DOI: 10.5769/C2012010 or http://dx.doi.org/10.5769/C2012010

Neural Network Predictor for Fraud Detection: A Study Case for the Federal Patrimony Department

Antonio Manuel Rubio Serrano(1,2), João Paulo Carvalho Lustosa da Costa(1), Carlos Henrique Cardonha(3), Ararigleno Almeida Fernandes(1) and Rafael Timóteo de Sousa Júnior(1)
(1) Departamento de Engenharia Elétrica
Universidade de Brasília - UnB
Brasília – DF (BRAZIL)
{jpdacosta, desousa}@unb.br, ararigleno@gmail.com
(2) Universitat Politècnica de Catalunya - UPC
Castelldefels, Barcelona, SPAIN
toni.rubio.serrano@gmail.com
(3)IBM Research
São Paulo, BRAZIL
carloscardonha@br.ibm.com

Abstract--- Fraud detection is necessary for any financial system. However, the way of committing frauds and also for detecting them have evolved considerably in the lasts years, mainly due the development of new technologies. Therefore, fraud detection via statistical schemes has become an important tool to reduce the chances of frauds.

In this paper, we present a study case applied to the tax collection per month of the Federal Patrimony Department (SPU). In this study case, we analyze some of the current methods for fraud detection, as Rule-Based Systems and Neural Networks classifiers, and propose the use of Neural Networks predictors for detecting fraud in time series data of the SPU.

I. Introduction

Fraud has been very common in our society, and affects private enterprises as well as public entities. However, in recent years, the development of new technologies has also provided criminals more sophisticated way to commit fraud and has required more advanced techniques to detect and prevent such events.

One way to detect fraud is applying statistical fraud detection schemes. In general, statistical methods can be classified as supervised or unsupervised. The supervised techniques require both fraudulent and nonfraudulent samples in order to construct models that allow classifying future behavior patterns. On the other hand, unsupervised methods simply seek those accounts, customers and so forth which are most dissimilar from the norm. One drawback of most of unsupervised techniques is that fraudsters adapt to new prevention and detection measures. In this sense, fraud detection needs to be adaptive and to evolve over time in order to gradually change their behavior over some period of time and to avoid spurious alarms [1].

Nowadays there are different types of solutions for facing fraud. Traditional statistical classification is still applied with a high detection probability [1,2]. For instance, rule-based methods are classifiers based on a set of conditions with the form *If{condition},Then{consequence}* that represents the behavior of the fraudulent actions. Examples of these algorithms are BAYES [1,3], FOIL [1,4] and RIPPER [1,5].

Also, Artificial Neural Networks (ANN) have been applied for fraud detection, mainly in the context of supervised classification. In [6], ANN is applied for credit card fraud detection.

In this paper, we propose the use of a neural network-based predictor to identify possible fraudulent patterns in the Federal Patrimony Department, in Portuguese Secretaria de Patrimônio da União (SPU). Our proposed method is based on the fact that the artificial neural network (ANN) can be used in the recognition of statistical characteristics on a time series and make predictions.

This paper is divided into six sections. In Section II, we review some methods applied for fraud detection and justify the proposed method based on ANN predictors. In Section III, the data model used in this study case is presented, while Section IV presents an overview about ANN. In Section V we show some experimental results and in Section VI we draw our conclusions.

II. Application of Neural Networks Predictor for Fraud Detection

A. *Current approach for Fraud Detection*

The detection of fraud and other misuses, such as computer intrusion, irregularities in telecommunications systems, and credit car theft, have been pursued since long time ago. But specially in the lasts years and even until now, new techniques have been developed according with the development and growth of new Information Technologies, which allows new ways for committing fraud and also new ways for detecting it.

Most current approaches to the process of detecting fraud and misuse utilize some form of rule-based analysis. Rule-Based analysis relies on sets of predefined rules that are provided by an administrator, automatically created by the system, or both. Expert systems are the most common form of rule-based intrusion detection approaches. Expert systems permit the incorporation of an extensive amount of human experience into a computer application in order to identify deviant activities. However, these systems have some important limitations. The most important is that the frauds evolutes and the system should be constantly updated and evaluated. Unfortunately, expert systems require frequent updates to remain current.

In order to solve the limitations of expert systems, the use of neural network has been extended in the lasts years. The main advantage of these systems is the ability to represent variables with a complex and nonlinear behavior with a very high accuracy. Of course, another very important point is the fact that an ANN learns adaptively and, therefore, it is a general solution for different kind of data if trained.

One major challenge faced by classical ANN is to decide the optimal network structure. Namely, performance varies significantly depending on the relation between the dataset and on the structure of the network. For solving these problems, an adaptive approach, where the network is able to reconfigure itself according to the performance obtained with a certain dataset, is crucial. Some examples of this adaptive approaches are the Generalized Adaptive Neural Network Architecture (GANNA) and Adaptive Logic Network (ALN) methods [7].

Traditionally the use of ANN for fraud detection is done using the generated network as a classifier. With this approach, the network is trained with examples of fraudulent and non-fraudulent actions. Once trained, the ANN is able to classify new data as fraudulent or non-fraudulent activities. However, some difficulties can appear. First a considerable imbalance between the nonfraudulent and the fraudulent samples should exist. Otherwise, this should be compensated be some preprocessing [6]. Moreover, the network should be trained with a certain amount of samples and this training may take a prohibitive amount of time [8].

B. Neural Network predictor for Fraud Detection

In this paper, we propose the use of a Artificial Neural Network (ANN) predictor, instead of a Neural Network classifier, for fraud detection. In the case of an ANN classifier, the ANN is used to classify an input into a group from a set of predefined groups (for example, fraudulent and nonfraudulent). In the case of ANN predictor, the ANN returns the prediction of a certain input data. The advantages and the limitations of both systems are exemplified through the study case presented below, that uses the tax collection data of the SPU.

For our study case, we use data from the SPU starting from 2005 until 2010 and divided into months. The simulations try to detect if 2010 is a fraudulent year or not based on the predicted data.

For our data, the traditional implementation of an ANN classifier for fraud detection finds two important difficulties: the low number of samples and the increasing tendency of the time series.

We propose the use of ANN prediction algorithms applied to fraud detection in time series data. Our ANN makes the prediction and the real results of this year are compared with the prediction. If the results presents a great difference with the prediction it will mean that this sample should be deeply investigated. As shown in our simulations results, the proposed methodology based on ANN offers a very high accuracy for predicting time series.

In the following section we explain the data model used in the study case presented in this paper.

III. Data model

The data model shown in Fig. 1 considered in this paper is related to the tax collection of the SPU. To model this data, we assume a random variable with the costs of each month, where for $n = 1, \dots, 12$ indicates the months of the first year, for $n = 13, \dots, 24$ indicates the months of the second year and so on.

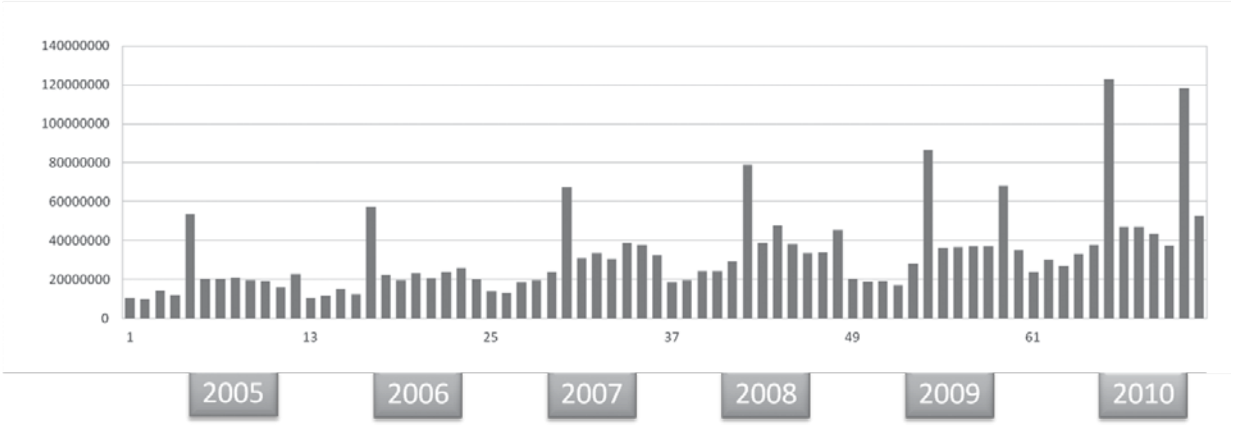

Figure 1. Collection per month in SPU.

We consider the months between the years 2005 until 2009 for estimating the parameters of the artificial neural networks (ANN). Therefore, we refer to these years as training set and our objective in this paper is to predict the data of both data models in 2010 by applying ANN.

