Abstract -- The purpose of this paper is to find optimal input values for neural network used in evaluation of bio-signals in order to achieve faster solution in adaptation process of a continuous Time-Delayed Dynamic Quadratic Neural Unit (TmD-DQNU). As optimization tool, a standard genetic algorithm is studied and the method is tested on a certain artificially generated waveform with a basic shape of ECG signal.

Index Terms -- bio-signals, ECG, genetic algorithm, neural network, optimization, time delay

I. 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 this 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.

II. PROCESS DESCRIPTION

The work consists of two parts, first one shows the adaptation of Time-Delayed Dynamic Quadratic Neural Unit (TmD-DQNU) – the process of weights and time-delay upgrade in each step of adaptation using RTRL method of a back propagation (Real Time Recurrent Learning). The second part of the paper shows a use of genetic algorithm as optimization tool. Genetic algorithm creates sets of random values for weights and time-delays and takes typical processes of elitism, elimination, cross-over and mutation to find the best solution. An evaluation parameter for the quality of the solution is a sum of squares of error for each solution. The error signal is represented as a difference between real values of the signal and values of the signal generated by a neural network.

A. 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 Fig. 1.

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

\[ y_n(t) = \int \left[ w_00 + w_{01}u(t) + w_{02}y(t-T_d) + w_{11}y^2(t) + \right. \]
\[ \left. + w_{12}y(t-T_d) + w_{22}y^2(t-T_d) \right] dt \]

In (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:

\[ w_{ij}(t+1) = w_{ij}(t) + \Delta w_{ij} \]

\[ T_d = T_d + \Delta T_d \]

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

\[ \Delta w_{ij} = \mu e(t) \frac{\partial y_n(t)}{\partial w_{ij}} = \mu e(t) \int \frac{\partial}{\partial w_{ij}} (x^T W x) dt = \]

\[ = \mu e(t) \left( \frac{\partial x^T}{\partial w_{ij}} W x + x^T \frac{\partial W}{\partial w_{ij}} x + x^T W \frac{\partial x}{\partial w_{ij}} \right) dt \]
\[ \Delta T_d = \mu e(t) \left\{ \begin{array}{l} -w_{02} \dot{y}_n(t - T_d) - \\ -w_{12} u(t) \dot{y}_n(t - T_d) - \\ -w_{22} 2 \dot{y}_n(t - T_d) \dot{y}_n(t - T_d) \end{array} \right\} dt \]  

(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:

\[ |e| = y_{real} - y_{neural} \rightarrow 0 \]  

(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].

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.**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Case 1</th>
<th>Case 2</th>
<th>Case 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>( w_{00} )</td>
<td>0</td>
<td>0</td>
<td>0.15</td>
</tr>
<tr>
<td>( w_{01} )</td>
<td>-0.4</td>
<td>0</td>
<td>-0.15</td>
</tr>
<tr>
<td>( w_{11} )</td>
<td>0</td>
<td>0</td>
<td>0.15</td>
</tr>
<tr>
<td>( w_{12} )</td>
<td>0</td>
<td>0</td>
<td>0.15</td>
</tr>
<tr>
<td>( w_{22} )</td>
<td>-0.2</td>
<td>-0.2</td>
<td>-0.05</td>
</tr>
<tr>
<td>( T_d )</td>
<td>0.05</td>
<td>0.05</td>
<td>1.05</td>
</tr>
</tbody>
</table>

A progress of adaptation is also shown graphically in Fig. 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].

![Fig. 2. Comparison of \( y_{real} \) and \( y_{neural} \) signals during the adaptation of weights and time-delay - Case 1](image)

**B. 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].

**C. 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 the 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.

**D. 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 stop of calculation in the...
neural model. Though infinite values in the integrator block can be restrained to boundary values, NaN cannot be replaced. For such situations, the whole calculation including the genetic algorithm process is stopped. This situation can be eliminated by restraining integrator output to a certain values. For the given signal and based on the experiments, weights and time-delay shall not overflow ±1, thus integrators outputs can be restrained to ±2 in case of weights, and to 0±2 for the time-delay.

In case that the random values for weights and time-delay in genetic algorithms do not lead to the unstable system, or the situation is secured by restrained values, the calculation proceeds. However, in a genetic algorithm the 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 (empirically, the size of the population should be greater than 4, i.e. m = 10). In the second part of the process, it runs twice in reproduction (r = 2) and once in mutation phase (l = 1). 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 6,000 steps (s = 6,000) as follows:

- in phase 1: m × s = 10 × 6,000 = 60,000 steps
- in phase 2: k × (r + l) × s = 100 × (2+1) × 6,000 = 1,800,000 steps

From the calculation above a total steps in neural network taken during the calculation of a genetic algorithm come to 1,860,000 steps.

From empiric data taken so far, one neural network can adapt to some acceptable value (with some certain error low enough and good input data) 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 ynoral displayed in Chapter II A). However, randomly selected weights and time-delay does not always let the network to converge and the genetic algorithm can be a good approximation tool when the input values need to be estimated and is able to find better input data in a relatively short time.

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

III. 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 in a shorter time. 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 relatively too many steps that prolong the calculation times. This is the main reason, why this method is not efficient for this particular signal. However, by use of a genetic algorithm it is possible to find good input values in a relatively short time that do not lead to instability of the system. These input data can be then used for the adaptation of the neural network in a much larger number of steps with assured output.

IV. REFERENCES