JAVA NEURAL NETWORK APPLICATION FOR THE ESTIMATION OF A WORKPLACE RANKING

Alexandru ENE, Daniel ANGHEL
FECC, University of Pitesti, Romania
alexandru.ene@upit.ro
daniel.anghel@upit.ro

Keywords: multi layer perceptron neural networks, non-liniar input-output relationships, workplace ranking

Abstract: *This paper presents a Java implementation of a feed forward neural network, used for the ranking of a working place. We took into account the following input parameters: temperature, humidity, noise and luminosity. These parameters determine the degree of comfort of a working place. Because the relationship between those four parameters and the output result (workplace ranking) is non-linear, a neural network is a very good tool to study this ranking. A neural network has the ability to quantify even non-evident relationships between input parameters as far as we have a set of good training examples.*

1. INTRODUCTION

Due to the rapid growth of industrial production in the last years, the employers have begun to monitor the factors that may contribute to the productivity in a specific workplace. The need to ensure a safe working environment is a prerequisite to providing high quality products and services. A comprehensive analysis of working conditions allows corporate managers to define competences required in specific jobs [1].

In order to evaluate the workplace , we have to consider the following categories of factors [1]:

-physical working environment factors (noise, vibration, microclimate, lighting, dust levels, toxicity, electromagnetic radiation)

-physical strain factors (energy consumption, static strain, repetitiveness of motion)

-psychological strain factors (information overload, monotony)

-technological and organizational factors (factors related to workplace organization and technical equipment).

In this paper, for the evaluation of a work place, we took into account only four input parameters:

- Temperature
- Humidity
- Noise
- Luminosity

All these four parameters can be easily measured.

The workplace is evaluated by calculating a value, a real number between 0.0 and 1.0 , the rank of the workplace, where 0.0 means the worst rank and 1.0 is the best rank.

For the ranking of a workplace we used a feed forward neural network, that was simulated in Java programming language.

2. FEED FORWARD NEURAL NETWOKS

A neural network is a network of artificial neurons. It can be software simulated or hardware implemented. It successfully represents complex input-output relationships, in a system. These relationships could be either linear or non-linear. In order to learn these relationships, that network has to be trained. The training of the neural network is supervised or unsupervised, depending on the architecture of the network. For our application we used a feed forward neural network, also called multi-layer perceptron (MLP). This architecture is very used in practical applications. They are mostly used to classify an input pattern (could be an image, a sound, etc) in a class from an output set of classes. For example a MLP neural network could be used to recognize the hand-written digits. In this case the image with the digit presented on the inputs of the neural network , will be classified as one from the 10 possible output classes: class 0 – that corresponds to digit 0, class 1- that corresponds to digit 1 etc. In majority of this kind of applications, the image that has to be recognized, must be preprocessed, in order to reduce the number of input neurons in the neural network. The same type of classification is also met when a MLP network is used to diagnose something. For instance, to classify a mechanism as good or bad, based on some input data.

The second type of applications in which MLP networks are used are those in which the network computes some certain output values, that represents some estimation values from a process. We use this kind of application in our work , that we present in this paper.

The processing units in a MLP network are grouped in layers. Typically there is an input layer, an output layer and one or more intermediate layers, called hidden layers. A neuron from a layer is connected only with each neuron from the next layer. The neurons from the same layer are not interconnected. Each interconnection has associated a real number positive or negative, called *weight*.

We use to specific a certain architecture for a MLP neural network, a notation in which is presented the number of neurons in each layer. For instance a neural network that has 25 input neurons, two hidden layers with 10 and 7 neurons each, and an output layer with a single neuron, will be noted : 25 - 10 – 7 – 1 .

The number of input neurons and the number of output neurons are determined based on the problem to be solved. For instance, a neural network used to recognise a black and white image of 10 x 10 pixels, that represents a hand written digit, will have 100 neurons in the input layer (each pixel from the image is connected to an input neuron) and will have 10 output neurons, that corresponds to the 10 possible digits.

Typically, the number of hidden layers and the number of neurons in each hidden layer, are experimentally chosen .

The first stage in designing a feed forward neural network for a specific application is to determine the number of layers and the number of neurons in each layer [2][3].

The second stage is training the network to learn a set of examples . For a NI – NH – NO neural network, a training example consists of NI values (the number of neurons in the input layer) and other NO values (the number of neurons in the output layer). The NO values are those obtained on the outputs of the system, when we place the NI values from the example, on the inputs of the system [2][3].

Training a feed forward neural network is supervised. The goal of the training is to determine all the weights of the network, so that the whole set of training examples to be learned with a specific error, The most used training algorithm for the feed forward neural networks, is the back propagation algorithm [2][3].

A brief description of this algorithm:

TOTAL_ERROR=0.01;
-initialize all the weights of the neural network with small random real numbers
converge=false;
do{
for(all training examples**){**

- forward propagate the input of the current example through the network

- compute the output error of the current example

- back propagate this error in the network, adjusting each weight

}

-compute E, the global output error of learning the whole set of examples

if(E<=TOTAL_ERROR)converge=true;

}while(converge==false);

The invention of this training algorithm, some thirty years ago, was a fact that decisively contributed to the success of neural networks practical applications.

