# Application of Artificial Neural Networks for Human Muscle Signal Analysis and Mechanical Equipment Control. 1. Problem overview.

Vadim Gerasimov [1] [a] , Gintaras Jonaitis [2], Vytautas Jonkus [1]
[1] Machine-to-Machine laboratory, Department of Radiophysics,
Faculty of Physics, University of Vilnius, Saulėtekio al. 9-III, LT2054 Vilnius, Lithuania
[2] Laboratory for Medical Rehabilitative and Assistive Technologies, Department of Biomechanics,
Faculty of Mechanics, Vilnius Gediminas Technical University, J. Basanavičiaus str. 28, Vilnius, Lithuania

*Received 12 June 2015, accepted 5 August 2015*

**Abstract.** General principles for creating the electronic system devoted for processing and recognition of myoelectric signals were observed. Due to functional importance, usage of Artificial Neural Networks (ANN) for recognicion purposes were estimated as principle.

**Citations:** Vadim Gerasimov, Gintaras Jonaitis, Vytautas Jonkus. Application of Artificial Neural Networks for Human Muscle Signal Analysis and Mechanical Equipment Control. 1. Problem overview – *Innovative Infotechnologies for Science, Business and Education,* ISSN 2029-1035 – **1(18)** 2015 – Pp. 7-12.

**Keywords:** Artificial Neural Networks; ANN; recognition of myoelectric signals; electromyogram; linear regression method; supervised learning; logistic regression; artificial neuron.
**Short title:** Artificial Neural Networks. 1. Literature overview.

## Introduction

Since the old times people were figuring out how to make their life easier by using machines and mechanisms. Only recently the technological development has become more rapid than anytime before. The control of mechanisms and their interaction with human stimulated the most considerable interest. Such control would not be just mere buttons, but rather sensors, suited for the registration of human body signal.

Such a sensor is a neuron activity scanner or myoelectric probes and electrodes. A signal sensed by these nodes can be barely interpreted seeking required accuracy. In that case signal processing algorithms, which automatically adapt to the required mode, are used. This function is performed by the elements of an *Artificial Inteligence* (AI) algorithm. In this case *Artificial Neural Networks* (ANN) were chosen.

The goal of this work could be formulated as follows: to observe the general principles for creating the electronic system devoted for processing and recognition of myoelectric signals.

## 1. Task overview

The muscular activity is registered by electrodes which form the electric signal as the difference in potential between the beginning node and the end node of certain muscle group. Such myoelectric electrodes (see Fig. 1) are connected to the respective muscle nodes using a conductive gel.

It is very important to realise the true connection circuit - ground of the probes must be connected to a human node lacking any muscles. It should be in very short distance to the point where registration of signal occurs - see Fig. 1.

Each time the electrodes are mounted, the difference in potential changes due to different mounting points being chosen, expendable electrodes may have different conductivity and differently accumulated electric charge can influence the registration of the signals.

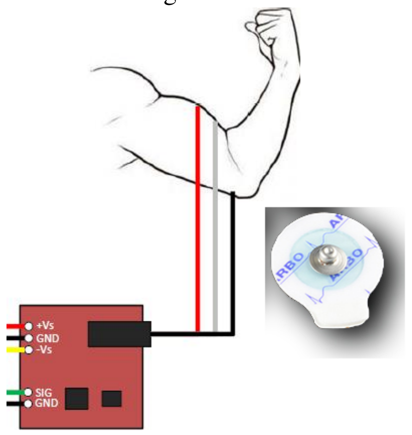

Fig. 1. Myoelectric expendable electrode (right) and basic myoelectric probing circuit. Two probe wires are connected to the beginning and the end of the muscle. The ground wire is connected to the elbow.

[a]Corresponding author, email: vadim.gerasimov@ateities.lt

Each signal is noisy. The noise is particularly powerful when the muscles are in motion while being probed. To reduce the noise, filters and an amplifier could be used. However, it could be expected that the ANN can average the noise as well.

