STRUCTURE OPTIMIZATION OF NEURAL NETWORK FOR TIME DOMAIN SIGNAL PREDICTION

J. Svobodová, Z. Lukeš

Ústav radioelektroniky, FEKT VUT v Brně

Purkyňova 118, 612 00 Brno

Abstract

Finding appropriate structure of neural network for the problem being solved is one of main problems of neural network design. It can be done empirically or some additional algorithms can be used. The aim of this paper is to compare selected evolutionary algorithms: genetic algorithms, evolutionary strategies and evolutionary programming. Each of these methods is used to design the structure (especially number of hidden layers and number of neurons in these layers) of feedforward backpropagation neural network and tested on one of the typical problems: signal prediction. Results, graphs of network learning and fitness function progress from Matlab environment and calculation limitations are discussed in the paper.

1. Introduction

The prediction of the signal can be found useful for example in medicine, when it is suitable to predict the progress of EEG or EMG signal. The main problem of the prediction is the long computational time, which is affected by the selected method, the signal complexity and number of signal samples. The methods described in this paper combine stochastic and gradient methods avoiding getting stuck in a local minimum, which is the main disadvantage of using only the gradient methods. Aim of this paper is to compare the selected evolutionary algorithms: genetic algorithms, evolutionary strategies and evolutionary programming. Each of these metods is used to generate the structure of the feedforward backpropagation neural network, which is suitable for predicting the signal.

2. The backpropagation neural network learning algorithm

The training set of the neural network is designated as a matrix of inputs (four subsequent samples of a signal or a function) and a vector of outputs (the next sample of a signal or a function following four appropriate samples). The functions of the neurons are the hyperbolic tangens with random initial values of steepnesses and the weights of the connections and biases are initially set random, as well. The backpropagation learning algorithm follows three main steps:

1. feedforward (the inputs are applied to the network and the real output is calculated according to functions of the network, weights and biases):

$$y_k = f\left(w_{0k} + \sum_{j=1}^{p} z_j w_{jk}\right), \qquad (1)$$

where y_k is the output signal value, w_{0k} is kth bias of the output layer neuron, p is the number of layers and w_{jk} are weights between neurons.

2. backpropagation:

$$\delta_k = (t_k - y_k)\, f'\left(w_{0k} + \sum_{j=1}^{p} x_j w_{jk}\right), \qquad (2)$$

where t_k is the target output for current input, y_k is the output computed by neural network, w_{0k} is the bias value, x_j is the current input and w_{jk} is the matrix of weights.

$$\Delta w_{jk} = \alpha \delta_k f\left(w_{0j} + \sum_{i=1}^{n} x_i w_{ij}\right) \tag{3}$$

3. the update of weights and biases:

$$w_{jk}(n+1) = w_{jk}(n) + \Delta w_{jk}. \tag{4}$$

3. Selected Evolutionary Algorithms

The number of layers, numbers of neurons in the layers, initial weights and initial biases are determined by the selected evolutionary algorithms. These methods differ in using genetic operators: genetic algorithms use selection, crossover and mutation, evolutionary strategies and evolutionary programming use only selection and mutation.

Processes of selected methods in general are the following:

Genetic algorithms:
1. creating initial random population of possible solutions (individuals)
2. evaluating fitness of each individual
3. testing the fitness of the best individual
4. generating a population of the fittest solutions, making random pairs of these solutions
5. crossover, mutation

Evolutionary strategies:
1. creating initial random population of possible solutions (individuals)
2. mutation producing the offspring for each one of these individuals
3. calculating the fitness of each member of initial population and each offspring
4. comparing each parent with his offspring, the better one is kept for next evaluation

Evolutionary programming:
1. creating initial random population of possible solutions (individuals)
2. producing the offspring for each individual by mutation
3. calculating the fitness of each parent and his offspring, keeping the best half of population

Each of these algorithms uses partially trained networks (with the same number of training epochs) as the initial population. The fitness of each individual is computed by means of each error of the network, then the selected algorithm is proceeded and the best individual determines the suitable network.

4. Results

The resulting network has the number of layers, numbers of neurons, steepnesses of the transfer functions of the neurons and initial weights and biases determined by selected evolutionary algorithm and it is partially learned. The learning process continues until the desired error is reached. The progress of the error function for evolutionary strategies is showed in the Figure 1. The mimimum error is the error of currently best network and it decreases only if better network is created by mutation, in other cases it remains stable. The progress of minimum error for evolutionary programming and genetic algorithm looks similar, these algorithms differ in the principles of selection and crossover, so there is a difference in the mean error. The error values shown in the Figure 1 also depend on the number of learning iterations of neural network, the more iterations are set, the smaller the errors are.

