function can be described with any degree of accuracy by a weighted sum of a finite number of kernel functions:

$$P(x) = \sum_{j=1}^L \lambda_j \phi^j(x) = \sum_{j=1}^L \lambda_j e^{-\alpha_j (\|x\|)} \quad (9)$$

where $\phi^j(x)$ of this type is referred to as a radial basis function because $\phi^j(x)$ depends on x only through the metrics $\|x\|$. One class of radial basis functions which is consistent with respect to the parent densities is the Gaussian function:

$$\phi^j(x) = \mathfrak{N}(x, \mu_j, \Sigma_j) \quad (10)$$

where μ_j is the mean vector and Σ_j is the covariance matrix.

Two layers are sufficient to implement the decomposition given by (9) in a neural network structure, and the basis functions are implemented by the hidden units. The inputs of the network are continuous - it can be fed with real values - and each network entry corresponds to a vector entry.

In our model, the hidden units activation function is:

$$\sigma_1^j[x_i] = \sum_{h=1}^N \left(\frac{w[j, h] - x_i[h]}{r[j, h]} \right)^2 \quad (11)$$

where i is the pattern and for each connection between inputs $h, h=1..N$ and hidden units $j, j=1..L$ corresponds two weights: $w[j, h]$ and $r[j, h]$.

The hidden unit output function is exponential:

$$\phi^j[x_i] = \exp(-\sigma_1^j[x_i]) \quad (12)$$

If we substitute (9) in (10) we obtain an unnormalized Gaussian function with a diagonal expression for the covariance matrix:

$$\phi^j[x_i] = \exp\left(-\left[x_i[1] - w[j, 1] \dots x_i[N] - w[j, N]\right] \bullet \begin{pmatrix} \frac{1}{(r[j, 1])^2} & \dots & 0 \\ \dots & \dots & \dots \\ 0 & \dots & \frac{1}{(r[j, N])^2} \end{pmatrix} \begin{bmatrix} x_i[1] - w[j, 1] \\ \dots \\ x_i[N] - w[j, N] \end{bmatrix}\right) \quad (13)$$

This function represents geometrically a hyperellipsoid where $w[j, h]$ is the centrum position and $r[j, h]$ is the shape parameter of dimension h , for the j 'th radial basis. The principals axis of each hyperellipsoid are aligned with the references axis of the input space. This type of functions are referred to, also as potential functions [6] because of their similarities with potential fields for electric charges. In this view, the hidden units have the localization property:

$$\begin{aligned} \|x_i - w[j]\| \rightarrow 0 &\Rightarrow \phi^j[x_i] \rightarrow 1 \\ \|x_i - w[j]\| \rightarrow \infty &\Rightarrow \phi^j[x_i] \rightarrow 0 \end{aligned} \quad (14)$$

They fire only for an input vector situated in a region around their function centrum, and the size of this region is given by $r[j, h]$.

The output units activation functions are a weighted sum:

$$\sigma_2^k[x_i] = \sum_{j=1}^L \lambda[k, j] \phi^j[x_i] \quad (15)$$

where k is the output unit, $k=1..M$ and $\lambda[k, j]$ is the weight associated with the connection between an output and a hidden unit. This weight shows the contribution of each hidden unit to the output decision. The outputs are sigmoidal functions:

$$y_k[x_i] = \frac{1}{1 + \exp(-\sigma_2^k[x_i])} \quad (16)$$

and their purpose is to limit the output values to the interval $[0,1]$.

For the training set, we know for each pattern x_i its corresponding class C_i and thus we know the desired outputs:

$$\begin{aligned} F_k[x_i] &= 1 \text{ for } X_i \in C_k \\ F_k[x_i] &= 0 \text{ for } X_i \notin C_k \end{aligned} \quad (17)$$

The outputs are concentrated, they show through activation the class of corresponding input pattern x_i , and the hidden units are sub-classes which make-up the output classes.

