Optimizations in Time Series Clustering and Prediction

Ventsislav Nikolov

Abstract: *In this paper a combination of time series clustering and prediction is considered. Both clustering and prediction are done by neural networks with supervised and unsupervised learning respectively. Some optimizations of the clustering procedure are proposed for software implementation. Prediction results for univariate time series are shown and analysed.*

Key words: *Neural Networks, Prediction, Self-organizing map, Time series, Forecasting*

INTRODUCTION

Time series prediction is a problem of present interest in the computer sciences. It is important because even if the predictions do not come true, they give valuable information about the underlying process behaviour. The time series prediction facilitates the future analysis of a given system that could considerably reduce some expences. Prediction of sun spots, currency exchange rates, economic indices, number of incoming requests to a given server, trajectory of a given moving object are only part of the most interesting applications in time series prediction.
Connectionist researchers also show interest in possibilities to combine clustering and prediction. They have used different approaches in the form of network committees, agent teams, stacked generalization, local models, etc. [9]. In [2] an approach using self-organizing map and local autoregressive (AR) models is proposed. It is well-known that the neural network can be regarded as a non-linear autoregressive model. That is why it is a more powerful tool compared to the linear autoregressive models [4]. In this paper some improvements of the local models by using neural networks instead of autoregressive models are proposed. The problem of time series prediction is solved by decomposition of the input data into many parts and separate non-linear models are used for each part. Moreover, some performance optimizations are proposed to the general self-organizing map clustering algorithm.

TIME SERIES PREDICTION

In the univariate time series prediction problem, the input data consists of a sequence of values and the next values should be estimated by analysis of the previous observations. Additional factors are not involved in this process. When the predicted values are calculated, often confidence levels are shown along with the results.
Some of the most commonly used methods for time series prediction are [3]: averages, autoregressive methods, decomposition, exponential smoothing, trend extrapolation, neural networks, regression, etc.

TIME SERIES PREDICTION BY A NEURAL NETWORK

In this paper a method of prediction by neural networks fed with subseries obtained from the initial time series is considered. The subseries are obtained through a sliding window that traverses the time series and thus generates input-output training patterns for the

network [11]. Most of the regression models use this technique.

An important matter is the determination of the window size. There are two main techniques for solving this problem. Most often autocorrelation and partial autocorrelation functions are analysed for this purpose. Sometimes because of the complexity of their structure brute force searching can be done. This search starts with a window containing a single value. Then model building is performed followed by test prediction and error calculation. After that the size of the window is increased by one element and the process is repeated until the size of the window is equal to the size of initial time series or satisfactory error is reached.
In the prediction stage the sliding window is situated at the end of the series to obtain input values to feed the neural network. The output of the neural network is considered as the first predicted value. Then the first prediction is considered as a part of the initial time series and the window is moved again in order to predict the second value. This procedure (recursive prediction) is repeated until the desired time horizon is reached. This process is graphically shown on fig.1. The prediction begins when the window is moved N-p steps, where N is the initial time series length, p is the window size (number of values shown in the grey rectangle).

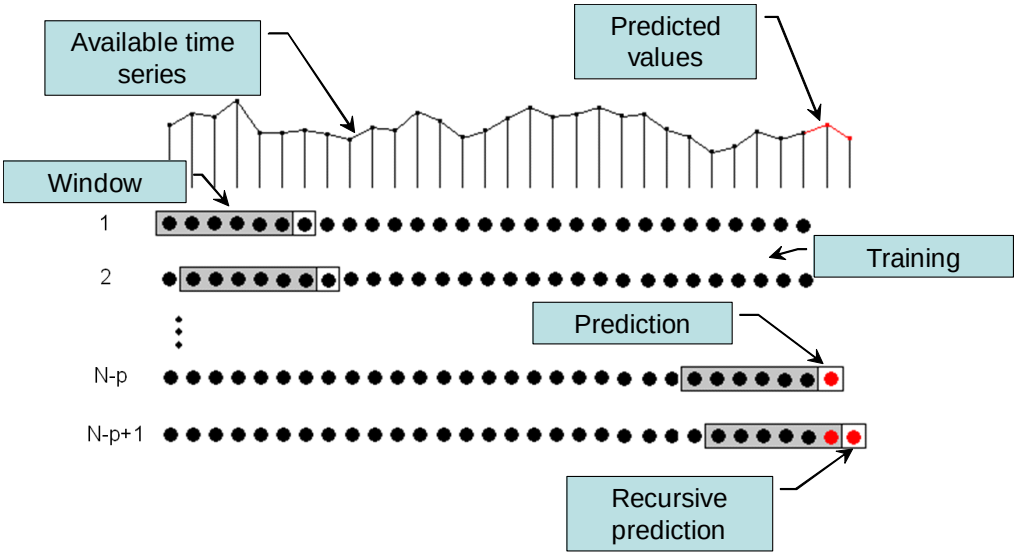

Figure 1. Windowing approach

Some pre-processing operations should be done before the training process. The subjective of the pre-processings is to transform the initial series into a form needed for the neural network to work properly. This stage consists mainly of the following operations: normalization, different techniques for trend removing and sometimes seasonality removing. They can be used in different order according to the time series characteristics.

CLUSTERING AND PREDICTION

An interesting and promising approach for time series prediction is clustering to be used as a pre-processing of the initial data [2, 7, 9]. Some advantages of this approach are: the model for every cluster can be built and trained in an independent computer system that leads to improved parallel computations; homogeneous training regardless of

the time series length; possibility for detailed analysis in some specific areas of the time series; better interpretability; an independent distribution of the testing patterns along the series. Moreover, this approach allows usage of combination of different model types for prediction. For example, a feed forward neural network and an autoregressive model can be used for two different parts of the input data. In addition, every part can be similarly separated into additional local models and hierarchical structure to be employed. This allows much more flexible modelling compared to the global models. In [8] are also described some other advantages of the local models. Here an approach is analysed in which the initial set of vectors obtained by windowing is clustered in homogeneous subsets. An example of separation of the vectors into nine clusters is shown on the left side of fig.2. For every such cluster a separate neural network is built and trained.

