ARTIFICIAL NEURAL NETWORKS FOR MODELING WIRE ANTENNAS

Ing. Petr ŠMÍD, Doctoral Degree Programme (1)
Dept. of Radio Electronics, FEEC, BUT
E-mail: xsmidp01@stud.feec.vutbr.cz

Supervised by: Prof. Zbyněk Raida

ABSTRACT

The paper describes an approach to learning artificial neural networks (ANN) with the genetic algorithm (GA). The goal is modeling the wire dipole by ANN. The arm dipole length, frequency and input impedance are the training parameters for learning the ANN. Two types of ANN were selected for mentioned problem: the recurrent Elman ANN and the feed-forward one. Neural networks are implemented in MATLAB. Results of training abilities are discussed.

1 INTRODUCTION

Artificial neural networks (ANN) are electronic systems (software or hardware ones), which structure is similar to a human brain [2]. ANN consists of a set of neurons (Fig. 1-A), which are organized to layers [1]. Each neuron multiplies an input number *p* by a weight *w*, then adds a constant *b* to the product, and finally, the result is *processed* by a non-linear function.

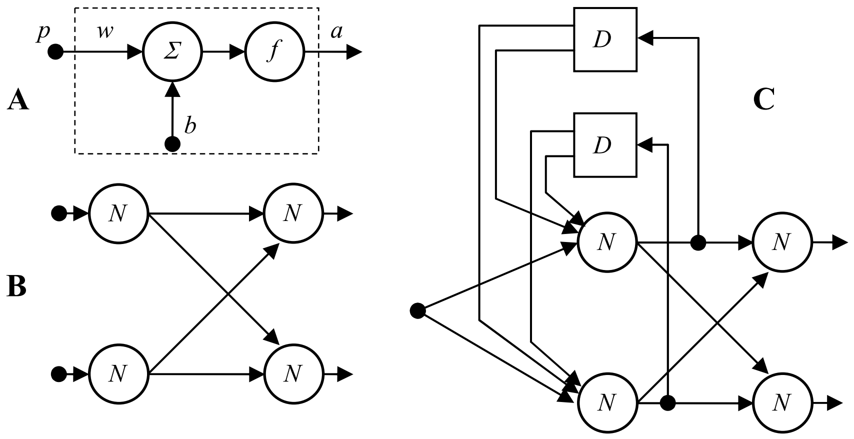


Fig. 1 *A – Neuron, B – Mutual connection of neurons in ANN, C – Elman ANN*

Layers of neurons are mutually connected in order to join output of all the neurons in a preceding layer and the inputs of all the neurons in the succeeding layer via weights *w* (Fig. 1-B). Weights between neurons are set during the training process. The trained ANN provides results very quickly, and therefore, it can be exploited to the solution of problems, which CPU-time demands are very high.

Architecture of a four-layer feed-forward ANN is depicted in Fig. 1-B. The network has two input parameters and two neurons in each layer. This type of ANN produces output signal corresponding with the input signal by multiplying, adding and processing by the neuron's output function described above.

Elman ANN differs from the feed-forward one in a feedback. The output of each neuron in the first hidden layer is connected back to the input of this layer via delay blocks *D*. Therefore this ANN produces an output signal, which depends (except the input signal) on the internal state of the network. Structure of the Elman network is shown in Fig. 1-C.

2 MODELING THE DIPOLE ANTENNA BY ANN

In order to obtain the learning patterns for the selected ANN, we use a model of a dipole antenna (Fig. 2). The input impedance *Z* of the dipole depends on the dipole arm length *l*, dipole wire radius *a* and the frequency *f*. We consider constant dipole wire radius for simplification. Having two input parameters *l* and *f* and two output parameters: dipole input resistance *R* and input reactance *X*, the feed-forward ANN has to consist of two input neurons and two output neurons. The number of the input neurons in Elman ANN is different: if this network should estimate the input dipole impedance on several frequencies, and the feedback abilities are going to be exploit for this task, the network has only one input *l*. For each input value of *l*, the Elman ANN iterates in several steps to be able to tell us several values of dipole input impedance. The output layer is the same as output layer of feed-forward ANN.

Selecting the number of hidden layers and neurons in them is the second problem. The first possibility is using the Bayesian regularization (implemented in the *Neural Network Toolbox* of MATLAB) and experimentally changing the number of hidden neurons until the number of effective parameters is in the prescribed interval [2]. In order to minimize the number of hidden neurons, we have to train an ANN consisting of as few hidden neurons as possible to manage satisfying training error (using the *Neural Network Toolbox* of MATLAB). If we know the number of the hidden layers and neurons in them, we can start training the ANN with the genetic algorithm.

