Reliability of Radial Basis Function - Neural Network Smart Antenna

MAJA SAREVSKA, BRATISLAV MILOVANOVIĆ, ZORAN STANKOVIĆ
Department of Telecommunications
University of Niš – Faculty of Electronic Engineering
Aleksandra Medvedeva 14, 18000, Niš
SERBIA AND MONTENEGRO
http://www.elfak.ni.ac.yu

Abstract: - This paper considers the problem of reliability of radial basis function neural network (RBFNN) based smart antenna, applied for direction of arrival estimation. One of the main parameters in these networks is the number of neurons in the hidden layer. Once this parameter is chosen another important issue is at which level we can count on a proper functionality of the hidden layer, which is mainly determined by a proper functionality of its neurons. After the presentation of the architecture and the basic concept of the RBFNN, we will discuss the problem of the hidden layer dimension. This will provide necessary information for the investigation of the reliability of these networks. Namely, we will assume a failure of some percentage of the neurons in the hidden layer and we will present the resulting mean error increase. The results of computer simulations will show us that the networks with less neurons in the hidden layer are more reliable. But when we need more precise direction of arrival estimations we should use more neurons in the hidden layer and more training samples. In this case the neurons in the hidden layer must be with long time of a proper functionality, since the mean error increase is much larger if some of the neurons fail.

Key-Words: Neural networks, radial basis functions, neurons, mean error.

1 Introduction

The field of artificial neural networks (NNs) has made tremendous progress in the past 20 years in terms of theory, algorithms, and applications. Notably, the majority of real world NN applications have involved the solution of difficult statistical signal processing problems. Compared to conventional signal processing algorithms that are mainly based on linear model, artificial NNs offer an attractive alternative by providing nonlinear parametric models with universal approximation power, as well as adaptive training algorithms. In particular, the nonlinear nature of NNs, the ability of NNs to learn from their environments in supervised as well as unsupervised ways, as well as their universal approximation property, make them highly suited for solving difficult signal processing problems.

The concept of frequency reuse has been successfully implemented in modern cellular communication systems in order to increase the system capacity. Extensive research has showed that further improvement can be achieved by employing adaptive arrays in the base stations. The main task in these systems is Angle Of Arrival (AOA) estimation in real time, after which the corresponding algorithm for beamforming could be applied. This approach together with Spatial Division Multiple Access (SDMA) scheme will rapidly increase the system capacity. Superresolution algorithms [1] have been successfully applied to the problem of Direction Of Arrival (DOA) estimation to locate radiating sources with additive noise. One of the main disadvantages of the superresolution algorithms is that they require extensive computation and as a result they are difficult to be implemented in real time. It has been shown that NNs have the capacity to track sources in real time. The paper [2] presents a generalization of the algorithm presented in [3] in such a way that the system would be able to track an arbitrary number of sources with any angular separation without prior knowledge of the number of sources. The new approach presented in [2] provides a dramatic reduction in the size of the training set required to train each smaller network. One proposal for larger reduction in network training time is to apply Probabilistic Neural Network (PNN) in the first stage [4]. In this way we do not perform classical network training, but we just design the appropriate network in the first stage, and for that we need only couple of seconds. We should mention that for the network in the first stage in [2] the time that is necessary for network training is about 5 minutes.

While there has been much success in developing nonlinear NN models, in particular multilayer (ML) networks trained by back-propagation, researchers,

and practitioners have found that, in some cases, it can be difficult to train a ML network and obtain a good performance without resorting to sophisticated algorithms. The main reason it is difficult to train a ML-NN is that such networks are nonlinear in the parameters. This means that learning algorithms typically use gradient descent approaches as a means of solving the nonlinear optimization problem. It is desirable, therefore, to obtain models, which can overcome this problem of training. RBFNNs [5,6] do offer a means of avoiding exactly this problem. RBF networks are able to model data in a local sense.

