Genetic Neural Networks for Modeling Dipole Antennas

Petr ŠMÍD, Zbyněk RAIDA, Zbyněk LUKEŠ

Dept. of Radio Engineering
Brno University of Technology
Purkyňova 118, CZ-61200 Brno
CZECH REPUBLIC

Abstract: The paper deals with original genetic neural networks for modeling wire dipole antennas. A novel approach to learning artificial neural networks (ANN) by genetic algorithms (GA) is described. The goal is to compare the learning abilities of neural antenna models trained by the GA and models trained by gradient algorithms. Developing the original design method based on genetic models of designed electromagnetic structures is the motivation of this work. Two types of ANN, the recurrent Elman ANN and the feed-forward one, are implemented in MATLAB. Results of training abilities are discussed.

Key-Words: artificial neural networks, genetic algorithm, wire dipole antenna.

1 Introduction

The design of electromagnetic (EM) structures is usually based on exploiting their numerical models. Numerical models request high computational power. Their evaluation has to be repeated many times in the optimization cycle: each update of the optimized parameters has to be followed by a new analysis. Replacing a numeric model of the designed EM structure by a neural one is one of ways to reduce CPU-time demands [1].

Artificial neural networks (ANN) are electronic systems (software or hardware ones) which structure is similar to a human brain. ANN consists of a set of neurons (Fig. 1-A).

Synaptic weights between neurons are set during the learning process [2].

Architecture of a two-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 statically maps the input patterns into output ones. The output response is formed by multiplying and summing input signals and by processing the result by a non-linear activation function.

The structure of Elman ANN is completed by a feedback compared to the feed-forward one. The output signal of each neuron in the first hidden layer is connected via delay blocks D to the inputs of this layer. Therefore, Elman ANN produces an output signal, which depends not only on the input signal but also on the internal state of the network. Elman ANN dynamically maps an input sequence into an output one. The structure of Elman ANN is shown in Fig. 1-C.

During the training process, ANN is learned to behave the same way as the numeric model of the antenna structure: considering dimensions and permittivities of the antenna structure as input parameters, the neural model provides...
input impedances, directivity patterns, gains and other parameters of the antenna as the output patterns. However, the training process takes more computational demands we can expect lower time demands, if the trained ANN is used instead of numerical model. Since summation, multiplication and evaluating non-linear activation function are the only three mathematical operations needed to compose the output, the modeling with learned ANN is very efficient and faster than numerical methods.

The training process is based on the minimization of the highly non-linear error function of the ANN. Exploiting standard gradient algorithm, the training is usually stopped in a local minimum located in the near distance from the starting point. Therefore, the training has to be repeated several times (considering random initial settings), and the best result has to be selected. If global optimization techniques (genetic algorithms in our case) are used for training, the learning process tends to converge to the global minimum of the error function, and the best neural model of the antenna structure is obtained that way.

The paper is focused in the development of novel broadband neural models of antenna structures (for simplicity, wire dipoles are selected for the demonstration). Neural models are based both on feed-forward ANN and on Elman ones. Both types of models are trained by genetic algorithms and by gradient ones as a reference. The results are in detail compared.

2 Neural model of the antenna

In order to obtain the learning patterns for the selected ANN, we use a numeric model of a dipole antenna (Fig. 2).

![Model of dipole antenna](image)

The input impedance $Z$ of the dipole depends on the dipole arm length $l$, the dipole wire radius $a$ and the frequency $f$. The dipole wire radius is considered constant for simplicity. Having two input parameters $l$ and $f$ and two output parameters $R$ (dipole input resistance) and $X$ (input reactance), the feed-forward ANN has to consist of two input neurons $(l, f)$ and two output ones $(R, X)$. Then, the input doublets $(l, f)$ are mapped into the output ones $(R, X)$.

The structure of Elman model is composed a different way: introducing the dipole arm length $l$ into the single input of the Elman ANN, the neural model produces a sequence of the input resistance $R(f_n)$ and reactance $X(f_n)$ related to the frequencies $f_n$. Hence, the output layer of the Elman ANN is identical with the output layer of the feed-forward ANN (Fig. 1-B, C).

Selecting the proper number of hidden layers and their neurons is the second problem. For solving, Bayesian regularization can be used in order to estimate the number of effectively used parameters (the number of layers and neurons is changed to reach the prescribed interval [2]). Then, training the ANN by the genetic algorithm can be started.

3 Genetic training algorithm

The first step consists in creating the generation of chromosomes. Each chromosome consists of several genes [3], [7]. Each gene (binary sequence) represents one numeric parameter of the optimized system. These parameters are weights and biases of the neural network. For training the feed-forward ANN to behave as a wire dipole, nine training patterns (all the combinations of $l = [0.5 \text{ m}, 1.0 \text{ m}, 1.5 \text{ m}]$ and $f = [75 \text{ MHz}, 150 \text{ MHz}, 250 \text{ MHz}]$) were used.