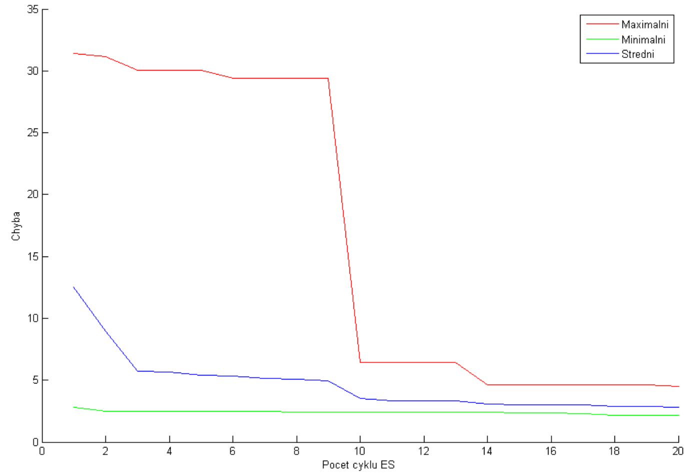

Figure 1: The progress of network error for evolutionary strategies

The outputs of the resulting neural network (red points) compared with original function (blue line) are plotted in the Figure 2. These can fit better, if the network is better trained (by more training iterations or with some of advanced training methods, both of these processes cause significant increasing of computation time).

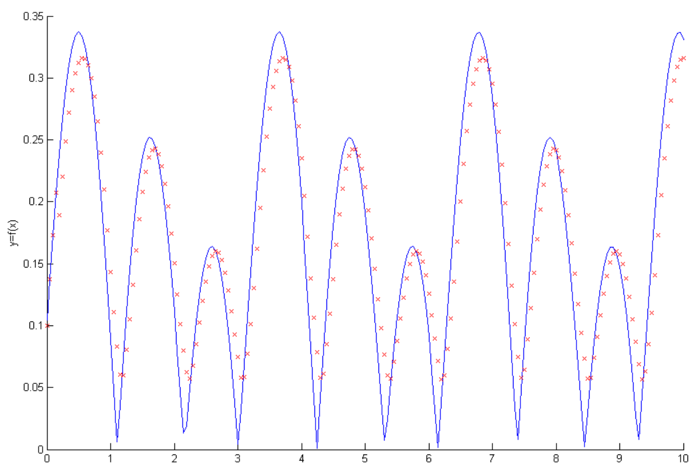

Figure 2: Prediction of the original function values by neural network

5. Conclusion

All the selected algorithms worked reliably, the network indicated as the best was always suitable for solving the problem. The main difference between them is in the computational time, which depends on the total number of computational steps. The most suitable algorithm for the selected problem is evolutionary strategies, which gives acceptable results and, in addition, it works relatively fast. However, all of the selected algorithms plenty of computational time to find appropriate structure of the neural network and the network then has to be trained to be able to predict the function or the signal satisfactorily.

References

[1] MAŘÍK V. a kol.: Umělá inteligence (4), ACADEMIA, Praha, 2003, ISBN: 80-200-1044-0.

[2] UMĚLÉ NEURONOVÉ SÍTĚ – ZÁKLADY TEORIE A APLIKACE (3) [online]. 2006 [cit. 2008-05-07]. Accessible from: < http://www.chemagazin.cz/Texty/CHXVI_1_cl3.pdf>.

[3] VOLNÁ, E. : Neuronové sítě a genetické algoritmy, Učební texty Ostravské univerzity, Ostrava 1998, ISBN 80-7042-762-0

[4] Freeman, J. : Neural Networks, Algorithms, Applications, and Programming Techniques, Addison-Wesley Publishing Company, U. S. A., 1991, ISBN: 0-201-51376-5.

[5] Haykin, S.: Neural Networks: A Comprehensive Foundation, Pearson Education, Singapore, second edition, 2001, ISBN: 81-7808-300-0

[6] Artificial Neural Networks and Evolutionary Algorithms [online]. 2009 [cit. 2009-03-16]. Accessible from www: < http://www.rgu.ac.uk/eng/compint/page.cfm?pge=28907>.

[7] Principles of training multi-layer neural network using backpropagation [online]. 2009 [cit. 2009-03-28]. Accessible from www: <http://home.agh.edu.pl/~vlsi/AI/backp_t_en/backprop.html>.

[8] Předpovídání pomocí neuronových sítí [online]. 2009 [cit. 2009-05-22]. Accessible from: http://www.obitko.com/tutorials/predpovidani-neuronovou-siti/uvod.html>.

Ing. Jitka Svobodová, e-mail: xsvobo61@stud.feec.vutbr.cz
Ing. Zbyněk Lukeš, Ph.D., e-mail: lukes@feec.vutbr.cz