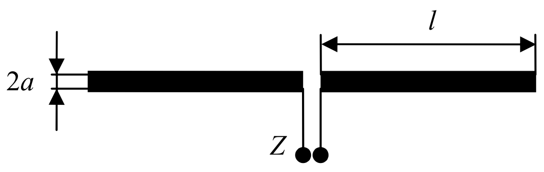


Fig. 2 *Model of dipole antenna*

3 TRAINING THE ANN WITH THE GENETIC ALGORITHM

The first step consists in creating the generation of chromosomes. Each chromosome consists of several genes [3]. Each gene (binary sequence) represents one numeric parameter of the optimized system. These parameters are weights and biases in our neural network. Learning a feed-forward ANN is described in detail in [3], [4]. For training the feed-forward ANN, nine training patterns (tab. 1) were used. These training patterns consist of two input numbers: value of the dipole arm length l and the frequency f and of two desired values R, X (real and imaginary part of antenna input impedance).

l	[m]	0,5	0,5	0,5	1,0	1,0	1,0	1,5	1,5	1,5
f	[MHz]	75	150	250	75	150	250	75	150	250
R	[Ω]	14,4	88,1	1219,6	71,0	1367,6	411,1	463,0	123,7	337,1
X	[Ω]	-616,4	38,7	959,4	41,6	-1482,5	513,0	890,4	120,2	472,9

Tab. 1 Training patterns – feed-forward ANN

If the Elman network has to be learned, the training algorithm has to be adjusted. It is necessary to put one training pattern containing the value of l at the first time step and the next time steps the training patterns contains instead of length values zeros. This way exploits only the internal state of the Elman ANN. An example of training patterns is concentrated in Tab. 2.

l	[m]	0,5	0,0	0,0	1,0	0,0	0,0	1,5	0,0	0,0
R	[Ω]	14,4	88,1	1219,6	71,0	1367,6	411,1	463,0	123,7	337,1
X	[Ω]	-616,4	38,7	959,4	41,6	-1482,5	513,0	890,4	120,2	472,9

Tab. 2 Training patterns – Elman ANN

4 RESULTS

In order to compare the learning abilities above two ANN, we set next conditions (the same parameters for both ANN): one hidden layer, number of chromosomes $N = 40$, number of bits per synoptic weight/biases $N_{bit} = 8$, probability of mutation $M = 10$ %, selection elite strategy [3], [4]. Every training process was run five times and the learning error was averaged.

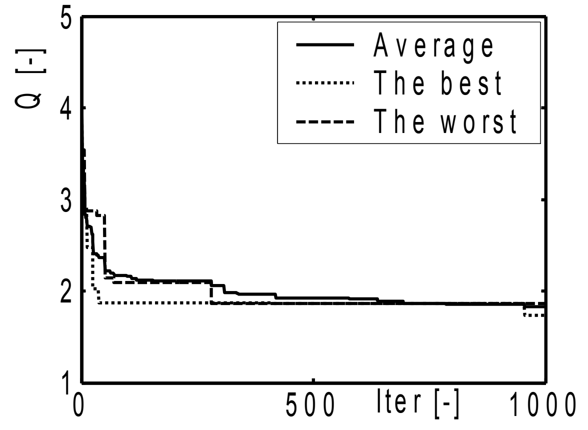
The results of training ANN with genetic algorithm are compared with convergence abilities if the feed-forward ANN with the same number of neurons in hidden layer is trained by gradient method (steepest-descent method) [4].

Tab. 3 concentrates the values of training error after 1000 iterations. In the first column, there is the lowest training error, average error of five training cycles is in the second column and the error of the worst training cycle is in the third column. Next two triplets of columns correspond to ANN consisting of the different number of neurons in the hidden layer.

Num. of neurons	5			8			10		
	Best	Average	Worst	Best	Average	Worst	Best	Average	Worst
Feed-forward ANN	2,58	2,63	2,69	2,57	2,68	2,73	2,59	2,70	2,84
Elman ANN	1,74	1,83	1,87	1,84	2,01	2,12	1,96	2,05	2,21
Gradient method	1,85	2,07	2,50	1,90	2,05	2,37	1,85	2,00	2,20

Tab. 3 Squared training error

Two charts in the next figure show the time behavior of the squared training error *Q* among the training patterns. In each chart, there are three patterns. The solid curve shows the average training error, the dashed curve represents the worst training error at the end of the training cycle and the dotted pattern denotes the best learning error (at the end of the training cycle).