It is insufficient to amplify the signals, they need to be processed, formatted and sent into the further processing node. Such node could be mobile like all the other nodes of the system. Then it is necessary to know how to implement the algorithm suited for such a mobile node - an embedded system. An open ANN library, viable with an embedded system, such as Raspberry Pi could be used.

The ANN architecture that would adapt itself to the existing calibrating data, and at the same time would not be a burden for the embedded system in terms of calculation power should be chosen.

Myoelectric signal recognition works were carried out long time ago. In 1990, Kelly et al [1] attempted to recognise myoelectric signals for prostheses control. At that time it was performed on a serial memory computer, and the processing power did not grant performance to solve the fundamental task. The method that was used in that work was the Hopfield network, suited for time dependent signal registration. That network uses a closed loop method for calculating the synaptic parameters. As stated in Ref. [1], classification was successful using *Sequential Least Square* algorithm.

For a similar purpose, A. Soares et al [2] tried to use *Multilayer Perceptron Backpropagation Feedforward Network* to recognise and classify the myoelectric signal in groups; what the signal was originated from and to control virtual prostheses. Network was comprised of at least 80 neurons. The tests were successful.

**Electromyogram.** The muscle signal in time distribution (potential time-resolved dependency) was titled as *Electromyogram* (EMG). The central nervous system, comprised of the spine cord and peripheral nerves, controls the actions of muscle fibres, which basically determines movements. The muscles consist of special cells, which are able to contract and relax. These actions are stimulated by the motor elements - neurons.

The EMG can be probed in two ways: a) surface mounted EMG (SEMG), when the electrodes are attached to the skin surface; and b) less popular intramuscular EMG - a needle shaped electrode is injected into the muscle. The former variant of electrodes was chosen due to low invasiveness. The SEMG measures muscle fibre electric potential difference for one motor unit. Such potential differences are called motor unit action potentials (MUAP). The potential difference is around 100 mV, but because of the fibre connecting layers and skin, the SEMG is a complex signal with very small amplitude (from $\mu$V up to 5 mV). Fig. 2 represents the typical SEMG as a EMG from skin surface.

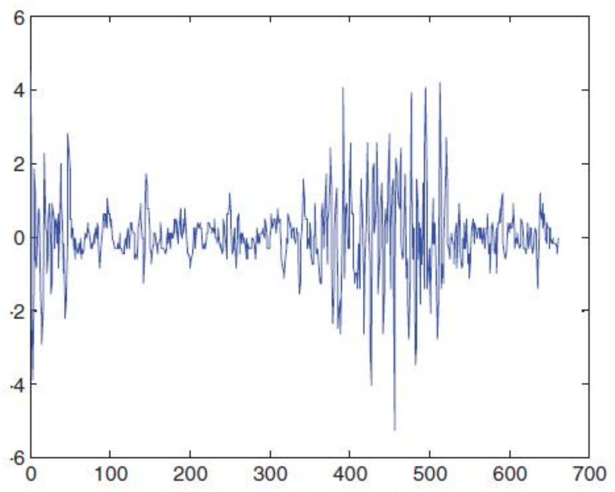

Fig. 2. Typical SEMG as a EMG from skin surface. Abscisse $x$ represents the index of set, ordinate $y$ - signal potential. Adapted according to Ref. [3].

## 2. Algorythms and learning processes

*Learning a program or algorithm* is a process that gives better performance, while storing information about the environment. Initially the multi-way algorithm could work not ideal due to absence of true conditions. To forecast all situations in life is impossible.

The task of robot in labyrinth is quite suitable for explaining. Let's say, a robot is given motion in a labyrinth. Each new labyrinth does not give any chance to the programmers to foresee each entangled turn; only a learning algorithm would be able to do something like that. The programmers could not expect any changes in a system, for example, changes in market prices that would be good to predict for future. There are such tasks that the programmers themselves do not know how to solve them; instead they rely on the learning algorithm.

Machine learning must have a feedback; otherwise the mechanism will not know what is *true* or *false*. There are several machine learning feedback types.