A major result that has emerged in recent years, with the growth of interest in NNs, is that multilayer perceptron (MLP), with single hidden layer, is capable of approximating any smooth nonlinear input-output mapping to an arbitrary degree of accuracy, provided that sufficient number of hidden layer neurons is used. This often is referred as universal approximation theorem. Also Park and Sandberg [7,8] had proved the universal approximation capabilities for RBFNNs. This property ensures that RBF networks will have at least the same theoretical capabilities as the well known ML networks with sigmoidal nonlinearities. The universal approximation property is shared by a rather wide range of model types. This property merely indicates that a generating function can be approximated but generally says nothing about the quality of the approximation. It is clear, however, that for solving practical problems, we may be more interested in which model is the best for a given task. The property of best approximation has been defined as an extension of the universal approximation property. In a given set of models, the model that most closely approximates the generating function, by some defined distance measure, is defined as having the property of best approximation. Thus, the best approximation is an important attribute in choosing a model type. It has been proved that RBF networks have this property.

There are many aspects analyzed in the literature for the application of the NNs. One of the important issues is the reliability of the NNs. This paper considers exactly this problem: the reliability of the RBFNNs applied for DOA estimation. Namely, an important aspect is the number of the neurons in the hidden layer, in order good performances to be achieved. Our aim was to investigate the level of degradation in the final performances of RBFNNs when some percentage of the neurons in the hidden layer is off-function.

The organization of the paper is as follows: Section 2 elaborates on the use of neural networks for direction finding. Subsection 2.1 presents the information about the number of neurons in the NN. Subsection 2.2 presents the NN training algorithm. Simulation results are presented in Section 3 and in Section 4 some conclusive remarks are summarized.

2 NN-Based Direction Finding

Let observe a linear antenna array with M elements, let U (U<M) is the number of narrowband plane waves, centered at frequency ω_0 impinging on the array from directions $\{\theta_1\ \theta_2\ .\ .\ .\ \theta_U\}$. Using complex signal representation, the received signal in the *i*th array element is:

$$x_i = \sum_{m=1}^{U} s_m(t) e^{-j(i-1)K_m} + n_i(t), \quad i = 1,2,...M \tag{1}$$

where $s_m(t)$ is the signal of the *m*-th wave, $n_i(t)$ is the noise signal received at the *i*-th sensor and

$$K_m = \frac{\omega_0 d}{c} \sin(\theta_m) \tag{2}$$

where *d* is the spacing between the elements of the array, and *c* is the speed of the light in free-space. In vector notation the output of the array is:

$$\boldsymbol{X}(t)=\boldsymbol{A}\boldsymbol{S}(t)+\boldsymbol{N}(t) \tag{3}$$

where $\boldsymbol{X}(t)$, $\boldsymbol{N}(t)$, and $\boldsymbol{S}(t)$ are:

$$\begin{aligned} \boldsymbol{X}(t)&=[\,x_1(t)\ x_2(t)\ .\ .\ .\ x_M(t)]^T \\ \boldsymbol{N}(t)&=[\,n_1(t)\ n_2(t)\ .\ .\ .\ n_M(t)]^T \\ \boldsymbol{S}(t)&=[\,s_1(t)\ s_2(t)\ .\ .\ .\ s_U(t)]^T \end{aligned} \tag{4}$$

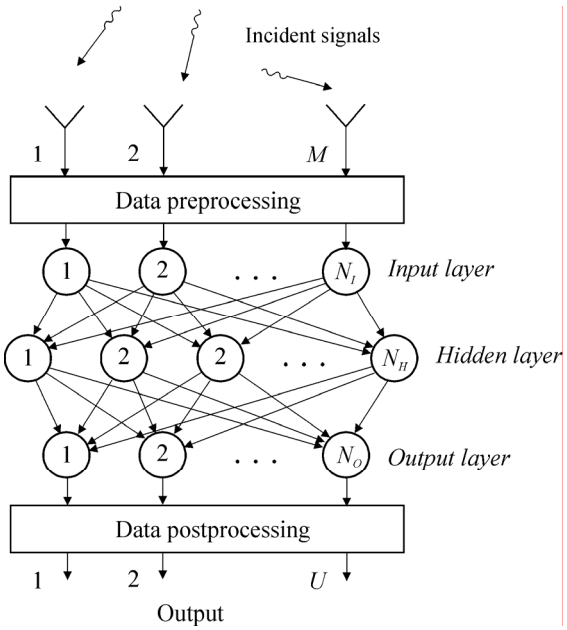

Fig.1 Block diagram of RBFNN with pre- and postprocessing stages

In (3) $\boldsymbol{A}$ is the M × U steering matrix of the array toward the direction of the incoming signals:

$$\boldsymbol{A}=[\boldsymbol{a}(\theta_1)\ \boldsymbol{a}(\theta_2)\ .\ .\ .\ \boldsymbol{a}(\theta_U)] \tag{5}$$

where $\boldsymbol{a}(\theta_m)$ is:

$$\boldsymbol{a}(\theta_m) = [1 \; e^{-jKm} \; e^{-j2Km} \; \ldots \; e^{-j(M-1)Km}]^T \quad (6)$$

The received spatial correlation matrix $\boldsymbol{R}$ of the received noisy signals can be estimated as:

$$\boldsymbol{R} = E\{\boldsymbol{X}(t)\boldsymbol{X}(t)^H\} = \boldsymbol{A}E[\boldsymbol{S}(t)\boldsymbol{S}^H(t)]\boldsymbol{A}^H + E[\boldsymbol{N}(t)\boldsymbol{N}^H(t)] \quad (7)$$

Following the Fig.1 and the above expressions we can conclude that the antenna array is performing the mapping G: $\boldsymbol{R}^U \rightarrow \boldsymbol{C}^M$ from the space of DOAs, $\{\Theta=[\theta_1,\theta_2, \ldots, \theta_U]^T\}$ to the space of sensor output $\{\mathbf{X}(t)=[x_1(t) \; x_2(t) \; \ldots \; x_M(t)]^T\}$. A neural network is used to perform the inverse mapping F: $\boldsymbol{C}^M \rightarrow \boldsymbol{R}^U$. For this task a RBFNN is used, instead of backpropagation neural network because the second is slower in training. As can be seen from Fig.1 the applied network generally has three layers of nodes, one input layer, one output, and between them there is one hidden layer of nodes. There is a sample data preprocessing block which will transform the input values in appropriate form that could be applied at the input of the NN. After the output layer there is postprocessing block, which will transform the outputs in appropriate form in order to give the estimated DOAs. In [2] the algorithms for detection and estimation stage are same, the difference is only in the number of nodes in the output layer. Namely, the number of the nodes in the output layer of the first stage (detection) is one (there is a signal gives one, and no signal gives 0), and the number of the nodes in the output layer of the second stage is determined by the angular resolution of the algorithm and the width of the corresponding sector.

There are a lot of learning strategies that have appeared in the literature to train RBFNN. The one used in [2] was introduced in [9], where an unsupervised learning algorithm (such as K-means [6,10]) is initially used to identify the centers of the Gaussian functions used in the hidden layer. The standard deviation of the Gaussian function of a certain mean is the average distance to the first few nearest neighbors of the means of the other Gaussian functions. This procedure allows us to identify the weights (means and standard deviations of the Gaussian functions) from the input to the hidden layer. The weights from the hidden layer to the output layer are estimated by supervised learning known as delta rule, applied on single layer networks [6]. With this procedure, for training we need 5min in detection stage and about 15min in estimation stage. An alternative is instead of using the same neural networks in both stages, to use different neural network in the first stage. The reason for this is the fact that the task of signal detection is a vector classification problem. Any input vector should be classified as 0 (there is NO signal in the corresponding sector) or 1 (there IS a signal in the corresponding sector). For this task an appropriate neural network is Probabilistic Neural Network (PNN), which is used in [4]. In our paper we are going to use the RBFNN, since we are going to investigate the reliability of the second stage NNs.

The main capability of NNs is their ability of generalization. The question is what types of generalization our NN is supposed to be able to perform, in order to use less training samples that we can. Although the set from which the training samples are picked for training in the detection stage is larger than that for DOA estimation stage, we will briefly analyze the case for DOA estimation stage, because of its higher difficulty in the sense of generalization (the detection stage is simple vector classification problem and generalization is much easier to be achieved). Let suppose that we have one user that is moving straightforward in some direction. We choose the training samples, according to eq. (7) with some angular step, let say $\Delta\theta_1$ degrees. The first type of generalization that the NN is supposed to perform is that after it is trained, it is expected to give satisfactory performance for samples picked with angular step smaller than $\Delta\theta_1$ degrees. Now, let say we have two users moving in parallel in some direction (this choice of parallel and linear moving is made for clear results presentation without loss of generality), and we chose the training samples for angular separation between the users of $\Delta\theta_2$ degrees. The second type of generalization that we expect our NN to perform, after it is trained, is to give satisfactory performances for some other angular separation between the users (smaller than $\Delta\theta_2$). Also, as third we can train the network for N_1 and N_2 users and we are expecting from the NN to give satisfactory performances for N_K users, where $N_1 \leq N_K \leq N_2$.