4. THE LEARNING ALGORITHM

We have to find in the case of supervised learning the optimal mapping between inputs and outputs in such of way, as the outputs to be as close as possible by the target outputs. In order to evaluate the performance of a network, we need a cost function which measures the difference between the network outputs and the true outputs. In the case of multi-layered network, one of the most used cost function is the mean squared error:

$$E_k = \int_{x \in D} p(x) |y_k(x) - \hat{y}_k(x)|^2 dx \quad (18)$$

A learning algorithm can be defined as to minimize the cost function for a given training set and relation (18) can be expressed as:

$$E = \sum_{k=1}^M \sum_{i=1}^Q (y_k[x_i] - F_k[x_i])^2 \quad (19)$$

where Q is the number of learning patterns and M is the number of network outputs.

A neural network is determined by its weights and by the number of hidden units. A smaller network doesn't have the capacity to approximate the underlying model and a too large network gives a poor prediction performance.

The backward propagation is a gradient descend technique which updates the weights as a function of the derivative of the cost function with respect to the corresponding weight.

The algorithm used for the training of the RBF neural network is a modified variant of backward propagation algorithm and consists of two distinct parts:

1. The weights updating by backward propagation
2. The network architecture updating.

The weights update:

$$weight^t = weight^{t-1} - \Delta(weight^t) \quad (20)$$

$$\Delta(weight^t) = \alpha \Delta(weight^{t-1}) + \eta \frac{\partial E}{\partial (weight^t)} \quad (21)$$

where t is the iteration from the beginning of the learning stage and $\eta, \alpha \in [0, 1]$ where η is the learning rate and α is the momentum and it is used for to speed the learning process. The momentum is established at $\alpha = 0.5$ and the learning rate varies with the learning time, and it is the same for all weights. This update is applied for all weights $weight \in \{\lambda, r, w\}$ in reverse order with their place in the network structure.

For the topology update we use an increasing method for the number of neurons. We define the number of

neural entries as equal to the number of features space coordinates N and the number of outputs equals the number of classes M . The number of hidden units is unknown and it is difficult to be estimated a priori for a peculiar task and it depends on the patterns distribution and on the type of radial basis functions used. We initialize the number of hidden units as the number of classes ($L_{initial} = M$). For the corresponding weights of these neurons, we use an initialization according to the maximum likelihood for the Gaussian distribution, as average on the class j for the weight $w[j, h]$ and variance for $r[j, h]$ on dimension h :

$$w[j, h] = \frac{\sum_{i=1}^{Q_k} x_i[h]}{Q_k} \quad (22)$$

$$r[j, h] = 2 \cdot \frac{\sum_{i=1}^{Q_k} (x_i[h] - w[j, h])^2}{Q_k} \quad (23)$$

where Q_k is the number of patterns in class k in the learning set. We first assume that each class has only one sub-class and they have Gaussian distribution. We initialize the output weights as $\lambda[k, j] = 0$ and thus, the first geometrical shape for the boundary separation surface is a hyperplane equally distant from all possible outputs. The first value for the cost function is well defined using this initialization, as $E = M \cdot Q \cdot 0.25$

In the architecture update we use the same cost function like for the weights update (19). A pattern is considered as correctly classified when:

$$|y_k[x_i] - F_k[x_i]| < 0.5, k = 1 \dots M \quad (24)$$

When the value of E does not vary from one iteration to another with more than a fixed amount, then we check if all the patterns are correctly classified or not. If some patterns are not correctly classified yet, a new neuron is added to the hidden layer $L \leftarrow L + 1$ and for its corresponding weights first values, we use the same initialization like (22) and (23), but computed only for the patterns not yet classified from one class:

$$\hat{C}_k = \{x_i | |y_k[x_i] - F_k[x_i]| > 0.5\}$$

The algorithm determines all the sub-classes which have local Gaussian distribution and classify correctly their corresponding patterns, according to the localization property (14) for the Gaussian functions. The update is for the enlarged set of weights using the same rule as (20) and (21). As many hidden units as necessary are added, to classify all the patterns in the learning set.