**1. Unsupervised learning.** At this point a machine or a robot must learn the structure of the given problem. These tasks are usually solved by data clustering or genetic algorithms. It means that the algorithm must choose the correct input data by itself. Suppose, if a robot-driver was let in town, it would have to know days, jammed by traffic, and free of it.

**2. Reinforced learning.** In this case the program must get "encouragement" or "penalty" for the results. This means, should a robot-driver hit a pillar, give him the *bad* signal, or give a *good* signal if the pillar was successfully bypassed. Which actions induced the encouragement or penalty is to be determined by the algorithm.

**3. Supervised learning.** Certain actions are provided to the programme taking into consideration sensory input data. Simply imagine, that the instructor dictates, where to turn the wheel, when the robot-driver drives up to the curb.

Table 1. Learning data.
Signal potential value from mioelectric sensor at different time ($t_1$,$t_2$,$t_3$,$t_4$,) and corresponding shift $d$

| $t_1$, | $t_2$, | $t_3$, | $t_4$, | $d, cm$ |
|---|---|---|---|---|
| 0,07 | 0,09 | 0,10 | 0,09 | 28 |
| 0,08 | 0,11 | 0,14 | 0,10 | 46 |
| 0,08 | 0,10 | 0,11 | 0,11 | 25 |
| 0,06 | 0,09 | 0,11 | 0,07 | 36 |
| 0,07 | 0,08 | 0,1 | 0,07 | 49 |

*Supervised Learning* plays the most important role in the artificial intelligence systems. Also supervised learning is an objective of this work.

Suppose, sensors give some input data, which will be denoted as matrix **X**. Its columns are of a separate myoelectric sensor input data, and rows represent a separate sample. Logically though, one needs a **Y** matrix, the columns of which are the decision data set (that or this action), and the rows are the decision sample set. Further, an example that was actually implemented in this work will be provided.

Myoelectric sensors of system produce the time-dependent signals as values $x_{tn}$ for recording. Index $t$ represents the registration time and $n$ - sensor order. Considering the **X**, one needs to know the $y_{tn}$ data, which at the given time $t$ show what decision $n$ should be made. At this point some conditions are being created, which are listed in Table 1.

Definitely, the number of samples of data is too little, it should be several hundreds. The task could be formulated as follows: how certain data values could be related with a continuous function? This is where the approximation algorithm is of great help.

The task of a supervised algorithm is to find such a so called *hypothesis* function $H$, which can be adapted to table's patterns and other unforeseen results, which is to approximate the actual function $f(x)$. Fig. 3 represents the results of a hypothetic experiment, where scattered two-dimensional data could be estimated as weak-correlated dependency.

To make conclusions about the experiment, it is necessary to summarize the data point distribution, i.e. describe the data with some kind of a function, which would approximate the pattern. Fig. 3 represents also and two possible approximations: linear as well as parabolic.

The error of the applied function could be measured while testing on test data, which were not used while fitting the function, although still connected with the learning data. A well chosen hypothesis function has not only the least error with the learning data, but also with the test data, which means, that given new unknown input data, the function predicts the answer the best.

Sometimes a function $f(x, \cdots)$ could be estimated as stochastic. At this point one needs to know a relative distribution of a function $P(Y|x)$. If the output data are discreet and of finite number, then the task is called *classification*. When the

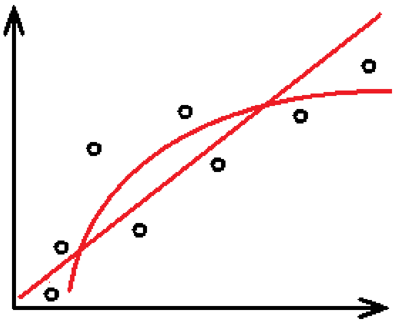

Fig. 3. Typical data of simulated experiment (open dots). Linear and parabolic approximations in red.

data are of *yes/no* type, then the task is called *Boolean classification*. If the output data are continuous, then the task is called *regression*. Solving a regression problem is to find the average expected value of $y$. Statement that **H** exactly matches the $f$ function is false, the probability is equal to zero.