In case of training Elman ANN to behave as a wire dipole, we composed a set of three input sequences, which contain encoded information about the dipole length $l = [0.5 \text{ m}, 1.0 \text{ m}, 1.5 \text{ m}]$. Elman ANN was asked to react on an input sequence by producing a sequence of doublets $R(f_n)$ and $X(f_n)$ where $f = [75 \text{ MHz}, 150 \text{ MHz}, 250 \text{ MHz}]$.

4 Results

In order to compare the learning abilities of the above two ANN, we set the following conditions (the same parameters for both ANN): one hidden layer, the number of chromosomes $N = 40$, the number of bits per
synaptic weight/bias $N_{\text{bit}} = 8$, probability of mutation $M = 10\%$, selection elite strategy [3], [4], [7] and length of one training process 250 iterations. Due to random initial weights and biases in ANN (and due to stochastic base of genetic algorithms), each training process was run five times and the learning error was averaged. The results of training ANN by a genetic algorithm are compared to convergence abilities of gradient training (steepest-descent method [4], [5] and Levenberg-Marquardt method) [6]. Tab. 1 concentrates the learning abilities.

Tab. 1. Squared training error

<table>
<thead>
<tr>
<th>Num. of neurons</th>
<th>ANN / Method</th>
<th>Worst</th>
<th>Average</th>
<th>Best</th>
<th>Time [s]</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Elman / GA</td>
<td>0.284</td>
<td>0.240</td>
<td>0.200</td>
<td>12.04</td>
</tr>
<tr>
<td></td>
<td>Feed-forw. / GA</td>
<td>0.150</td>
<td>0.154</td>
<td>0.150</td>
<td>4.13</td>
</tr>
<tr>
<td></td>
<td>Elman / SD</td>
<td>0.061</td>
<td>0.044</td>
<td>0.020</td>
<td>0.59</td>
</tr>
<tr>
<td></td>
<td>Feed-forw. / SD</td>
<td>0.292</td>
<td>0.196</td>
<td>0.167</td>
<td>0.55</td>
</tr>
<tr>
<td></td>
<td>Elman / LM</td>
<td>0.854</td>
<td>0.422</td>
<td>0.273</td>
<td>31.48</td>
</tr>
<tr>
<td></td>
<td>Feed-forw. / LM</td>
<td>0.163</td>
<td>0.160</td>
<td>0.154</td>
<td>23.11</td>
</tr>
<tr>
<td>3</td>
<td>Elman / GA</td>
<td>0.229</td>
<td>0.214</td>
<td>0.187</td>
<td>12.96</td>
</tr>
<tr>
<td></td>
<td>Feed-forw. / GA</td>
<td>0.159</td>
<td>0.154</td>
<td>0.148</td>
<td>4.55</td>
</tr>
<tr>
<td></td>
<td>Elman / SD</td>
<td>0.221</td>
<td>0.135</td>
<td>0.033</td>
<td>0.59</td>
</tr>
<tr>
<td></td>
<td>Feed-forw. / SD</td>
<td>0.444</td>
<td>0.226</td>
<td>0.163</td>
<td>0.55</td>
</tr>
<tr>
<td></td>
<td>Elman / LM</td>
<td>0.641</td>
<td>0.402</td>
<td>0.325</td>
<td>74.93</td>
</tr>
<tr>
<td></td>
<td>Feed-forw. / LM</td>
<td>0.171</td>
<td>0.163</td>
<td>0.156</td>
<td>37.21</td>
</tr>
<tr>
<td>4</td>
<td>Elman / GA</td>
<td>0.236</td>
<td>0.215</td>
<td>0.199</td>
<td>13.94</td>
</tr>
<tr>
<td></td>
<td>Feed-forw. / GA</td>
<td>0.156</td>
<td>0.154</td>
<td>0.151</td>
<td>4.94</td>
</tr>
<tr>
<td></td>
<td>Elman / SD</td>
<td>0.316</td>
<td>0.140</td>
<td>0.025</td>
<td>0.61</td>
</tr>
<tr>
<td></td>
<td>Feed-forw. / SD</td>
<td>0.520</td>
<td>0.296</td>
<td>0.183</td>
<td>0.57</td>
</tr>
<tr>
<td></td>
<td>Elman / LM</td>
<td>1.192</td>
<td>0.610</td>
<td>0.318</td>
<td>151.91</td>
</tr>
<tr>
<td></td>
<td>Feed-forw. / LM</td>
<td>0.153</td>
<td>0.150</td>
<td>0.143</td>
<td>56.05</td>
</tr>
<tr>
<td>5</td>
<td>Elman / GA</td>
<td>0.224</td>
<td>0.218</td>
<td>0.205</td>
<td>14.59</td>
</tr>
<tr>
<td></td>
<td>Feed-forw. / GA</td>
<td>0.164</td>
<td>0.156</td>
<td>0.152</td>
<td>5.29</td>
</tr>
<tr>
<td></td>
<td>Elman / SD</td>
<td>0.372</td>
<td>0.332</td>
<td>0.262</td>
<td>0.60</td>
</tr>
<tr>
<td></td>
<td>Feed-forw. / SD</td>
<td>0.827</td>
<td>0.421</td>
<td>0.189</td>
<td>0.56</td>
</tr>
<tr>
<td></td>
<td>Elman / LM</td>
<td>1.469</td>
<td>0.690</td>
<td>0.263</td>
<td>297.87</td>
</tr>
<tr>
<td></td>
<td>Feed-forw. / LM</td>
<td>0.190</td>
<td>0.166</td>
<td>0.146</td>
<td>79.80</td>
</tr>
</tbody>
</table>