3. NEURAL NETWORK DESIGN

The number of input neurons and the number of output neurons of the network is determined by the problem we want to solve. So, we use 4 input neurons that corresponds to the 4 input variables: temperature, humidity, noise and luminosity, and we use a single output neuron that corresponds to the ranking result.

This result was normalized to be a real number between 0.0 (the worst rank) and 1.0 (the best). This is also the range of output data for a sigmoidal artificial neuron. These kind of neurons are used in the hidden layer and in the output layer of a feed forward neural network.

We use also a single hidden layer with 5 neurons.

We had a set of 16 examples, experimental data used to rank a working place.

Here is such an example:

Temperature = 21 (Celsius degrees)
Humidity = 40
Noise = 60 dB
Luminosity = 300 lux

For this set of parameters , the corresponding ranking result is 0.7 .

In order to be used in training the neural network, we normalized all the input parameters in range 0,0 – 1.0. So , we divided by 100 the data for temperature, and the data for the humidity and the data for the noise. We divided by 1000 the data for the luminosity.

We used 75% from examples we had, to train the network. So, we had a 12 examples training set. The rest of 4 examples, we used to test the neural network.

We fixed the total learning error to 1%.

4. DESCRIPTION OF THE PROGRAM

The program implements in Java language the backpropagation algorithm that is used to train a feed forward neural network. The examples used in training are stored in a text file SabloaneCotare.txt. For each training example there are two lines in the text file , as in the following example:

0.25 0.35 0.50 0.500
0.9

In the first text line there are the 4 input values for the neural network, and in the second line there is the ranking value that has to be obtained at the output of the network.

The test values are stored in a text file called TestareCotare.txt. It has the same format as the training text file.

5. EXPERIMENTAL RESULTS

We present some experimental results obtained during the training process.
In a training epoch the whole set of examples (in our application the 12 examples) are presented forward and backward to the network.
In the simulation below, are shown the number of the current epoch in the training process and the total current error of learning.

epoch: 1 error = 0.7464226274084106
epoch: 2 error = 0.7288003243213822
epoch: 3 error = 0.7260531055598923
epoch: 4 error = 0.7240722071623417
epoch: 5 error = 0.7212415875648821
epoch: 6 error = 0.7176699874232971
epoch: 7 error = 0.7134997676804629
epoch: 8 error = 0.7087725735746828
epoch: 9 error = 0.7034692978733738

epoch: 10 error = 0.697541681345853
……..…..…
epoch: 847 error =0.010088997882058968
epoch: 848 error =0.01007433634670244
epoch: 849 error =0.01005970813633451
epoch: 850 error =0.010045113148643273
epoch: 851 error =0.010030551281929114
epoch: 852 error =0.010016022435100599
epoch: 853 error =0.010001526507670519
epoch: 854 error =0.009987063399752028

So, in this example, the neural network converged in 854 epochs. Of course, due to the random nature of the initialization of the weights, this number of epochs in which the neural network learns the training set, is variable.

Anyway, through simulations we can conclude that it learns quickly the training set.

After the training is complete, with the weights obtained, the program calculates the results obtained by the network for input test examples, those that are stored in the file TestareCotare.txt.

Here are these calculated results presented in comparison with the ideal results (those that are stored in the test file).

Ideal result: 0.2
Computed result: 0.227455987010402

Ideal result: 0.5
Computed result: 0.39180674849614694

Ideal result: 0.9
Computed result: 0.9635665298840688

Ideal result: 0.3
Computed result: 0.3305110639549912

It is interested to note that if we use some bad corrupted test data , the neural network will not correctly answer. For example:
With the following corrupted test:

0.37 0.75 0.90 ***0.600***
0.2

The correct test was:

0.37 0.75 0.90 ***0.060***
0.2

The program gave the following answer:

Ideal result: 0.2
Computed result: 0.6607622665694328

6. CONCLUSIONS

In this paper we presented a practical application for the MLP neural networks : the ranking of a workplace.

As it can be seen from the experimental results, the designed feed forward neural network recognizes well the test data. Even better results could have been obtained if we had a big number of training examples.

For the moment we used only 4 parameters to characterize the workplace: temperature, humidity, noise and luminosity. Because as shown through this application, the results are encouraging, we intend to add more input parameters.

7. REFERENCES

[1] Wieslaw Grzybowski, A method of ergonomic workplace evaluation for assessing occupational risks at workplaces, International Journal of occupational safety and ergonomics 2001 vol. 7, No. 2

[2] Alexandru Ene, Reţele neuronale. Teorie şi aplicaţii în limbajul C, Editura Tip Naste, Pitesti, 2001

[3]Alexandru Ene, Cosmin Stirbu, Retele neuronale. Teorie si aplicatii in limbajul Java. Editura Universitatii din Pitesti, 2008