IV. Proposed Artificial Neural Networks Predictor with cost functions

An artificial neural network (ANN) is a computational model that is loosely based on the neuron cell structure of the biological nervous system. In most cases, an ANN is an adaptive system that changes its structure based on external or internal information flowing through the network during the learning phase. ANN has a potential for intelligent systems because they can learn and adapt, they can approximate nonlinear functions, and they naturally model multivariable systems [9].

An ANN is composed of *neurons*, explained later in detail. These neurons are grouped in layers, where the last one is called the *output layer*, and the previous ones are called *hidden layers*. The connection of the neurons can be done in different ways, originating different kinds of ANN. In this work, we will focus on Feed-Forward Networks, which are the most widely used for time-series prediction [10][11][12].

In Feed-Forward Networks, the information travels only in one direction, from the input to the output, without any feedback nor between neurons of the same layer. The most commonly Feed-Forward Network used is the Multilayer Perceptron (MLP), where all the neurons of a layer are connected to all the neurons of the following layer. The Figure 2 shows an example of MLP. The network has three inputs, four units in the first layer, which is called hidden layer, and one unit in the second layer, which is called the output layer.

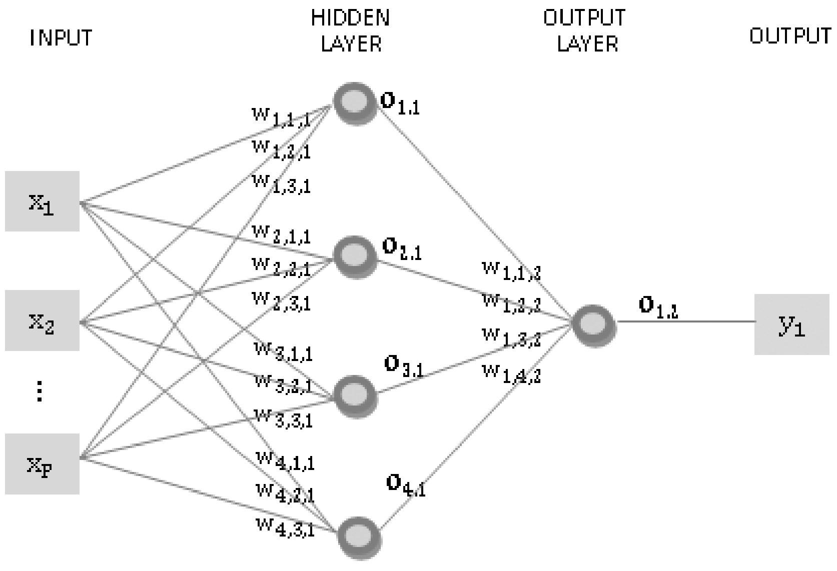

Figure 2. A two-layer feed-forward Neural Network.

According to Figure 2, the network receives an input vector. Each element on the input vector is connected to all the neurons in the hidden layer, and the outputs of this layer are connected to the neurons on the output layer. In the each layer, each neuron assigns a certain weight to each input. According to that, we can define the weight matrix W_c of size $\mathbb{R}^{C \times l}$ containing all the weights for every layer connection.

$$W_{c,p,l} = \begin{bmatrix} w_{1,1,l} & w_{1,2,l} & \cdots & w_{1,P,l} \\ w_{2,1,l} & w_{2,2,l} & \cdots & w_{2,P,l} \\ w_{C,1,l} & w_{C,2,l} & \cdots & w_{C,P,l} \end{bmatrix}, \tag{1}$$

where l indicates the current layer (from 0 to L starting from the input vector, that can be considered the layer 0, where L is the number of layers), the column indices p on the elements of matrix W_c indicate the source neuron in the previous layer l-1, and the row indices c indicate which is the destination for that weight in the current layer l. Thus, the indices in w is referred to the weight connecting the second element of the previous layer, which in the example above is the input vector, with the first element of the actual layer l.

Taking into account the inputs and the weights, each neuron generates an output. This output will also depend on the activation function of the neuron *h()*. The Figure 3 shows a neuron model in detail. The output produced by the *c*-th neuron of the *l*-th layer, $o_{c,l}$ can be calculated as

$$o_{c,l}(k) = h\big(z_{c,l}(k)\big), \tag{2a}$$

where

$$z_{c,l}(k) = \sum_{p=1}^{P} o_{p,l-1}(k)\, w_{c,p,l}(k) + b_{c,l}(k)\,, \tag{3}$$

and where k is referred to the iteration, $o_{p,l-1}$ is the output generated by the neuron p of the previous layer l-1, $w_{c,p,l}$ is the weight that connects the neuron p of the previous layer with the neuron *c* in the current layer and $b_{c,l}$ is the bias for the neuron *c* which is used for regulate the level of the output. Equation (3) can be written in the matrix form as

$$o_{c,l}(k) = h\left(W_{c,l}\,O_{p,l-1} + b_{c,l}\right), \tag{2b}$$

where $o_{c,l}$ is represented by y_c in the case of the output layer and the output of the layer is the output of the network. The Figure 3 shows how $w_{c,p,l}$ and $o_{c,l}$ are used in the network.

According to (2a) and (2b), the output generated by a neuron will depend on its activation function. The neurons can have different activation function in the same network, but the more used are the sigmoid function for the hidden layers and the linear function for the output layers. This is also the configuration used in this study. We can define the activation functions h_{Output} for the output layer and h_{Hidden} for the hidden layers as

$$h_{Output}(z) = c_1 z + c_2 \tag{4a}$$

$$h_{Hidden}(z) = \text{sgm}(x) = \frac{2c_1}{1 + e^{-c_2 z}} - c_1 \tag{4b}$$

In both cases, and are constants. The derivatives of the previous equations are:

$$h'_{Output}(z) = c_1 \tag{5a}$$

$$h'_{Hidden} = \text{sgd}(z) = \frac{c_2}{2\,c_1}\left[\,c_1{}^2 - \text{sgm}^2(z)\right] \tag{5b}$$

In order to build an ANN, several neurons as shown in Figure 3 are interconnected.

Once the first output of the network is defined, the network should be able to learn in order to behave as expected. There are two ways for making the network learn: supervised or unsupervised learning.

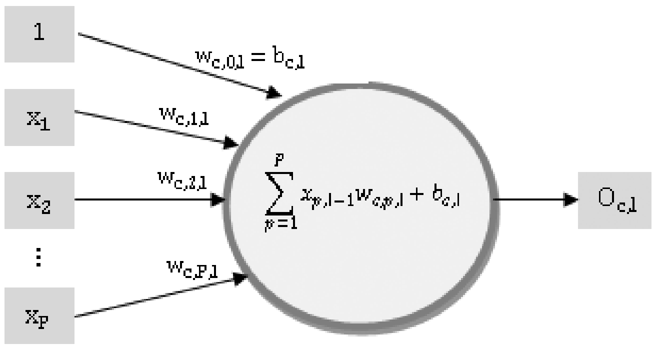

Figure 3. Schematic representation of a neuron.

In *Supervised Learning*, a training dataset with labeled responses is given to the classifier. The classifier is trained for obtaining 'best' results in the training dataset and we suppose that the system will have good generalization properties for non-trained inputs.

In *Unsupervised Learning,* the classifier is given a set of samples and we don't know the classes a-priori. The algorithm has to learn the classes in the dataset.

Also, inside these two groups, many different learning algorithms have been developed. In this study, the *backpropagation learning algorithm*, which is a specific implementation of supervised learning and the most useful for training an MLP, has been considered.

Backpropagation is a common method of training artificial neural networks so as to minimize the error at the output of the network. The term is an abbreviation for "backward propagation of errors". To make meaningful forecasts, the neural network has to be trained on an appropriate data series. Examples in the form of <input, output> pairs are extracted from the data series, where input and output are vectors equal in size to the number of network inputs and outputs, respectively. A detailed description of the Backpropagation algorithm can be found in [13].

Once the prediction is performed by the ANN, the Normalized Root Mean Square Error (NRMSE) and the Coefficient of Determination (COD) are considered in order to indicate if there is a fraud or not. the.

- A normalized Root Mean Square Error

$$NRMSE = \sqrt{\frac{1}{N}\frac{\sum_i (d_i - y_i)^2}{\sum_i (d_i)^2}} \tag{7}$$

- Coefficient of determination

$$COD = R^2 = 1 - \frac{\sum_i (x_i - y_i)^2}{\sum_i (x_i - \bar{x}_i)^2} \tag{8}$$

In (7) and (8), , for $i = 1, \dots, N,$ is referred to the actual or real values of the *i*th month, while are referred to the predicted values of the *i*th month. The optimum value for NRMSE is zero, while for the COD is one

V. Experimental results

In this section, the experimental results are presented. First, we search empirically the best ANN configuration, i.e. with best number of neurons and of layers, for our data. We find out that the Multi-Layer Perceptron of two layers, containing 8 neurons on the hidden layer and 1 neuron on the output layer, presents the smallest error.

In Figure 6, we apply (7) and (8) to compare the predicted and the actual data, and we obtain NRMSE = 4 % and COD = 0.76. Therefore the similarity is very high between the predicted and the actual data. Such similarity can be also easily visualized.

Once we confirm the good performance of the chosen ANN configuration, the real data is corrupted in order to simulate the fraudulent data in the year 2010. We corrupt the data by adding a 10 % error distributed over all the months of the year.

In Fig. 7, the real and predict are still very similar. The NRMSE = 6% and the COD = 0.86. Depending on the threshold level, this fraud could be detected. For instance, if the threshold for the NRMSE is 5 %, the fraud is detected, if the threshold is equal to 10 %, the fraud will not be detected.

In Fig. 8, the 10 % error is concentrated on the third month only, the fraud becomes visible. In this case, we obtain a NRMSE of 13% and a COD equal to 0.32.

In Fig. 9, we divide a 15 % error through four months randomly. First, we consider a difference of 15% on the overall year, with the increments divided in 4 months. The NRMSE = 16,8% and the COD = 0.33. Due to the values of NRMSE and COD, most probably frauds have taken place in this year.

In Fig. 10, we consider that an error due to the fraud leading to a difference of 20 % distributed randomly in four months.

For the scenario in Fig. 10, we have obtained 20% of NRMSE and a COD equal to 0.62. The NRMSE indicates that the results on this year should be investigated.

Now we compare our solution to the traditional ANN predictor. Instead of predicting the data, the ANN gives an hard output, which can be '1' for fraudulent sample and '0' for nonfraudulent samples. Due to the small number of samples available as shown in Fig. 1, the traditional ANN has not been able to correctly classify fraudulent and nonfraudulent samples. Therefore, for small number of data, our proposed scheme should be applied for fraud detection.

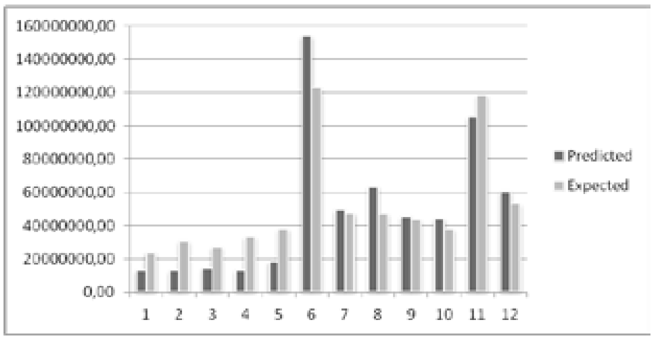

Figure 6. Predicted and Expected values using an 8-1 MLP. NRMSE = 4% and COD = 0.76.

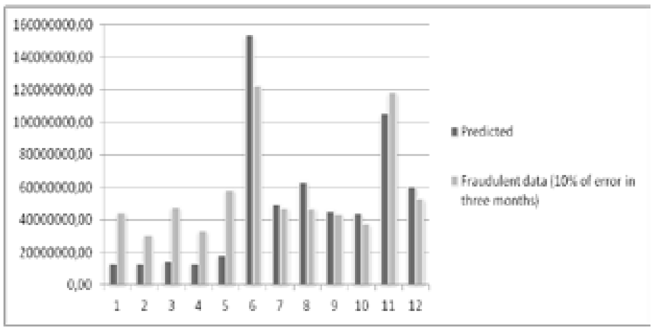

Figure 7. 10% of error distributed in 3 months. NRMSE = 6% and COD=0.86.

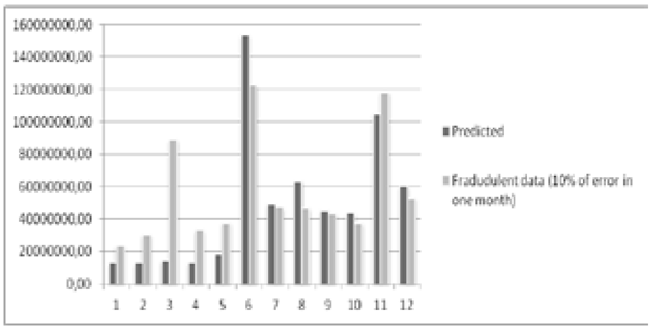

Figure 8. 10% of error in 1 month. NRMSE = 13% and COD=0.32.

