Optimization of Input Parameters for the Neural Network Used in Evaluation of Bio-Signals

Ing. Martina Mironovová

Vedoucí práce: Prof. Ing. Jiří Bíla, DrSc., Ing. Ivo Bukovský, PhD.

Abstrakt

Účelem této práce je nalézt optimální vstupní parametry pro neuronovou síť tak, aby bylo dosaženo řešení v co nejkratším čase při adaptaci spojité, dynamické kvadratické neuronové jednotky s dopravním zpožděním pro vyhodnocování biologických signálů. Možné použití genetických algoritmů je uvažováno a testováno na daném signálu.

Abstract

The purpose of this work is to find optimal input values for neural network in order to achieve faster solution in adaptation process of a continuous Time-Delayed Dynamic Quadratic Neural Unit used for evaluation of bio-signals. A use of genetic algorithm is studied and tested on a certain waveform.

Klíčová slova

Neuronová síť, biologický signal, EKG, genetický algoritmus, optimalizace, váhy, dopravní zpoždění, TmD-DQNU

Keywords

Neural network, bio-signal, ECG, genetic algorithm, optimization, weights, time-delay, TmD-DQNU

1. Introduction

Basic concepts of artificial neural networks were laid in forties of twentieth century. The then researchers draw an inspiration from neural structures of living organisms, predominantly of a human brain, that consists of more than a hundred billion neurons – basic building units with ongoing data processing.

A main motivation for the work is a creation of an artificial neural system for evaluation and prediction of bio-signals, namely a non-conventional Time-Delayed Dynamic Quadratic Neural Unit (TmD-DQNU). Concept of this work assumes that by studying a behavior of characteristic parameters of a TmD-DQNU during the adaptation process can reveal and predict unwanted states in bio-signals, i.e. detection and prediction of arrhythmias in ECG signal.

1.1 Adaptation of TmD-DQNU

Biological neurons process and transmit information by so called synapses that connect individual neurons between each other and then create complex structures [1]. Very fundamental sketch of a continuous, time-delayed quadratic artificial neural unit TmD-DQNU is displayed in *Figure 1*.

This mathematical model described in [7] poses better approximation capabilities and can be expressed by a following equation:

$$\begin{aligned} y_n(t) &= \int \left(\mathbf{x}(t)^{\mathrm{T}} \mathbf{W} \mathbf{x}(t) \right) dt \\ &= \int \left(w_{00} + w_{01} u(t) + w_{02} y(t - T_d) + w_{11} u^2(t) + w_{12} u(t) y(t - T_d) + w_{22} y^2(t - T_d) \right) dt \end{aligned} \tag{1}$$

In equation (1) a column vector $\mathbf{x}(t) = \begin{bmatrix} 1 \\ u(t) \\ y(t - T_d) \end{bmatrix}$ represents neural inputs and a triangular matrix $\mathbf{W} = \begin{bmatrix} w_{00} & w_{01} & w_{02} \\ 0 & w_{11} & w_{12} \\ 0 & 0 & w_{22} \end{bmatrix}$ displays weights that are adapted in each step of the algorithm.

Adaptation process is based on RTRL method of a back propagation (Real Time Recurrent Learning) [7], [8], [9]. Weights and a time-delay can be updated according to following formulae:

$$w_{ij} = w_{ij} + \Delta w_{ij} \tag{2}$$

$$T_d = T_d + \Delta T_d \tag{3}$$

Equation for a weight increment and time-delay increments, respectively, can be expressed using formula (4) and (5) [7].

$$\Delta w_{ij} = \mu e(t) \frac{\partial y_n(t)}{\partial w_{ij}} = \mu e(t) \int \frac{\partial}{\partial w_{ij}} \left(\mathbf{x}^{\mathrm{T}} \mathbf{W} \mathbf{x} \right) dt = \mu e(t) \int \left(\frac{\partial \mathbf{x}^{\mathrm{T}}}{\partial w_{ij}} \mathbf{W} \mathbf{x} + \mathbf{x}^{\mathrm{T}} \frac{\partial \mathbf{W}}{\partial w_{ij}} \mathbf{x} + \mathbf{x}^{\mathrm{T}} \mathbf{W} \frac{\partial \mathbf{x}}{\partial w_{ij}} \right) dt \tag{4}$$

$$\Delta T_d = \mu e(t) \int \left(- w_{02} \dot{y}_n(t - T_d) - w_{12} u(t) \dot{y}_n(t - T_d) - w_{22} \cdot 2 \cdot y_n(t - T_d) \dot{y}_n(t - T_d) \right) dt \tag{5}$$

