NATURAL GAS CONSUMPTION PREDICTION WITH FEED FORWARD NEURAL NETWORKS

Alexandru ENE[1], Cosmin STIRBU[2]
University of Pitesti
Department of Electronics, Communications and Computers
[1]alexandru.ene@upit.ro, [2]cosmin.stirbu@upit.ro

Keywords: feed forward neural network, gas consumption prediction, nonlinear function approximation

Abstract: *Feed forward neural networks are typically used in pattern recognition applications. They are also used for the approximation of nonlinear functions. In this paper we use a feed forward neural network for the prediction of the monthly consumption of gas in a house. The neural network is trained using the data about the monthly consumption of gas in the previous three years. We use as N inputs in the network the consumption of N previous months, in order to calculate the output of the network, the consumption of gas for the next month. The simulations are done using Java language. In the last section of this paper are presented the experimental results, including the approximation error*

1. FEED FORWARD NEURAL NETWORKS

Feed forward neural network are excellent tools for solving complicated problems like pattern recognition [1].

They learn to recognize patterns using examples, so they learn like human beings do: through examples.

A feed forward neural network consists from layers of interconnected artificial neurons. Every connection has associated a real number, positive or negative, called weight. Typically, in pattern recognition applications or in approximation of non-linear functions, the network has an input layer, one or more middle layers (these are called hidden layers), and an output layer.

The first model of artificial neuron was the McCulloch – Pitts neuron. It is a device that has binary inputs, each having a certain weight (a real value) associated to it, and a single binary output. We use the following notations:

$x_1,\dots,x_n$. – inputs 0 or 1

$w_1\dots w_n$. - weights.

θ - Threshold (usually is 0).

There is the activation function, which has the following formula:

$$a = w_1 \cdot x_1 + w_2 \cdot x_2 + \cdots + w_n \cdot x_n$$

or, with a simplified notation:

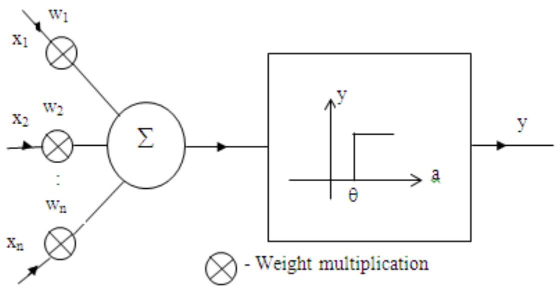
$$a = \sum_{i=1}^{n} x_i w_i$$

The output (y) of the McCulloch-Pitts neuron is obtained with the formula:

$$y = \begin{cases} 1 \text{ if } a \ge \theta \\ 0 \text{ if } a < \theta \end{cases}$$

Figure 1 shows a McCulloch-Pitts neuron with n binary inputs.

Figure 1. Artificial neuron McCulloch-Pitts



But with McCulloch- Pitts neurons, small variations of weights lead to big variations of outputs.

In order to obtain with small variations of weights, also small variations of outputs, the McCulloch-Pitts neuron has been replaced in the structure of feed forward neural networks, by another type of artificial neuron: sigmoid neuron. Its transfer function is plotted in figure 2, where y is the output of the neuron and x is the activation computed as a weighted sum of its inputs.

Figure 2. Sigmoidal function

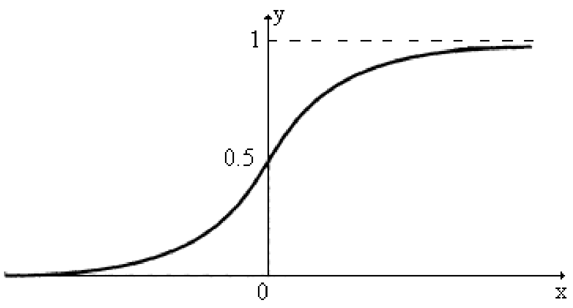
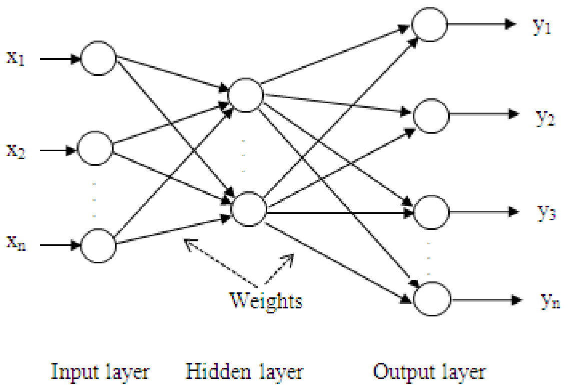


Figure 3. Feed forward neural network 3-2-4



The transfer function has the formula:

$$y = \frac{1}{1+e^{-x}}$$

In a feed forward neural network each neuron from a layer is connected with all the neurons from the next layer.