It is very important then to define the "best approximation". The error rate of a hypothesis is a quantity of $H(x) \neq y$ occurrences in a (**X**,**Y**) space. A low error rate of an $H$ function does not mean that chosen hypothesis function is good. There is a chance, that the function is precise to the given training data, though not to the test data. This effect is called *overfitting*, which is withdrawn when testing the function with the test data or rather called the validation data, decreasing the function's precision until reaching the least average error with the test data. It is called the cross validation and model selection.

This technique is implemented in the MatLab [4] modules. However, since the chosen instrument happens to be an open library which was installed in the processing unit, it has no validation options and function precision is chosen by default - first order.

## 4. Linear regression method

Let us formulate the following task [5]: we need to adjust a linear function $y$ to the set of data points:

$$y = \theta_1 \dot{} x + \theta_0 \tag{1}$$

Three vectors: **X**, **Y** and **Θ** represent the arguments, values and weight coefficients (also known as the synaptic parameters in the neural networks). Choosing the weight parameters is intended to minimize the overall empirical error. Traditionally that is achieved using the function of difference square $L_2$:

$$Error(\mathbf{H}_e) = \sum_{j=1}^{N} L_2(y_j, \mathbf{H}_e(x_j)) \tag{2}$$

$$Error(\mathbf{H}_e) = \sum_{j=1}^{N} (y_j - \mathbf{H}_e(x_j))^2 \tag{3}$$

$$Error(\mathbf{H}_e) = \sum_{j=1}^{N} (y_j - (w_1^{\cdot} x_j + w_0))^2 \tag{4}$$

This task requires to minimize the function $Error$. First derivative is equal to zero:

$$\frac{\partial}{\partial\Theta_0}\sum_{j=1}^{N}(y_j-(w_1 x_j+w_0))^2=0 \quad (5)$$

$$\frac{\partial}{\partial\Theta_1}\sum_{j=1}^{N}(y_j-(w_1 x_j+w_0))^2=0 \quad (6)$$

The solution of the system is then unique:

$$\Theta_1=\frac{N\cdot(\sum x_j y_j)-(\sum x_j)\cdot(\sum y_j)}{N(\sum s_j^2)-(\sum x_j)^2} \quad (7)$$

$$\Theta_0=\frac{(\sum y_j-\Theta_1(\sum x_j))}{N} \quad (8)$$

The learning algorithms are based upon the change of the weight coefficients so that the error function is as small as possible. In case of a straight line, the space of weight coefficients is two dimensional. Such function, arguments of which are only two weight coefficients and the value of it is the error, has a concave shape and a global minimum. The successful choosing of such a minimum terminates the learning in terms of a linear regression. Choosing other functions, far more complex than a mere straight line, may render issues with representing the weight coefficients. In this situation an overall optimization problem must be solved altogether.

To do that, hill leap or a gradient descent algorithm is used, while tracking the gradient value in the given point. Starting from any parameter set, and:

$$\Theta_1 \leftarrow \Theta_1-\alpha\frac{\partial}{\partial\Theta_i}Error(\mathbf{\Theta}) \quad (9)$$

Here $\alpha$ represents so called learning rate or size of the step. It can be constant or change in time. Let's express the hypothesis function $\mathbf{H}$ as the multiparametric function:

$$\mathbf{H}_{sw}(\mathbf{x}_j)=\Theta_0+w_1\cdot x_{j,i}+\cdots+w_n\cdot x_{j,n} \quad (10)$$

Here $x_{j,0}$ is equal to one. Such function can be represented as a product of two matrices:

$$\mathbf{H}_{sw}=\mathbf{\Theta}\cdot\mathbf{X}_j=\sum_i\Theta_i\cdot x_{j,i} \quad (11)$$

Hence the solution can be found not only in the gradient proximity, but with an equation:

$$\mathbf{\Theta}^*=(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y} \quad (12)$$

## 5. Classification and logistic regression

Linear functions can also be used for classification. Seismic data task is very suitable for such purposes. Let $x$ and $y$ are seismic data as bulk and surface wave amplitudes. The probability of an earthquake or an underground explosion depends on these two data sets, which, in turn, are of great interest to seismologists or military experts.