The algorithm adapts weights and time-delay in a way that error between real values and values coming from the neural network is minimized:

$$e = y_{real} - y_{neural} \quad \rightarrow \quad 0 \tag{6}$$

Due to the RTRL technique and a convex nature of TmD-DQNU neural system is able to converge to a solution faster than other conventional neural models and does not get stacked in some less accurate local minima [10].

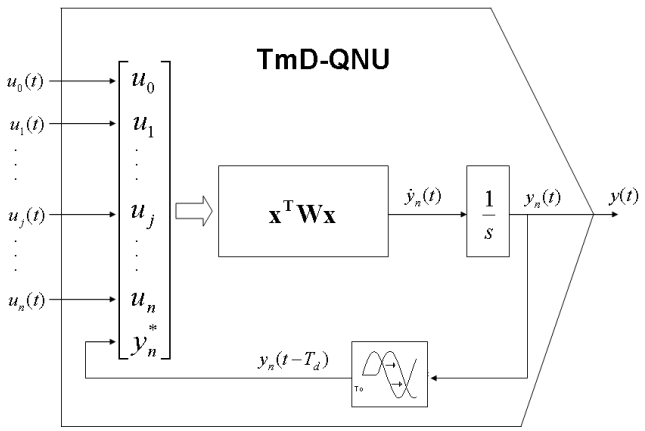

Figure 1. Schematics of continuous-time TmD-DQNU; a possible representation of a biological neuron by a quadratic non-linear higher-order mathematical model presented in works [2], [3], [4] and in [5] and [6].

However, in adaptation to a complex signal, such as ECG signal, it is not straightforward which initial values of the weight matrix and a time-delay should be set to start the adaptation. Different initial values of weights and time-delay can influence adaptation process significantly and this leads to different time periods to find the correct solution. For example, a test data in a basic shape of ECG signal were loaded as y_{real} and TmD-DQNU was set to adapt to the signal.

To explain the initial weight values and a time-delay problem, the adaptation ran under three different conditions. Input values for each condition is summarized in *Table 1*.

Table 1. - Input data for three cases of adaptation to one pattern (y_{real})

	Case 1	**Case 2**	**Case 3**
w_{00}	0	0.3	0.2
w_{01}	0	0	0.05
w_{02}	-0.4	-0.4	-0.05
w_{11}	0	0	0.15
w_{12}	-0.2	-0.2	-0.15
w_{22}	-0.2	-0.2	-0.05
T_d	0.05	0.05	1.05

A progress of adaptation is also shown graphically in *Figures 2 – 4*. Blue waveform represents y_{real} data that are periodic and purple curve displays the output of the neural network. Eventually, all three cases shown will converge to the correct solution and the error between y_{real} and y_{neural} signals will be minimal. However, each case with a different number of steps of adaptation, or with different time intervals. Plots show only the first 30,000 steps of adaptation and it is visible, that different initial values of weights and time-delay influence the adaptation process.

The question that arose in this phase of research was simple. Is there a possibility to find or estimate such input values of weights and time-delay, so that the neural network would converge as fast as possible? An option is a genetic algorithm – a heuristic method used in optimization tasks [11].

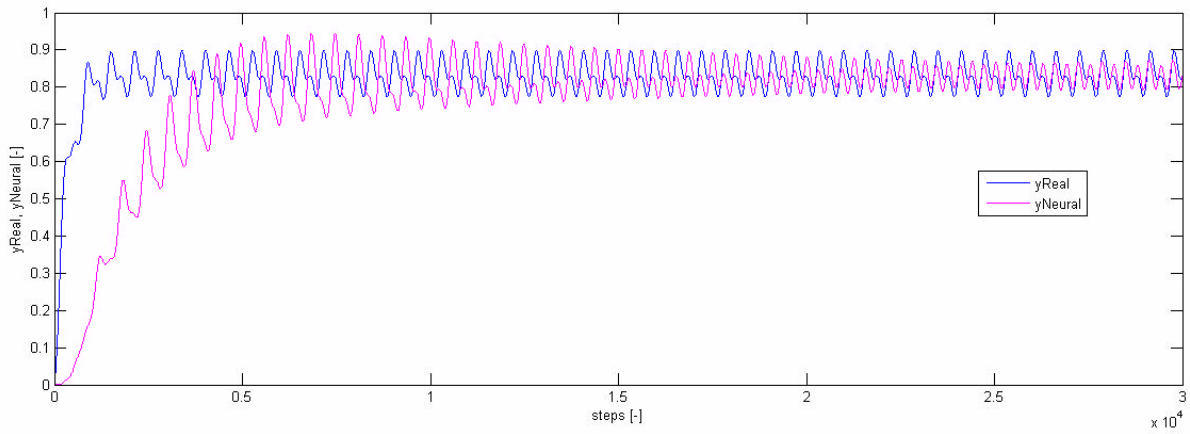

Figure 2. Comparison of y_{real} and y_{neural} signals during the adaptation of weights and time-delay Case 1

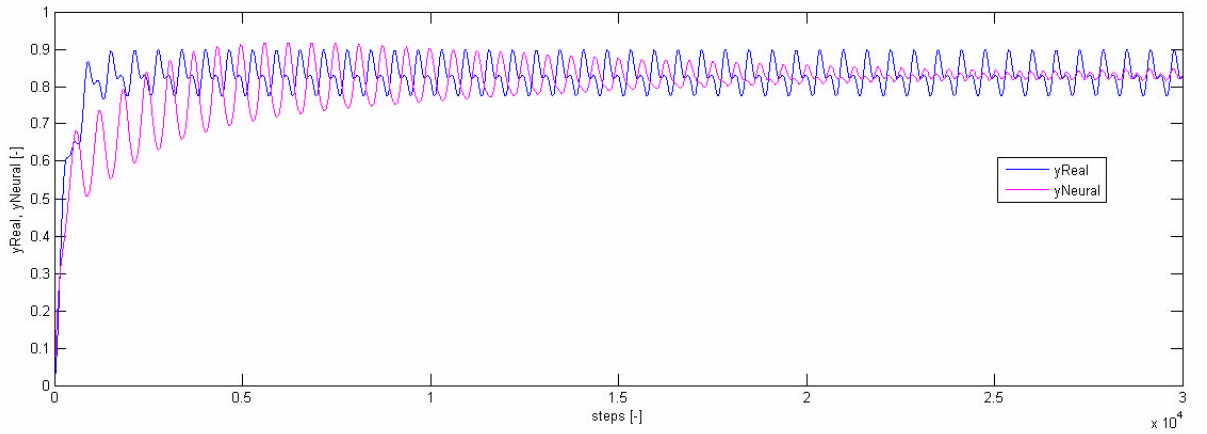

Figure 3. Comparison of y_{real} and y_{neural} signals during the adaptation of weights and time-delay Case 2