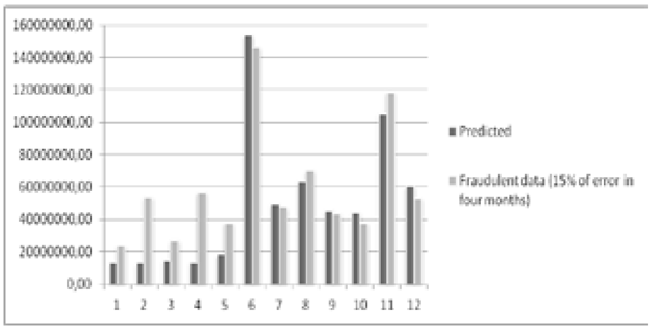

Figure 9. 15% of error distributed in 4 months. NRMSE=16.8 and COD=0.33.

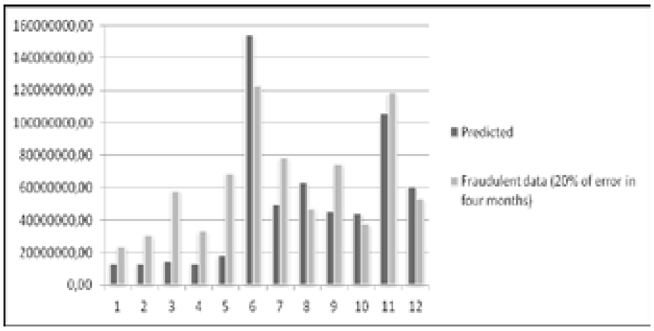

Figure 10. 20% of error distributed in 4 months. NRMSE=20% and COD=0.62.

VI. Conclusions

In this paper, we have proposed the application of ANN for time series prediction together with NRMSE and COD in order to detect frauds in financial systems.

Our proposed solution is particularly interesting when only a small number of samples are available. Since the traditional solution does not work for such scenarios, our proposed scheme should be applied. Therefore, our method based on ANN predictor should be considered as an alternative to ANN classifiers when the data is a time series which is evolving with time and when the number of sample for training the network is small.

To validate our study, we have considered data from the tax collection of the SPU.

References

[1] R.J. Bolton and D.J. Hand, "Statistical fraud detection: A Review", Statistical Science, vol. 17, No. 3 (2002)

[2] Hand, D. J. (1981). Discrimination and Classification. Wiley, Chichester

[3] Clark, P. and Niblett, T. (1989). The CN2 induction algorithm. Machine Learning 3 261-285.

[4] Quinlan, J. R. (1990). Learning logical definitions from rela-tions. Machine Learning 5 239-266.

[5] Cohen, W. (1995). Fast effective rule induction. In Proceedings of the 12th International Conference on Machine Learning 115- 123. Morgan Kaufmann, Palo Alto, CA

[6] Dorronsoro, Ginel, Sanchez and Cruz (1997)

[7] K. Fanning, K.O. Cogger, R. Srivastava, "Detection of Management Fraud: A Neural Network Approach", IEEE (1995)

[8] J. Cannady, "Artificial Neural Networks for Misuse Detection", Nova Southeastern University

[9] Jantzen J., "Introduction to Perceptron Networks", Technical University of Denmark (1998).

[10] Frank, R. J.; Davey, N. and Hunt, S.P., Time Series prediction and Neural Networks. University of Hertfordshire, UK.

[11] Patterson D W, Chan K H, Tan C M., Time Series Forecasting with neural nets: a comparative study.. Proc. the international onference on neural network applications to signal processing. NNASP 1993 Singapore pp 269-274., 1993.

[12] T. Koskela et al., Time Series Prediction with Multilayer Perceptron,

[13] S. Haykin, Neural Networks: A Comprehensive Foundation. Prentice Hall, Englewood Cli_s, NJ, 2nd edition, 1999.

[14] Drossu, R. and Obradovic, Z., "Rapid design of Neural Networks for Time Series prediction". Washington State University (1996).

[15] Faraway, J. and Chatfield, C., "Time series forecasting with neural networks: a comparative study using airline data", In: Royal Statistical Society (1996).

[16] Plummer, E. A., "Time Series forecasting with feed-forward neural networks: guidelines and limitations". University of Wyoming (2000).

[17] Luhn, H.P., "A Business Intelligence System". IBM Journal (1958) .