The table is divided horizontally in four parts. Each part corresponds to one of four ANN which differ in the number of neurons in the hidden layer written in the first column. The type of selected ANN (Elman or the feed-forward one) and the method which was used to train this ANN is in the second column, where GA means the genetic algorithm, SD means the steepest descent method and LM denotes Levenberg-Marquardt method.

Except the average mean square error values (in Average column), two others columns are present: Worst designates columns containing the highest error values after 250th iteration and Best specifies the column which contains the lowest error values of trained ANN.

Computational times in seconds are given in the last column. Each number is the average time over the 250-iteration cycle.

Graphical representation of the learning progress of the Elman ANN trained with the GA is depicted in Fig. 3.

Fig. 3. Squared learning error of Elman ANN; genetic algorithm

The chart shows the time course of the squared training error $e$ over 250 iterations. In the chart, there are three curves: squared error of the best ANN (dotted line), squared error of the worst ANN (dashed line), and the average squared error computed over five realizations (solid line). The best ANN and the worst ANN are chosen according to the value learning error in the 250th iteration. The end-points of all three curves corresponds to the first three values in the “Elman / GA” line in the bottom (five neurons in hidden layer) part in Tab. 1.

Similarly, the Fig. 4 shows the time learning ability of the feed-forward ANN trained with the GA.

Fig. 4. Squared learning error of feed-forward ANN; genetic algorithm
In similar way, Fig. 5 and Fig. 6 describe the learning ability of Elman ANN and feed-forward one respectively, when the steepest descent method used.

![Squared learning error of Elman ANN; steepest descent](image1)

**Fig. 5.** Squared learning error of Elman ANN; steepest descent

![Squared learning error of feed-forward ANN; steepest descent](image2)

**Fig. 6.** Squared learning error of feed-forward ANN; steepest descent

Finally, the two last figures show the learning ability of both types of ANN trained using Levenberg-Marquardt method.

![Squared learning error of Elman ANN; Levenberg-Marquardt method](image3)

**Fig. 7.** Squared learning error of Elman ANN; Levenberg-Marquardt method

![Squared learning error of feed-forward ANN; Levenberg-Marquardt method](image4)

**Fig. 8.** Squared learning error of feed-forward ANN; Levenberg-Marquardt method

All learning methods were implemented in MATLAB. The Levenberg-Marquardt method was used without backpropagation algorithm [5] for simplification. The results above were obtained in MATLAB 6.1 on a PC equipped by AMD Athlon 2700+, 512 MB RAM. Windows XP.

5 Conclusion

At the first look, the feed-forward ANN learned with the genetic algorithm and the gradient method give quite similar results if they are trained to model the dipole antenna. The feed-forward ANN with the genetic algorithm reaches better learning abilities than the Elman ANN. The Elman ANN learned by the steepest descent method gives the best result especially if the number of neurons is low (Fig. 3, Fig. 4). If the number of neurons is high (a typical situation in practical realizations), both types of ANN trained by the genetic algorithm exhibit better learning abilities than ANN learned by the steepest descent method. The Levenberg-Marquardt method (without the backpropagation) used for the feed-forward ANN gives similar results as the genetic algorithm.

Comparing the values of average time, the gradient algorithm can be seen to have the lowest time demands. The genetic algorithm spends more time against low computational demands. The highest time demands are exhibited by genetic learning of Elman ANN caused by necessity of an additional cycle for evaluating the internal state of this recurrent ANN, especially if Levenberg-Marquardt algorithm was used. That is caused by the fact, that complicated symbolic equations have to be
evaluated instead of the backpropagation algorithm.

Nowadays, we are strongly interested in: the improvement of learning the ANN with GA to obtain the ANN outputs more accurately. The second area of our current interest is the computational demands dependency on the size of training set.

After these two tasks, a system utilizing the ANN instead of a numerical model can be developed and compared with conventional methods (easier designed structures) and tested, to design structures for which no exact formula exists.

Acknowledgements:

Research described in this paper was financially supported by the Grant Agency of the Czech Republic under grants no. 102/04/1079 and 102/03/H086.

Further financial support was obtained from the Czech Ministry of Education under the research programs MSM 262200011 and MSM 262200022.

References:


