A graphical software application for the ergonomic design of a working place

A Ene1,*

1 University of Pitesti, Electronics, Computers and Electrical Engineering Department, Târgul din Vale Street No.1, Romania
Email: alexandru.ene@upit.ro

Abstract. A working place has to be designed in order to offer with minimum cost adequate working conditions. The labor yield is much influenced by environmental measurable parameters like temperature, humidity, noise and the luminosity at the working place. In this paper we present a graphical application that could be used to investigate which is the best combination for the values of these input parameters from the point of view of work efficiency. The relationship between the input parameters and the work efficiency is nonlinear. Therefore, we used a neural network to model these nonlinearities. The neural network is simulated using Java programming language.

1. Introduction
The rapid evolution of industrial production in the recent years has led the employers to monitoring the factors that can contribute the work productivity. The need to ensure a safe working environment is a prerequisite for the provision of high quality products and services. A comprehensive analysis of working conditions allows business managers to define the skills needed for specific jobs [1].

The designer of a working place, has to take into account different categories of factors that influence the quality of the working place:
- microclimate factors (temperature, humidity, noise, lighting, electromagnetic radiation);
- physical factors (energy consumption, static stress, repetitiveness of movement);
- psychological factors (excess of information, monotony);
- technological and organizational factors (factors related to the workplace organization and to the technical equipment).

We developed a software application that is used for the qualitative evaluation of a working place, ranking it in three different categories: poor quality, moderate quality and high quality.

In order to evaluate a working place, we considered four input factors, that can be easily measured:

- Temperature;
- Humidity;
- Noise;
- Luminosity.

*Temperature - is a factor that has a high influence on the efficiency of work. The thermal comfort is establishing when there is a maximum difference of 2-3 °C between the temperature of the objects and the ambient temperature.
Relative Humidity - can be defined as the amount of water vapor in the air in percent. The human body, during physical activities, sweats for the purpose of cooling the body. More relative humidity is high, more the evaporation rate of the sweat is reduced, increasing the discomfort.

Noise - can be defined as a chaotic mixture of disagreeable, embarrassing or harmful sounds. Noise has the effect of lowering the power of concentration and the occurrence of fatigue. It is characterized by: frequency, intensity and duration. Above 85 dB, noise becomes disturbing, and for longer period, even dangerous.

Lighting - affects visual comfort, a poor lighting affects the human operator, causing operator fatigue and lowering work efficiency. Fatigue caused by inappropriate illumination is manifested by: reducing the precision of the operator's movements, headaches, the onset of dizziness and insomnia. Limit values: 200 lux for typical work, for general mechanics 400 lux, for difficult and laborious activities 800 lux.

The quality of the working place is estimated with a feed forward neural network.

2. Feed forward neural networks

The feed forward neural networks [4] are built from interconnected artificial neurons, grouped in layers. They can be software simulated or can be hardware implemented. In this paper the neural network was software simulated using Java programming language.

Neural networks are excellent tools both in pattern recognition applications and, also due to their ability to discover regularities in their input data, in prediction applications [2][3]. For these type of applications, usually there are three layers of neurons: an input layer, a hidden layer and an output layer.

The neurons from the input layer are the only one that receive external inputs.

An artificial neuron from a layer must be connected with each neuron from the next layer. In the same layer the neurons are not interconnected. Each connection has associated a real numeric value, called weight.

The first step in designing a neural network is to determine the number of neurons in each layer.

The number of hidden layer neurons is experimentally determinate, but the number of the neurons from the input and the output layer are derived from the problem that is solved by the network.

The second step is to train the network using training patterns. The network has to learn the training patterns with a specified accuracy (the total error of learning). In the process of training, the weights of the network are adjusted, and at the end of the training, all the weights are calculated. When the network succeeded to learn the training set with the specified error, it is said that the network converged.

The “intelligence” of a feed forward neural network to solve a problem is located in the values of its weights. So, the goal of the training stage is to calculate the values for the weights.

The results obtained with the neural network are very much influenced by the quality of the training patterns that are used (their number and their distribution for the output classes).

Feed forward neural networks are trained using the backpropagation algorithm [4].