In the same layer the neurons are not interconnected with each other.

The next figure shows a feed forward neural network that has 3 input neurons, 2 neurons in the hidden layer and 4 neurons in the output layer. We use the notation for this network: 3-2-4 .

A feed forward neural network can be hardware implemented or can be software simulated.

For the software simulation we can use a general programming language (in this paper we used Java language) or Matlab or we can use a dedicated environment.

The software simulation has two parts. The first part is the training of the network, in which typically it is used the back propagation algorithm. In this training phase, the training patterns that are used are very important. The learning phase is an iterative process in which all the weights from the network are computed. We can say that the specific knowledge of the neural network for solving a certain problem, is stored in its weights.

The second phase of the simulation, is the testing phase. In the testing phase we place some values on the inputs of the network, values that had not been used for the training of the network, and then we calculate the outputs from the network. First we calculate the output from each neuron from the hidden layer, and then we calculate the outputs of neurons from the next layer, etc. In the case of our application there is a single output to be calculated in the output layer.

2. TIMES SERIES PREDICTION

Neural networks are able to predict time series data, in our case the data represents the monthly gas consumption, in a house.

The data consist from a sequence of values recorded at successive moments of time (time series): x_1, x_2... x_n. The aim is to predict the value corresponding to the time n + 1. The basic idea is to assume that the current value x_i depends on N previous values: x_{i-1} x_{i-2}..., xi-N. Based on this assumption, a neural network could be trained to extract the association between any consecutive subsequence of N values and the next (N+1) value.

The prediction of data series [3] is based on the assumption that there are functional relationships between data series values of the past, present and future. The usual assumption is that the functional relationship is not completely deterministic, but contains some stochastic behavior. Often, especially in the case of time series, it is assumed that the behavior is not dominant deterministic, but is dominated by the stochastic component. The idea of applying neural networks originates in the observation that the functional relationships between time series values often are nonlinear.

In order to approximate nonlinear functions, neural networks are themselves excellent tools. In [2], the authors use a feed forward neural network to predict the failure rate of a radio reception system prediction, and this can be used for improving predictive maintenance

Since the 90s there were some books that discuss the application of neural networks in the context of time-series predictions. The books [4] and [5] are the first collections of articles centered on the analysis and practical application of neural networks in the prediction data series.

The purpose of the use of neural networks in this paper is that based on the history of monthly consumption of gas to predict the future values of monthly gas consumption.

Predicting monthly gas consumption in a house helps us to have a better management of expenses.

3. THE SOFTWARE SIMULATION

In order to estimate the gas consumption in a house for a certain month of the year, we implemented a feed forward neural network, with a single hidden layer, using a Java application.

We have a set of 35 experimental data (monthly natural gas consumption for a certain location), so we have the consumption for each month in the last three years. These data are recorded in a text file, each month consumption being in a different line. Here is an example of these experimental data:

```
1407.872
726.816
129.396
75.313
150.808
```

We use N input values to predict the next (N+1) value from this series.

So the structure of the feed forward neural network is: N – NH – 1, where N is the number of input neurons, NH is the number of hidden neurons. The network, predicting a single value has only a single output neuron. We experimented with different values of N and NH, as shown in the next paragraph, in order to obtain the best approximation for the natural gas consumption.

We use 30 values from the available natural gas consumption data, to train the network.

Because in the hidden layer and in the output layer are sigmoid neurons that have a nonlinear transfer function, and their output is in (0, 1) interval, is best for the input data to be scaled. Using our available data, we calculate firstly the maximum of monthly consumption of (max) and a minimum of monthly consumption of (min).