Additionally we should have in mind that we use noisy training samples, which means that in all cases the network should be able to "learn" the noise influence on the signal. Using binary signals, the network should also "understand" the nature of this signal. The interested reader can read more about this issue in [11]. Our results showed the great potential of this type of neural network for DOA estimation in the application of the antenna arrays. The main further interest is to investigate the ability of the network to perform all the types of generalization in the same time. Namely, from the results [11] it is obvious that the network is capable of doing it, but the main problem is the ill-

conditioning of the system determinant of the output linear layer, which occurs when the number of training samples is large. The point is that we must use much more training samples in order to achieve all types of generalization at once, and to avoid the ill-conditioning in the same time.

2.1 Number of Neurons in NN

The input data of RBFNN is the matrix $\boldsymbol{R}$, which is the correlation matrix of the antenna array outputs, rather than the antenna outputs themselves. The dimensionality of the input samples determines the number of the neurons in the input layer, and the transformation of the antenna array outputs to input samples of the NN, as showed in Fig.1, is performed in the block "sample data preprocessing". The dimensionality of $\boldsymbol{R}$ is M×M, and $2M^2$ could be the dimensionality of the input samples (since the NN cannot deal with complex numbers). But in order to simplify the total system, an interesting point is to reduce the number of input neurons in the NN, which means not to use all the elements of the matrix $\boldsymbol{R}$. Using the fact that the matrix Real{$\boldsymbol{R}$} is symmetric, we could use only the upper triangular part of Real{$\boldsymbol{R}$}, which gives M(M+1)/2 elements. The diagonal elements of matrix Imaginary{$\boldsymbol{R}$} are zeros and the absolute values of the elements of the upper triangular and lower triangular matrix, are same. This lead us to conclusion that we should use only the elements of the upper triangular part of the matrix Imaginary{$\boldsymbol{R}$} (without the diagonal elements), which gives M(M-1)/2 elements. So, the dimensionality of the input layer (sum of the number of imaginary parts and real parts of the elements), should be M^2, which is actually the number of neurons in the input layer. Just for comparison the number of neurons in the input layer in [2] is M(M+1), thus the input redundancy is decreased. In order to achieve satisfactory performances the number of neurons in the hidden layer should be equal or larger than the number of neurons in the input layer [12]. The number of neurons in the output layer, for DOA estimation stage, is determined by the angle width of the corresponding sector and the specified angular resolution of the network [2]. The outputs of the output layer are processed in the "postprocessing" block in order to give the final outputs, which are the DOAs.

The fact that many training samples, for DOA estimation problem, are available is very satisfactory in the sense of providing the necessary generalization. Unfortunately, we are dealing with Single Layer Perceptron Neural Network (SLP-NN) [12] at the output layer, which can deal and find a solution only for limited number of training samples. We have performed an extensive research for the capabilities and power for generalization of the output layer, and here, briefly we can expose some of our conclusions, important for this paper.

As we have mentioned earlier we are dealing with SLP-NN at the output layer, since the parameters of the hidden layer are determined with unsupervised learning. This means that we are attempting to approximate the input-output mapping for DOA estimation with number of linear equations. For example, if the number of the neurons in the hidden layer is L, and if the number of training samples is N, then the NN should deal with N linear equations with L+1 unknowns (including the bias for the node in the output layer), where the coefficients of the equations are the outputs of the hidden layer, and the unknown variables that should be estimated by the NN are the weights from the hidden to the output layer including the bias. NN with single linear layer is not much powerful for this type of problems, but it is the hidden layer that is providing the successful application of this architecture. Namely, the hidden layer is transforming the input samples into appropriate form and usually in higher dimensionality, so as to provide successful generalization. The question that should be answered in our problem is: how much neurons we should use in the hidden layer? We have concluded that the number of neurons in the hidden layer should be as large as possible but it doesn't have to be larger then M^2 if we solve the IC appearance [12], in order to reduce the cost of the total system.

The solution for IC appearance is an additional problem, which overcomes the scope of this paper. Our task in this paper is steered to the reliability of the NN depending on the number of neurons in the hidden layer, since the number of neurons in the input and output layer are determined, as previously explained. It is obvious that if we use larger number of neurons in the hidden layer we will get much powerful network, but the precise value for that number is hard to be determined. The main limitation could be the cost of the total system, namely, we are trying to use as less as possible neurons. After we determine this value, we are interested in the performance degradation after the failure of some portion of the neurons in the hidden layer.

We should stress that this case is not the same as we are using less neurons than determined in the start, since using less neurons is not the same as a failure of some neurons in the hidden layer!

2.2 NN training algorithm

As earlier mentioned we are using hybrid learning method. That means we use unsupervised learning for the parameters in the hidden layer: the centers and the variances of the Gaussian functions, and we use supervised learning for the weights from the hidden layer to the output layer. Namely, for each neuron in the hidden layer there is one Gaussian function associated to it, determined by its center $\mathbf{v_i}$ and its variance σ_i. K-means clustering algorithm is as follows [12]:

Step1: Choose L initial cluster centers $\mathbf{v_1}(1)$, $\mathbf{v_2}(1),\ldots\mathbf{v_L}(1)$ from the training set.

Step2: At the nth iterative step distribute the input patterns $\{\mathbf{x}(p), p=1\div N$, among the L custer domains, using the relation:

$$\mathbf{x}(p)\in S_j(t) \text{ if } \|\mathbf{x}(p)-\mathbf{v_j}(t)\| < \|\mathbf{x}(p)-\mathbf{v_k}(t)\| \qquad (8)$$

for all k=1,2,…L, and k≠j, where $S_j(t)$ denotes the set of input patterns, whose center is $\mathbf{v_j}(t)$.

Step3: From the results in *Step2*, compute the new cluster centers $\mathbf{v_j}(t+1)$, j=1,2,…,L. The new cluster centers are given by the rule:

$$\mathrm{v}_j(t+1)=\frac{1}{N_j}\sum_{\mathrm{x}\in S_j(t)}\mathrm{x}, \quad j=1,2,...,\mathrm{L} \qquad (9)$$

where N_j is the number of input patterns in $S_j(t)$. The cluster centers defined by (9) guarantee that the sum of the squared distances from all the points in $S_j(t)$ to the new cluster center is minimized.

Step4: If $\mathbf{v_j}(t+1)= \mathbf{v_j}(t)$ for j=1,2,…,L, the algorithm has converged and the procedure is terminated. Otherwise go to *Step2*.

Once the centers are identified, the variances of the Gaussian functions are chosen to be equal to the mean distance of every Gaussian center from its few (usually from 3 to 7) neighboring Gaussian centers.

When the centers and variances of the Gaussian functions are chosen, we are left with the task of choosing the interconnection weights from the hidden layer to the output linear layer. These interconnection weights are chosen in a way that minimizes the error function:

$$E(\mathrm{w}_i)=\sum_{p=1}^{N}E^p(\mathrm{w}_i) \qquad (10)$$

where i is the index of the neuron in the output layer, and $\mathbf{w}_i$ are the weights. The weights are found by following the well known gradient descent procedure [12], which modifies the weights by an amount proportional to the negative gradient of $E(\mathbf{w_i})$. The error function can also be written as:

$$E(\mathrm{w}_i)=\frac{1}{2}\sum_{p=1}^{N}[d_i(p)-y_i(p)]^2 \qquad (11)$$

where $d_i(p)$ is the actual output (the target) and $y_i(p)$ is gained output. Further this expression can be written as:

$$E(\mathrm{w}_i)=\frac{1}{2}\sum_{p=1}^{N}\left[d_i(p)-\sum_{j=1}^{L}w_{ij}u_j(p)\right]^2 \qquad (12)$$

where

$$u_j(p)=\exp\left(-\frac{1}{2}\sum_{k=1}^{M^2}\left[\frac{x_k(p)-v_{jk}}{\sigma_j}\right]^2\right) \qquad (13)$$

Let observe two networks with number of neurons in the hidden layer N_1 and N_2, where $N_1>N_2$. By training algorithm we could achieve same value for E. But knowing the fact that network with larger number of neurons has better generalization properties, will lead us to a conclusion that the first network will give smaller average errors at the output. Namely the value of E estimated for testing samples will be smaller for the network with larger number of neurons in the hidden layer. Let Ω_{OFF} is the set that contains the indexes of neurons that are off function and Ω_{ON} is the set of indexes of neurons that are on function. The error function will be:

$$E(\mathrm{w}_i)=\frac{1}{2}\sum_{p=1}^{N}\left[d_i(p)-\sum_{j\in\Omega_{ON}}w_{ij}u_j(p)\right]^2 \qquad (14)$$

since $u_j(p)$ for $j\in\Omega_{OFF}$ is zero. In this case:

$$u_j(p)=\exp\left(-\frac{1}{2}\sum_{k=1}^{M^2}\left[\frac{x_k(p)-v_{jk}}{\sigma_j}\right]^2\right), \quad j\in\Omega_{ON} \qquad (15)$$

Observing the expression (14) it is obvious that when some neurons are off function the value of E will be increased, since $d_i(p)$ will largely differ from $y_i(p)$. As the set Ω_{OFF} is becoming larger for both networks, the part $d_i(p)$ will be more dominant in (14). This lead us to a conclusion that in this case the both networks will give similar errors at the output (since $d_i(p)$ is same for both networks), and better generalization capabilities of the first network will be lost. Knowing the fact that the first network gives smaller errors when all neurons are on function, larger portion of neurons that are off function will lead to larger relative error increase for the network with larger number of neurons in the hidden layer.

Computer simulation results in the next section will confirm that neural networks that contain less neurons in the hidden layer are more reliable than those who contain more neurons.

3 Computer Simulations and Results

In computer simulations we were using a linear antenna array of 12 elements placed at mutual distance of one half of a wavelength, and Signal to Noise Ratio (SNR) of various values. The number of users whose signals were tracked was chosen to be 4, moving at mutual distance of 2°, in the sector wide 10°. In the training phase the angle step of training samples was 1°, and in the testing phase the angle step was 0.25° (first type of generalization explained in section 2). The number of neurons in the input level is 144, and the number of neurons in the output level was chosen to be 6 (angle resolution of 2° [2]). The achieved Mean Square Error (MSE) of the NN was chosen to be around 10^{-3}. Parameter that shows the quality of system performances is the mean error of DOA estimation for one of the users. The neurons that were off function were randomly chosen, and there were performed 50 trials of randomly chosen failed neurons of 3% to 30% with step of 3%.

As a first example we are going to see the results gained for *SNR*=10dB, and for number of neurons in the hidden layer *L*=144 and *L*=175, and for number of training samples *N*=145, *N*=172, and *N*=176.

Fig.2 presents the case when we have used *L*=144 neurons in the hidden layer and *N*=145 training samples. The straight line is the mean error for estimated DOA, 0.23°, when all neurons are in function. As we can see, as the percentage of failed neurons increases, the mean error for DOA is increasing from 0.234° (for 3% of failed neurons) to around 0.36° (for 30% of failed neurons), which corresponds to relative error increase from 1.7% to 56%.

Fig.3 presents the case when we have used *L*=144 neurons in the hidden layer and *N*=172 training samples. As is expected the mean error for DOA estimation when all neurons are in function is decreased and its value is 0.185°. The relative error increase is from 13.5% (for 3% of failed neurons) to 67% (for 30% of failed neurons).

Fig.4 presents the case when we have used *L*=175 neurons in the hidden layer and *N*=176. As is expected the mean error for DOA estimation when all neurons are in function is decreased and its value is 0.12°. The relative error increase is from 33% (for 3% of failed neurons) to 158% (for 30% of failed neurons, the error is more than doubled).