A general form of the back propagation algorithm is:

\[
\text{read the value of total error of learning (TOTAL\_ERROR)} \\
\text{initialize all the weights of the neural network with small real values} \\
\text{set converge flag to false} \\
\text{REPEAT} \\
\text{for (all patterns from the training set)} \\
\text{begin} \\
\text{- forward propagate the input of the current training pattern through the network (i.e. compute all the outputs of the neural network, having the inputs of the current training pattern, on the network’s inputs)} \\
\text{- calculate the error of learning the current pattern} \\
\text{- modify each weight, starting from the output layer to input layer (back propagate)} \\
\text{end} \\
\text{calculate the total output error of learning the whole set of examples, E} \\
\text{set converge flag to true} \\\	ext{UNTIL converge flag is true} \\
\text{END} \\
\]


if ($E \leq TOTAL\_ERROR)$ set converge flag to true
\textbf{UNTIL} converge flag becomes true

The weights are modified at the back propagation stage, in accordance with the delta rule [4].

3. Implementation

We developed three software applications.

The first application has as input, the xml file with the training patterns.

A training pattern consists of five numerical values: four values for the inputs and one output value.

An example of a training pattern from the xml file, is the following:

\begin{verbatim}
<sablon>
  <t>22</t>
  <u>40</u>
  <z>40</z>
  <l>800</l>
  <nota>10</nota>
</sablon>
\end{verbatim}

The first software application takes the xml file, preprocess the data and then, saves the preprocessed data into a text file.

In the text file, a training pattern example, has the following structure:

\begin{verbatim}
0.22 0.4 0.4 0.8 1.0
\end{verbatim}

The second software application uses the text file of patterns in order to train the feed forward neural network.

\begin{figure}[h]
\centering
\includegraphics[width=0.4\textwidth]{interface.png}
\caption{The interface with the user}
\end{figure}

The feed forward neural network was simulated using Java programming language. We used the following neural network architecture:

- number of input neurons: 4;
- number of hidden layers: 1;
- number of neurons in the hidden layer: 10;
- number of output neurons: 1.
An output value in the range (0, 0.33) is for a poor quality working place, an output value in the range [0.33, 0.66) is for a moderate quality working place, and an output value in the range [0.66, 1) is for a high quality working place.

The third application has a graphical interface. The user inputs in some text fields the input data for a specific working place.

The Calculate button will evaluate the working place, using the neural network.

The New button is used to clear all the input text fields, in order to introduce new input data.

4. Results
We choose the total training error to be 0.01.

The weights are iteratively calculated with the backpropagation algorithm.

Here is a log from the running of training application:

\[
\text{Epoch 1011: total error} = 0.010026545811339894 \\
\text{Epoch 1012: total error} = 0.010012017936378629 \\
\text{Epoch 1013: total error} = 0.0099975144827424
\]

*The network converged in 1013 epochs.*

*The weights had been saved.*

In order to test the network’s ability to estimate the quality of a working place, we tested it with different sets of input data. These input data were new for the network. They were not used when we trained the neural network.

All the results obtained when we tested in this way the network, were correct.

Here is such an example:

![Figure 2. The result of the simulation](image)

5. Conclusions
This paper proposes a practical application for neural networks: the estimation of the quality of a working place. The working place is evaluated using four measurable input factors: temperature, humidity, noise and luminosity.

In order to be easily used, the software application has a graphical interface.

As shown in the experimental results section of this paper, the neural network has the ability to discover regularities in the training data set that it learned, and based on these regularities to estimate the quality of a working place, when different combinations of values for temperature, humidity, noise and luminosity that are measured at a working place, are placed on networks’ inputs.
The results obtained with this software application are useful both to designers of working places and to managers.

References
[1] Grzybowski W. 2001 *Int. J. of occup. Saf. and ergonom.* 7 (2)
[2] Ene A. and Știrbu C. 2016 *ECAI2016* 8 159-162
[3] Ene A. and Stirbu C. 2016 *Univ. of Pitesti Sc. Bull. (Series: Electr. and Comp. Sc.)* 16 (2) 8-11
[4] Ene A. and Stirbu C. 2008 *Retele neuronale. Teorie si aplicatii in Java*, Editura Universitatii din Pitesti
[5] Manolescu A., Lefter V., Deaconu A. 2010 *Ergonomie*, Editura Economică 333-338.