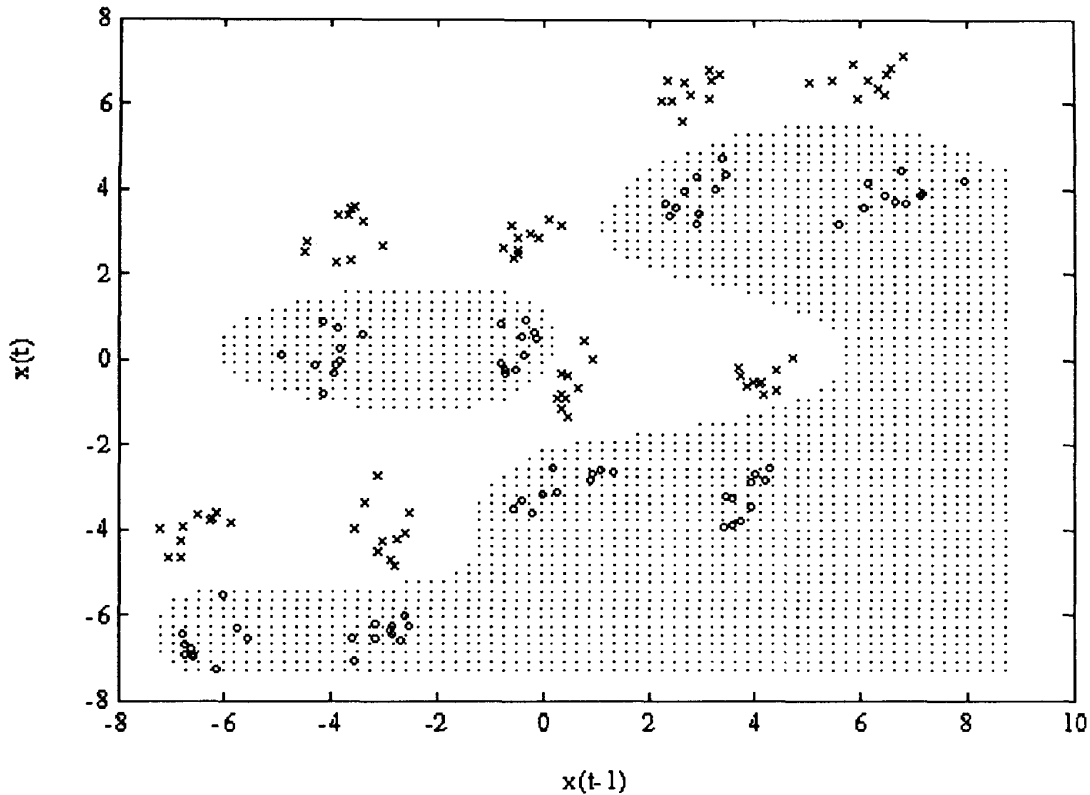


Fig. 2. Decision region obtained with a RBF neural network equalizer. The shaded area represents the set of points which when presented as inputs they are decided to 1 symbol.

5. SIMULATION RESULTS

For simulation we have assumed equal probability of occurrence for the two symbols $P(H_0) = P(H_1)$. We consider an equalizer model with order $N=2$ and without any delay $d=0$. Each decision at time t takes in consideration only the received symbol and the previous one. When we generate the learning data we have assumed known the channel degree (3) as equal to $m=3$ and we take the probability of occurrence for all the combinations possible for 4 consecutive symbols. In table 1, we present the received values for all possible situations at transmission in the case when the channel model is given by (5,6,8) without taking into consideration the noise (noise-free states). The learning set consists of 160 vectors and they are generated assuming a white Gaussian noise with $\sigma^2 = 0.16$. From fig. 2 it can be seen that these values will cluster around the noise-free states.

The neural network topology is initialized with $N=2$

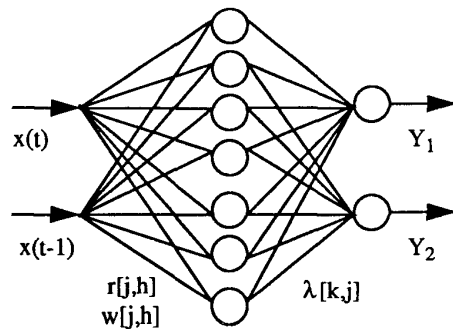


Fig. 3. The network topology obtained after learning

inputs, $L=2$ hidden units and $M=2$ outputs, one for $Y_1=1$ and another for $Y_2=-1$. After training, we obtain the final network topology with 7 hidden units, see fig. 3. The final cost function has the value $E=0.269$, and all the patterns from the learning set are correctly cla-