Observing the relative error increases we can easily conclude that the reliability of the network is much better when *L* and *M* are smaller. But the reason for that is that for larger *L* and *M* the mean error, when all neurons are in function, is much smaller. So, when we need very precise DOA estimation we should use larger values of *L* and *M*,

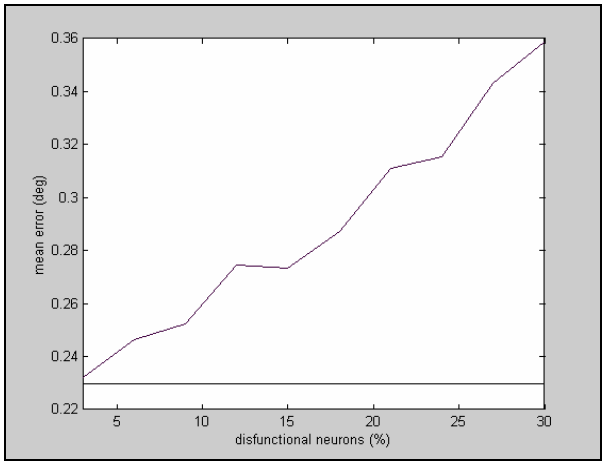

Fig.2 Mean error of DOA depending on the percentage of failed neurons for *L*=144 and *N*=145 (*SNR*=10 dB)

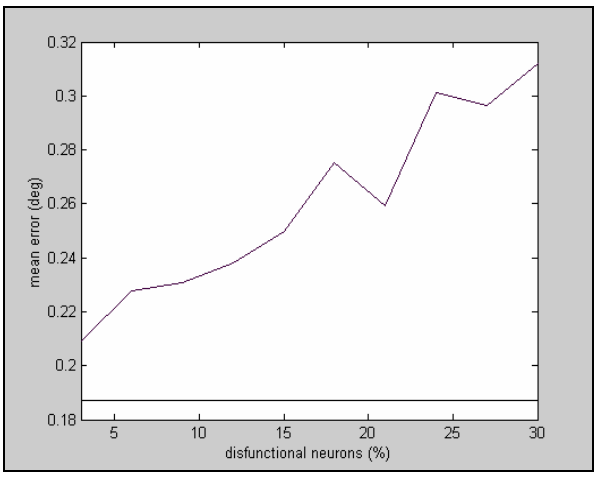

Fig.3 Mean error of DOA depending on the percentage of failed neurons for *L*=144 and *N*=172 (*SNR*=10 dB)

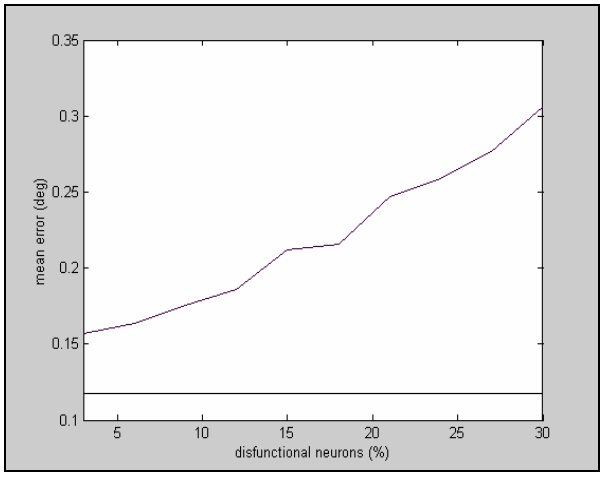

Fig.4 Mean error of DOA depending on the percentage of failed neurons for *L*=175 and *N*=176 (*SNR*=10 dB)

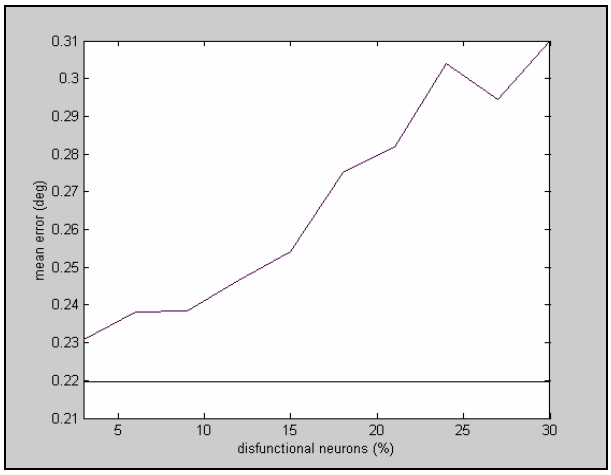

Fig.5 Mean error of DOA depending on the percentage of failed neurons for L=144 and N=145 (SNR=20 dB)

but we should use neurons with long time of well functioning.