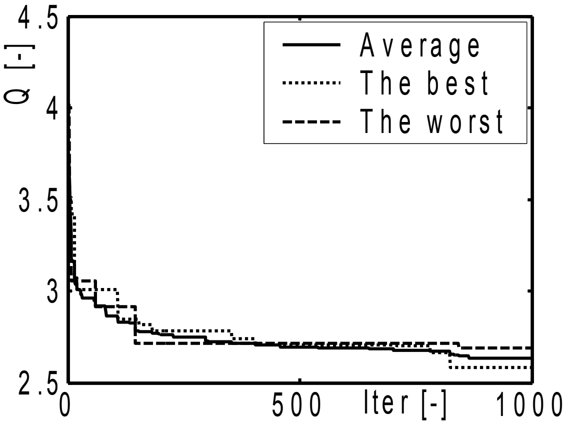


Fig. 3 *Training error of ANN with 5 neurons in hidden layer – 1000 iterations left: Elman ANN, right: feed-forward ANN*

Figure 4 shows the training error of the feed-forward ANN, when five neurons in the hidden layer are present, trained with gradient method.

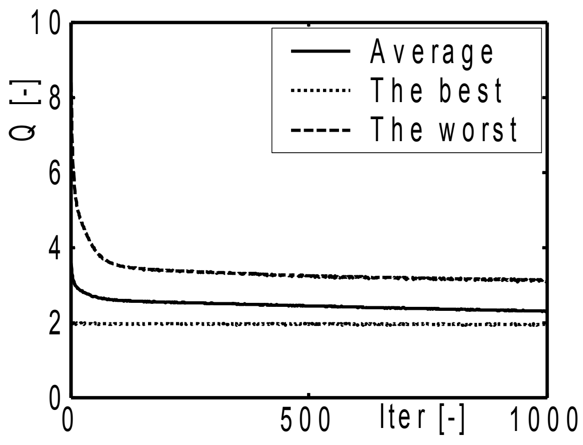


Fig. 4 *Training error – gradient training algorithm*

The result above was obtained in MATAB 6.1 on PC with AMD Athlon 2700+, 512 MB RAM, Windows XP. Computational times in seconds are given in Tab. 5 and Tab. 6.

Num. of neurons	5	8	10
Feed-forward ANN	31,6	31,7	47,2
Elman ANN	78,5	109,7	125,7
Gradient method	10,3	14,4	17,0

Tab. 4 Time demands

5 CONCLUSION

The learning abilities of genetic algorithm applied on selected ANNs (if they are trained to model the dipole antenna) are given in fig. 3. Comparing the patterns in above figure is evident that genetic learning algorithm gives better results if used for Eman ANN.

At the first look, the genetic algorithm used to learning the Elman ANN and the gradient method gives quite similar results (Fig. 3, Fig. 4). The Elman ANN trained by the genetic algorithm reaches little better learning abilities if the number of neurons in hidden layer is lower (Tab. 3). If ten neurons are used, the gradient algorithm gives the best results.

Comparing the values in the Tab. 5, we can see that the gradient algorithm has the lowest time demands.

ACKNOWLEDGEMENTS

Research described in the paper was financially supported by the research programs MSM 262200011 and MSM 262200022, by the grants of the Czech Grant Agency no. 102/03/H086 and 102/04/1079.

REFERENCES

[1] Demuth, H., Beale, M: Neural Network Toolbox For Use with MATLAB. User’s Guide (nnet.pdf in MATLAB help).

[2] Raida, Z: Neural Networks in Antennas and Microwaves: A Practical Approach. Radioengineering. 2001, vol. 10, no. 4, p.24 – 35. ISSN 1210-2512

[3] Černohorský, D., Raida, Z., Škvor, Z., Nováček, Z: Analýza a optimalizace mikrovlnných struktur. (Analysis and optimization of the microwave structures). Brno: VUTIUM Publishing, 1999. ISBN 80-214-1512-6.

[4] Raida, Z: Modeling EM Structures in the Neural Network Toolbox of MATLAB. IEEE Antennas & Propagation Magazine. 2002, vol. 44, no. 6, p.46 – 67. ISSN 1045-9243