y(t-3)	y(t-2)	y(t-1)	y(t)	x(t-1)	x(t)	Decision
1	1	1	1	-6.49	-6.49	H ₁
1	1	-1	1	-3.95	0.375	
1	-1	1	1	0.375	-2.91	
1	-1	-1	1	2.91	3.95	
-1	1	1	1	-2.91	-6.49	
-1	1	-1	1	-0.375	0.375	
-1	-1	1	1	3.95	-2.91	
-1	-1	-1	1	6.49	3.95	
1	1	1	-1	-6.49	-3.95	H ₀
1	1	-1	-1	-3.95	2.91	
1	-1	1	-1	0.375	-0.375	
1	-1	-1	-1	2.91	6.49	
-1	1	1	-1	-2.91	-3.95	
-1	1	-1	-1	-0.375	2.91	
-1	-1	1	-1	3.95	-0.375	
-1	-1	-1	-1	6.49	6.49	

Table 1: The noise-free states

ssified. In fig.2 we present the decision boundary between the classes, the potential regions associated with each hidden unit can easily be identified.

After training, we apply different test sets consisting of symbols received after passing through the same channel model but corrupted with different noise power. The set of tests was evaluated for different signal-to-noise ratios between 0dB and +10dB, for a lot of 1120 signals for each test.

The optimum linear filter with respect the mean square error is the Wiener filter. The Wiener filter solution of order $N=2$, for this example, can be shown to be $w^1=[0.17 \ 0.25]$. We have tested both equalizer structures in the same conditions and the results are depicted in fig. 4. The Wiener filter has a big and almost constant probability of error because of the nonlinear estimation problem where it is not appropriate to be used. The ability of this RBF topology to achieve a good performance even for a higher signal-to-noise ratio is evident from these tests results

6. CONCLUSIONS

Adaptive equalization can be considered as a pattern recognition task. When a bipolar signal is passed through a dispersive channel which has also some nonlinearities in structure, the equalizer must have a

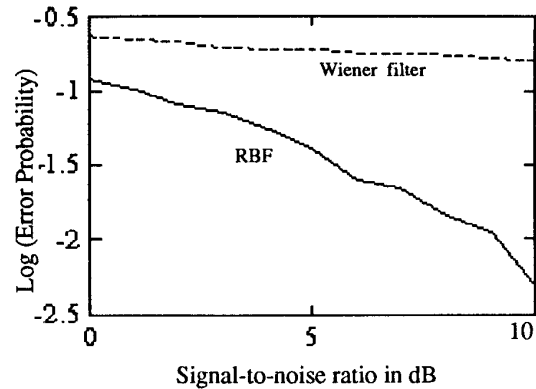


Fig. 4. The probability of error obtained for different signal-to-noise ratio when a RBF neural network is used as equalizer and comparison with a Wiener filter

nonlinear structure. The RBF network has been cited as one architecture capable to achieve a bit error rate superior to linear structures

For real implementation, the minimum necessary number of processing units must be known. We provide a solution of RBF structure with an optimum number of hidden units, assuming a particular nonlinear channel model.

REFERENCES

- [1] Shahid U.H. Qureshi, "Adaptive equalization", Proc. IEEE, Vol. 73, No. 9, 1985, pp. 1349-1387.
- [2] S. Chen, G. J. Gibson, C. F.N. Cowan and P.M. Grant "Reconstruction of binary signals using an adaptive radial-basis-function equalizer", Signal Processing, vol. 22, No. 1, 1991, pp 77-93.
- [3] S. Chen, G. J. Gibson, C.F.N. Cowan and P.M. Grant "Adaptive equalization of finite non-linear channels using multilayer perceptrons", Signal Processing, vol. 20, No. 2, 1990, pp 107-119.
- [4] S. P. Day, M. P. Davenport, D. S. Campoprese "Dispersive networks for nonlinear adaptive filtering", IEEE Workshop on Neural Networks for Signal Processing, Helsingör, Denmark, 1992.
- [5] S. Lee, R. M. Kil "A gaussian potential function network with hierarchically self-organizing learning", Neural Networks, Vol. 4, 1991, pp. 207-224.
- [6] M. Niranjan, A. J. Robinson, F. Fallside "Pattern recognition with potential functions in the context of neural networks", Proc. 6th SCIA, Oulu, Finland, Vol. I, 1989, pp. 96-103.