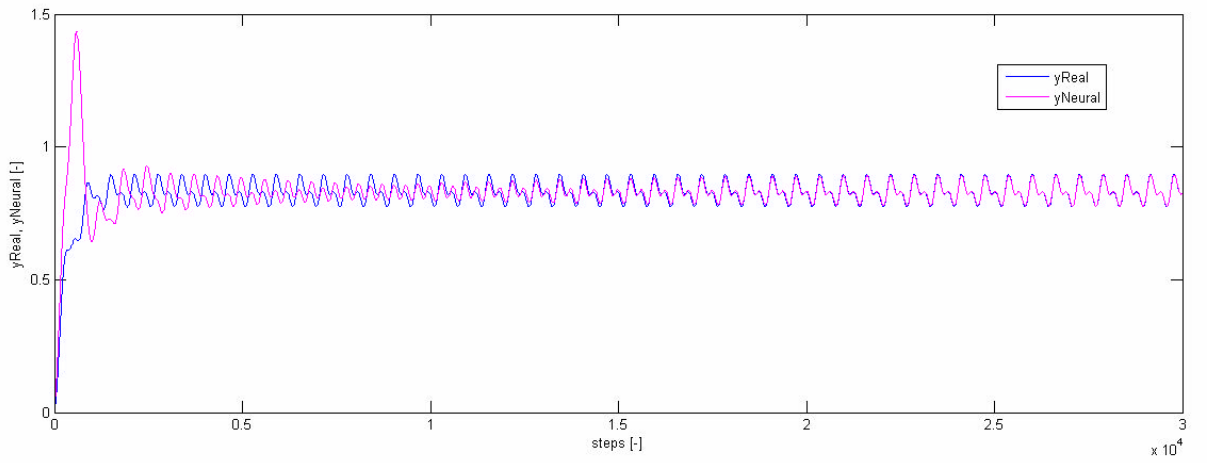

Figure 4. Comparison of y_{real} and y_{neural} signals during the adaptation of weights and time-delay Case 3

2 Main Characteristics of a Genetic Algorithm

Genetic algorithm is a heuristic method inspired by Darwin's theory of descent [11]. It uses a group of solutions between which it selects the best ones. Selected solutions are combined between each other (so called crossover process) that lead to newly incurred solutions. The week solutions (worst ones) are being eliminated. In order to prevent the group of solutions to be restrained just to combination of already existing ones, the mutation of selected specimens occurs in every step of the algorithm with known probability. The effort is to enhance the population in every step of the algorithm so that it leads to the retrieval of optimal solution, or solution close to the optimal one [12].

2.1 Principle of Finding Optimal Input Parameters

Input values of weights and time-delay of a concrete solution are for the purpose of algorithm coded into so called genome, or chromosome. Such a chromosome contains of genes and each gene represents one combination of input parameters. At the beginning of the algorithm input data are selected randomly for each specimen. Together, data create a population matrix. The population matrix goes through four phases in the algorithm, similarly as described in work [12].

In order to estimate the solution quality, the so called fitness function can be used. This function can take arbitrary form and in this example it is represented as an error between y_{real} and y_{neural} after specified iterations of the neural network. The aim is to minimize this function.

2.2 Results from Genetic Algorithm

During the estimation of optimal input values of the neural network by use of a genetic algorithm, several problems occurred. Firstly, because the fitness function represents an error between real and neural data, the part of calculation is actually run of the neural network itself in order to see, how weights and time-delay influence the adaptation process of the neuron. Some values of weights or time-delay lead the whole network to instability, causing the output from the integrator block in the neural model (in Matlab) to give out infinite values or not a number (NaN) value. Such a situation leads to the halt of calculation in the neural model. Though infinite values in the integrator block can be restrained to a boundary values, NaN cannot be replaced. For such situations, the whole calculation including the genetic algorithm process is stopped.

In case that the random values for weights and time-delay in genetic algorithms do not lead to the unstable system, the calculation proceeds. However, in a genetic algorithm a fitness function (or a calculation of the error) has to run multiple times. In the first part of the genetic algorithm it runs m-times, for each population (i.e. $m = 10$, the size of the population). In the second part of the process, it runs in reproduction and mutation phase ($l = 2$). Together, mutation and reproduction phase is done k-times, for k number of iterations. From experiments and empiric data, number of iterations for an efficient genetic algorithm should be at least 100.

From the data above, total number of steps taken in the neural network during the genetic algorithm calculation can be estimated if a neural network calculation is taken on 30,000 steps ($s = 30{,}000$) as follows:

- in phase 1: $m \times s = 10 \times 30{,}000 = 300{,}000$ steps
- in phase 2: $k \times l \times s = 100 \times 2 \times 30{,}000 = 6{,}000{,}000$ steps

From the calculation above a total steps in neural network taken during the calculation of a genetic algorithm come to 6,300,000 steps.

From empiric data taken so far, one neural network can adapt to some acceptable value (with some certain error low enough) in 300,000 ÷ 500,000 steps. Obviously, this value is much lower than value obtained from the genetic algorithm and thus it is faster to select random input data to the neural network and run the adaptation for more steps (300,000 ÷ 500,000 steps to the waveform y_{real} displayed in Chapter 1.1).

3. Discussion

The main purpose of the work was to try to find optimal input weights and time-delay values for TmD-DQNU in order to speed up the adaptation process and to converge to a solution faster. To find optimal input values, a genetic algorithm was selected. However, some applications find this heuristic algorithm useful, this particular example proved that evaluation of possible solutions (input data) takes too many steps that prolong the calculation times. This is the main reason, why this method is not efficient.