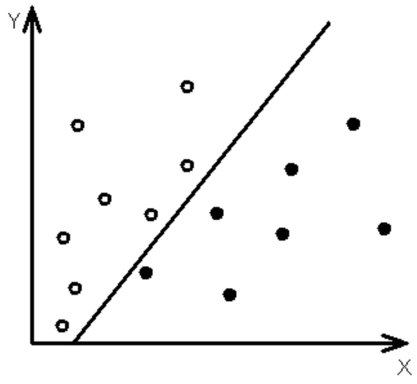

Fig 4. Typical seismic data (open dots). Straight line separates data groups into classes.

For example, lets determine the quake value as 0 and explosions as 1. Fig. 4 represents data distribution, where separation of data groups takes place.

The decision boundary is a line (or a surface in a multidimensional space) which separates two data classes. This boundary is called the linear separator. This time the equation can be inexplicit :

$$\theta_0+\theta_1\cdot x-y=0 \quad (13)$$

Hence the classification condition can be formulated as follow:

$$\mathbf{H}_\Theta=1,\quad if\quad (\mathbf{\Theta})\cdot\mathbf{X}\geqslant 0 \quad (14)$$

$$\mathbf{H}_\Theta=0,\quad if\quad (\mathbf{\Theta})\cdot\mathbf{X}<0 \quad (15)$$

Similar to the threshold function algorithm shown below is called the perceptron learning rule:

$$\Theta_i\leftarrow\Theta_i+\alpha(y-\mathbf{H}_\Theta(\mathbf{X}))\times x_i \quad (16)$$

Sometimes it is inconvenient to classify using threshold functions, when a continuous traverse over the separator is needed. Also the threshold function cannot be differentiated at the separator (because a derivative of a sudden leap function is delta function). For that purpose a continuous differentiable sigmoidal function was created:

$$\sigma(z)=\frac{1}{1+\exp(-z)} \quad (17)$$

Second order sigmoid containing linear regression function could be used for validation and regularization purposes:

$$\mathbf{H}_\Theta(\mathbf{X})=\sigma(\mathbf{\Theta}\cdot\mathbf{X})=\frac{1}{1+\exp(-\Theta\cdot X)} \quad (18)$$

Sigmoid and the threshold function are presented in Fig. 5. The sigmoid weight coefficient selection for error minimization is called the logistic regression. This problem cannot be solved analytically; however it can be solved using the "gradient descent". Eqns. (19-20) represent the mentioned algorithm.

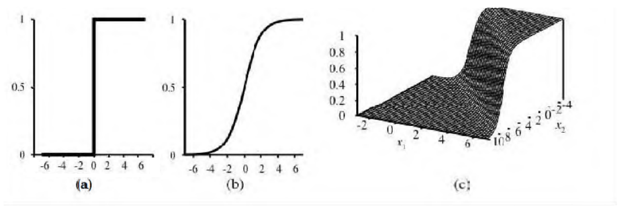

Fig 5. Different threshold functions: a) 1-D threshold function; b) 1-D sigmoid function; c) 2-D sigmoid function, suitable for data approximation presented in Fig. 4. Adapted according to Ref. [3].

$$\frac{\partial}{\partial\Theta_i}Error(\mathbf{\Theta}) = \frac{\partial}{\partial\mathbf{\Theta}_i}(\mathbf{y} - \mathbf{H}_\Theta(\mathbf{X}))^2 \qquad (19)$$

$$\frac{\partial}{\partial\Theta_i}Error(\mathbf{\Theta}) = 2\cdot(\mathbf{y} - \mathbf{H}_\Theta(\mathbf{X})) \times \mathbf{H}' \times x_i \qquad (20)$$