We use a linear scaling function in order get all input data from [min, max] interval to [A, B] interval. We experimented with different values of A and B (A=0.1 and B=0.9, etc.), as shown in the experimental results section of this paper. The scaled data that will be used for training and testing of the network are saved in a different text file scaledData.txt.

We obtained the scaled data (y) in the range [A,B] from the range [min,max] using the following formula::

y=(x*(b-a)+a*max-b*min)/(max-min)

A training pattern for this application has (N+1) values: N input values for the network (the network has N input neurons) , and another value that is the desired output that is expected from the network, when these N values are placed on its inputs.

The training patterns are automatically computed by the software program from the scaledData.txt file, in the following manner:

The first training pattern consists of data from lines 1,2,...N of scaledData.txt as inputs of the training pattern, and from line N+1 of this file as the value for the output of the first training pattern..

Second training pattern consists of data from lines 2,3,...N+1 of scaledData.txt as inputs of the training pattern, and from line N+2 of this file as the value for the output of the second training pattern.

The third training pattern consists of data from lines 3,4,...N+2 of scaledData.txt as inputs of the training pattern, and from line N+3 of this file as the value for the output of the second training pattern.

And so on.

We trained the network using the backpropagation algorithm, which for a feed forward neural network with N input neurons, NH hidden neurons and a single output neuron, is the following:

```
Initialize all the weights of the network with small random values, between (-0,5, +0.5)
 REPEAT
   for all training patterns execute
  begin
     foward propagate the current input pattern
     compute the error in the output neuron
     adjust all the weights
   end
   compute E, the total error of learning
 UNTIL E<E_MIN
```

4. THE TESTING RESULTS

The scaling interval [A, B] is important for the approximation error and also for the total number of epochs in which the network learns the training patterns. Experimentally the best results, for the available training data set, had been obtained for A=0.3 and B=0.7 (scaling interval [0.3, 0.7]).

A 3–5-1 neural network (3 input neurons, 5 hidden neurons and a single output neuron) , for a 5% global error for the whole set of training patterns, learns the patterns in about 200 epochs. It estimates the last value from the data set (when testing the trained neural network with new input data that were not used in the training set) with an approximation error of about 12% (which is not very good).. Approximately the same results are obtained with a 3–7-1 neural network and with a 3–2-1 neural network (this signifies that the number of hidden neurons does not have an important role in this application).

A more important influence on the approximation error, has the number of input neurons (how many previous consumption values are necessary to estimate the consumption for the next month.)

Best results had been obtained for a 6-7-1 feed forward neural network, and a total error for the learning of training set of 1%. The network learns all the training patterns in about 4000 epochs. It estimate the values that had not been used in the training set with an approximation error less than 5%, which is better.

Using the same architecture 5-7-1 and the same global error of 1%, but changing the scaling interval to A=0.1 and B=0.9, we do not obtain better results. The network converges much more slowly (in about 40000 epochs, and the approximation error is not better).

5. CONCLUSIONS

In this paper we used a feed forward neural network, to estimate the monthly gas consumption in a house. The network has only analog inputs (scaled data of gas consumption of some previous months).

The best approximation error that we obtained was less than 5%, for a 6-7-1 feed forward neural network.

The approximation error could be improved if we took into account other extra inputs for the network. For instance we could use 12 binary inputs in which to specify the month of the year in which the neural network computes the gas consumption. The estimation could be also improved if we had more data of gas consumption, available. This estimation could be used to improve the management of expenses in a home.

6. REFERENCES

[1] A. Ene, C.Stirbu,"Retele neuronale. Teorie si aplicatii in Java", Ed. Univ. din Pitesti, 2008.

[2] A. Ene, C.Stirbu,"A Java application for the failure rate prediction using feed forward neural networks ", ECAI 2016, International conference, Ploiesti, 2016

[3] Enăchescu, C., "Data Predictions Using Neural Networks", Proceedings of the International Conference on Knowledge Engineering, Principles and Techniques, KEPT2007, Cluj-Napoca (Romania), June 6-8, 2007, pp. 290-297.

[4] Weigend, A.S., Gershenfeld, N.A., "Time series prediction: Forecasting the future and understanding the past", Addison Wesley Reading MA, 1994.

[5] Rogers D.A., Vemuri, V., „Artificial Neural Networks" IEEE Computer Society, 1993.