Another problem that occurred with combination of neural network itself and genetic algorithm was fact that some input data make neural network unstable. This situation leads to instability of the system and stops the whole calculation.

4. Further Work

A possibility how to solve the problem of instability of the neural system described above can be observed and studied in the software, where simulations are processed (Matlab). Possible ways how to remove NaN values coming from the integrators may exist and should be investigated. A possibility how to speed up the genetic algorithm may be in number of steps taken in neural network – number of steps taken should be lowered and an output should be tested.

The problem of optimal input values may also be solved by a method of wavelet transform. With this method, parameters of y_{real} waveform, such as frequencies present in the signal, can be found. Knowledge of these parameters may help in selection of input data to the neural network.

List of Used Symbols

y_{real}	output – real waveform	[-]
y_{neural}	output from the neural network	[-]
e	error ($y_{real} - y_{neural}$)	[-]
w_{ij}	neural weight	[-]
T_d	time-delay	[-]
$\boldsymbol{x}$	input to the neural network – column vector	[-]
$\boldsymbol{W}$	weight matrix	[-]
μ	learning rate	[-]
m	size of population in genetic algorithm	[-]
k	number of iterations of the genetic algorithm	[-]
l	number of phases (reproduction and mutation)	[-]
s	number of steps for neural network	[-]

References

[1] SRAMEK, Bo et al: Biomechanics of The Cardiovascular System. Faculty of Mechanical Engineering, CTU, Foundation for Biomechanics of Man, 1995

[2] BUKOVSKY, I.- HOU, Z. - BILA, J., GUPTA, M.: Foundation of Nonconventional Neural Units and their Classification. International Journal of Cognitive Informatics and Natural Intelligence (IJCiNi), October-December 2008, IGI Global, Hershey PA, USA, pp.29-43, ISSN 1557-3958.

[3] BUKOVSKY, I.: Modeling of Complex Dynamical systems by Nonconventional Artificial Neural Architectures and Adaptive Approach to Evaluation of Chaotic Time Series. Ph.D. Thesis, Faculty of Mechanical Engineering, Czech Technical University in Prague (in English, defended September 7, 2007, supervisor Bila, J., supervisor-specialist Gupta, M., online at http://www.fs.cvut.cz/~bukovsky/ivo.htm).

[4] BUKOVSKY, I. - BILA, J.: Adaptive Evaluation of Complex Time Series using Nonconventional Neural Units. ICCI 2008, The 7th IEEE International Conference on COGNITIVE INFORMATICS, California, USA, 2008, ISBN 9781424425389.J.

[5] VIZI, E.: Role of High-Affinity Receptors and Membrane Transporters in Nonsynaptic Communication and Drug Action in the Central Nervous System. Pharmacological Reviews, Vol. 52, No. 1, pp. 63-90, 2000

[6] HINES, W.: A logarithmic neural network architecture for unbounded nonlinear function approximation. in Proc. Int. Conf. Neural Networks 1996, vol. 2, pp. 1245–1250.

[7] BILA, J., MIRONOVOVA, M.: Adaptation of a time-delayed dynamic quadratic neuron for evaluation of bio-signals. ARTEP 2012, ISBN 978-80-553-0835-7.

[8] WILLIAMS, R. – ZIPSER, D.: A learning algorithm for continually running fully recurrent neural networks. Neural Computation, vol. 1, pp. 270–280, 1989.

[9] BUKOVSKY, I. - HOMMA, N.: Dynamic Backpropagation (in Czech). Automatizace, Vol. 52, No. 10, Prague, Czech Republic, Oct 2009, p.586-590, ISSN 0005-125X.

[10] BUKOVSKY, I. et al: Quadratic Neural Unit is a Good Compromise Between Linear Models and Neural Networks for Industrial Applications, 2010 IEEE

[11] MAJDA, F.: Utilization of Artificial Intelligence in Operational Research (in Czech). Research project, FJFI CTU in Prague, 2009

[12] MIRONOVOVA, M., HAVLIS, H.: Calculation of GDOP Coefficient. STC 2011, ISBN 978-80-01-04796-5