As we have mentioned, we have used 50 trials for randomly chosen failed neurons. The smaller error for larger percentage of failed neurons (Fig.3) lead us to a conclusion that on the final mean error, except the percentage of failed neurons, there is an influence of the positions of the failed neurons in the hidden layer! This issue needs further deep analyze.

As another example we are going to present some results for larger value of *SNR*.

Fig.5 is presenting the results gained for the case when *SNR*=20 dB. We were using L=144 neurons in the hidden layer and the number of training samples was N=145. From the figure we can conclude that the relative error increase is from 5% (for 3% of failed neurons) to 41% (for 30% of failed neurons). This example and including many other results that we have analyzed, lead us to a conclusion that the value of *SNR* has not much influence on the mean error increases.

Also we should mention that for all cases we have used appropriate values for some parameters for RBFs, like for example the number of neighboring neurons taken into account while estimating the variances for RBFs, or appropriate mse of the NN, in order to gain comparable results.

4 Conclusion

We have presented the architecture of RBFNNs based smart antenna for DOA estimation problem in order to investigate the reliability of the hidden layer applied in the network.

Using computer simulation results we have showed that the networks with less neurons in the hidden layer are more reliable than those with larger number of neurons. But we have concluded that when we need more precise DOA estimations we should use more neurons in the hidden layer and more training samples. In this case we should be aware of the large performance degradation in the case if some percentage of the neurons fail.

References:

[1] R.O. Schmidt, "Multiple emitter location and signal parameter estimation", *IEEE Trans. Antennas Propagat.*, Vol.34, pp. 276-280, March 1986.

[2] A.H. El Zooghby, C.G. Christodoulou, and M. Georgiopoulos, "A neural network-based smart antenna for multiple source tracking", *IEEE Trans. Antenna Propagat*., Vol. 48, No.5, May 2000.

[3] A.H. El Zooghby, C.G. Christodoulou, and M. Georgiopoulos, "Performance of radial basis function networks for direction of arrival estimation with antenna arrays", *IEEE Trans. Antenna Propagat*., Vol. 45, pp. 1611-1617, Nov. 1997.

[4] M. Sarevska, B. Milovanović, and Z. Stanković, "Alternative Signal Detection For Neural Network-Based Smart Antenna", *IEEE Conference NEUREL'04*, Belgrade, Sept. 2004.

[5] Yu Hen Hu, and Jenq-Neng Hwang, "*Handbook of Neural Network Signal Processing*", CRC Press LLC, Boca Raton Florida, 2001.

[6] M. M. Gupta, L. Jin, and N. Homma, "*Static and Dynamic Neural Networks*", A John Wiley & Sons, Inc., Publication, 2003.

[7] Park, J., and I. W. Sandberg, "Universal Approximation Using Radial Basis Function Networks", *Neural Com*, Vol.3, 1991, pp. 246-257.

[8] Park, J., and I. W. Sandberg, "Approximation and Radial Basis Function Networks", *Neural Computation*, Vol. 5, 1993, pp. 305-316.

[9] T.J. Moody and C.J. Darken, "Fast learning in networks of locally tuned processing units", *Neural Computat*., Vol.1, pp. 281-294, 1989.

[10] J.T. Tou and R.C. Gonzales, *Pattern recognition Principles*. Reading, MA: Addison Wesley, 1976.

[11] M. Sarevska, B. Milovanović, and Z. Stanković, "Generalization Capabilities of Neural Network-Based Smart Antenna for DOA Estimation", *Conference TELFOR'04*, Belgrade, November 2004.

[12] C.G. Christodoulou, and M. Georgiopoulos, "*Applications of Neural networks in Electromagnetics*", Artech House, Inc., 2001