The derivative of sigmoid function $H$ satisfies the given condition:

$$\mathbf{H}'(z) = \mathbf{H}(z)(1 - \mathbf{H}(z)) \qquad (21)$$

hence the error minimization procesure was done by following expression:

$$\Theta_i \leftarrow \Theta_i + \alpha(y - \mathbf{H}_\Theta(\mathbf{X})) \times \mathbf{H}_\Theta(\mathbf{X})(1 - \mathbf{H}_\Theta(\mathbf{X})) \times x_i \quad (22)$$

## 6. Neural networks

The term *neural networks* is derived from the brain activity research fields; however, the only thing they have in common is the idea and the components' name. The artificial neuron is comprised of several parts, including input node, working node (input function, bias weight, activation function) and output node (dendrite) - see Fig. 6.

The artificial neuron is basically a way the regression or the classification functions and their parts are represented - it is a concept. To be able to talk about *Artificial Neuron Network* (ANN), they should be connected into an entity - a certain topology. Neuron $i$ is connected to the neuron $k$, so that the activation function $a_i$ would be passed from $i$ node to $j$ node. Each node has its own weight parameter $\theta_{i,j}$ which determines the connection status. Additionally the neuron has also the bias weight node $a_0$=1 with a related weight parameter $\theta_{0,j}$. First, each $j$ node calculates its input sum:

$$in_j = \sum_{i=0}^{n} \theta_{i,j} \cdot a_i \qquad (23)$$

And then it interposes it into the activation function - usually sigmoid:

$$a_j = g(in_j) = g\cdot(\sum_{i=0}^{n} \theta_{i,j} \cdot a_i) \qquad (24)$$

The ANN can be divided into two big topological groups: one way sequential and recursive (with a feedback) transfer. One way sequential network is grouped by layers.

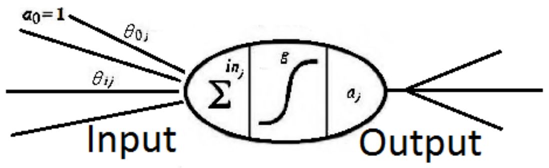

Fig. 6. Diagram of a neuron. Adapted according to Ref. [4].

During the execution mode each a layer receives the data only from the previous layer, and accordingly transfers only to the next layer. Fig. 7 represents the typical ANN diagram.

The input layer consists of the input information neurons, which receive the data directly or reduced from the sensors.

The intermediate or "hidden" layers are the main processing neurons, which contribute to the overall hypothetic function shape. The output neurons retrieve the data to the execution nodes directly or in a reduced form, for example, to an electromechanical system. Each argument of the neuron activation function stems from the previous neuron activation function value.

A single hidden layer can represent any continuous function to a certain precision and adding the second hidden layer enables the network to represent discontinuous functions. Adding more layers may represent any other kind of functions, though the working mechanism becomes rather difficult to understand. Due to it, choosing the ANN architecture demands model experimental tests - selecting such a network that would satisfy the objective conditions.

The task of the selected network is to find such a hypothetic function, which would give the needed feedback after taking certain inputs. In other words, the network training is a convergence of the hypothetic function to the real function, which in turn describes the pattern. To converge the function one needs to calculate the error in the output neuron, dependent on the synaptic parameter $\Theta$:

$$\frac{\partial}{\partial\Theta}Error(\mathbf{\Theta}) = \frac{\partial}{\partial\Theta}[\mathbf{Y} - \mathbf{H}_{\mathbf{\Theta}}(\mathbf{X})] \qquad (25)$$

$$\frac{\partial}{\partial\Theta}Error(\mathbf{\Theta}) = \frac{\partial}{\partial\Theta}\sum_k(y_k - a_k)^2 \qquad (26)$$

$$\frac{\partial}{\partial\Theta}Error(\mathbf{\Theta}) = \sum_k \frac{\partial}{\partial\Theta}(y_k - a_k)^2 \qquad (27)$$

Here $Y$ represents the output data vector, which shows a single exemplar output datum, comprised of several parts; for example, if probed muscles are stretched - turn the motors with a certain speed, which is demanded by the controller. **H** represents the hypothetic function vector, parts of which evaluate the output neuron components separately (e.g. for each sensor). $\Theta$ represents the synaptic parameter matrix, the elements of which evaluate the components of the previous neurons and the output neurons.

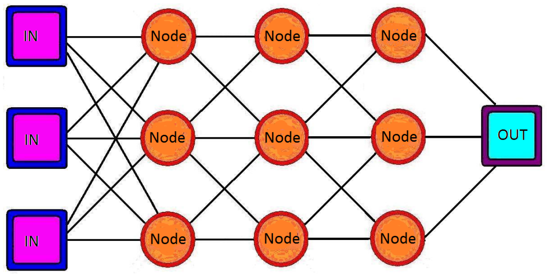

Fig. 7. ANN system based on three hidden layers.

The $k$ index determines the number of the output neurons, which indicates how many controllable parameters are there, e.g. motor power or actuation. The calculated and minimized error in the equation is written only in terms of the output layer.

For the network to converge, it is necessary to transfer the calculated error from the output layer back to the hidden layers, down up to the input layer. This algorithm is called the back propagation algorithm.

Let $Err_k$ is a component of the $(\mathbf{Y} - \mathbf{H})$ vector. Then we describe the modified error:

$$\Delta_k = Err_k \times g^{'}(in_k) \tag{28}$$

where

$$g^{'} = \frac{d}{dx}[\frac{1}{1+\exp(in_k)}] \tag{29}$$

Hence the weight coefficient selection (or simply learning) rule is as follows:

$$\theta_{j,k} = \theta_{j,k} + \alpha \times a_j \times \Delta_k \tag{30}$$

The hidden layer $j$ node function component error is calculated with the following equation:

$$\Delta_j = g^{'}(in_j)\sum_k \Theta_{j,k}\Delta_k \tag{31}$$

Briefly, the back propagation learning algorithm is in fact the minimization of the error between the output data and the calculated neural network output data in each layer where each node minimizes the error depending on the error of the $j$ node, which in turn is the next after the $i$ node. Mathematically the error calculation in each of the nodes and its transfer look like this:

$$\frac{\partial Error_k}{\partial \Theta_{i,j}} = -2(y_k - a_k)\frac{\partial a_k}{\partial \Theta_{i,j}} = -2(y_k - a_k)\frac{\partial g(in_k)}{\partial \Theta_{i,j}} \tag{32}$$

$$\frac{\partial Error_k}{\partial \Theta_{i,j}} = -2\Delta_k \Theta_{j,k} g^{'}(in_j) a_i = -a_i \Delta_j \tag{33}$$

Here $\Delta_j$ was described above, and the equation ultimately means, that the point of minimizing the error is to transfer the minimized error back into the next layer - actually the previously standing layer's nodes.

The neural network execution mode is comprised of data transfers into the input layer, and the calculation of the reactive data with the converged hypothetic function.

## Conclusions

General principles for creating the electronic system devoted for processing and recognition of myoelectric signals were observed.

## References

1. Micheale F. Kelly, Philip A. Parker, Robert N. Scott. The Application of Neural Networks to Myoelectric Signal Analysis: A Preliminary Study. – *IEEE Tran. Biomed. Eng.* 37(3) (1990) 221-230.
2. Alcimar Soares, Adriano Andrade, Edgard Lamounier, Renato Carrijo. The Development of a Virtual Myoelectric Prosthesis Controlled by an EMG pattern Recognition System Based on Neural Networks. – *Jour. Intel. Info. Sys* 21(2) (2003) 127-141.
3. Ramaswamy Palaniappan. Biological Signal Analysis. – Ramaswamy Palaniappan Ventus Publishing, 2010.
4. <https://se.mathworks.com/help/nnet/ug/multilayer-neural-networks-and-backpropagation-training.html?requestedDomain=se.mathworks.com>, accessed 2014.06.25.
5. Stuart J. Russel, Peter Norvig. Artificial Intelligence. A Modern Approach. 3rd ed.. – Publ. Prentice Hall, 2010 – P. 1134.