In the current solution a self-organizing map is used for clustering because it allows assigning of different priorities for the training patterns. It is reasonable to assign a higher priority to the last elements in the time series. The usage of self-organizing map allows easy modification of the general training algorithm for this purpose [1].

Some clusters can be empty, or they may have a lot of elements (patterns). That is why the number of patterns in every cluster should be controlled by a threshold value for the minimal number of elements. In the current solution, first the number r of elements in every cluster, as if they were equally distributed, is calculated (1), where n is the number of all training patterns, c is the number of clusters. After that the real number of patterns ω_i in cluster i (2) is divided by r and if the result φ_i is smaller than the threshold then this cluster must be merged with the nearest cluster j according, for example, to the Euclidean distance [10, 6].

$$r = \frac{n}{c} \tag{1}$$

$$\varphi_i = \frac{\omega_i}{r} \tag{2}$$

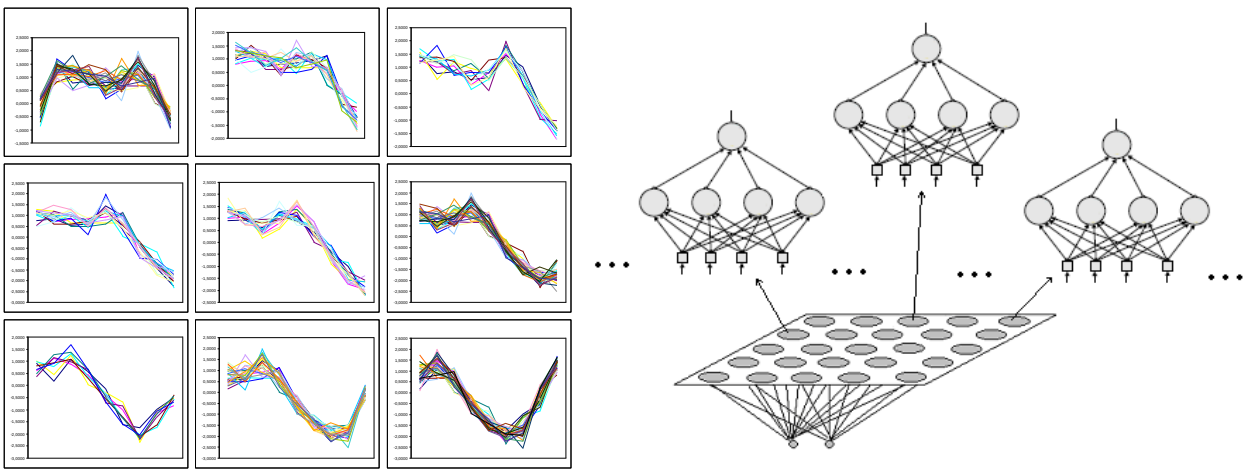

Figure 2. Clusters of vectors (left). For every output neuron (cluster) in the self-organizing map a multilayer perceptron is created (right).

Another important matter is the separation of the initial data into training and test subsets. It is well known that the error calculated for the training patterns is not always a good indicator of the neural network generalization ability – fig.3. Such data separation must be done for every cluster.

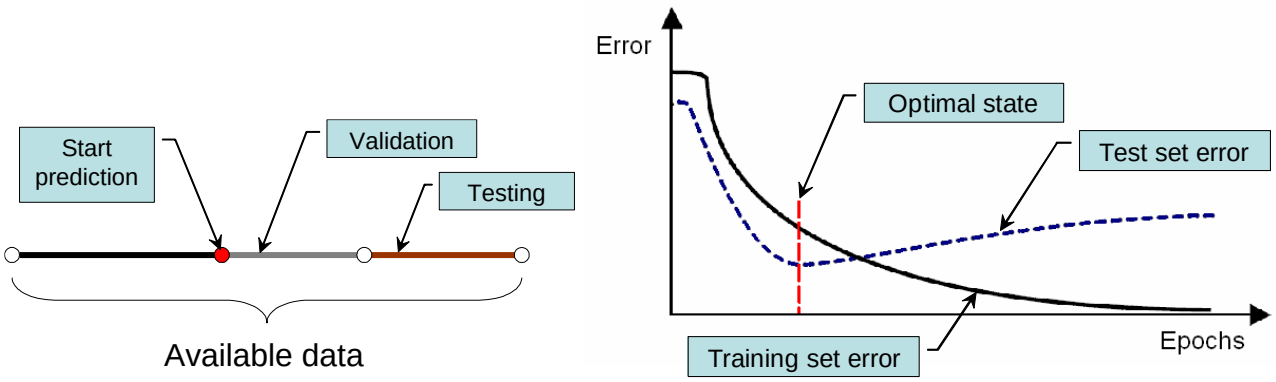

Figure 3. Training and test error.

The separation of the data into training and test subsets can also be performed before clustering, though this may cause irregularity in the clusters. As a result, any cluster may have training and testing patterns, but other may have only training or even only testing patterns. To overcome this problem, first the clustering can be done for all (training and testing) patterns and then the relative number of testing patterns in every cluster should be determined. If the number of testing patterns is small or there are no testing patterns, then the neural network corresponding to this cluster cannot be trained with out-of-sample validation. If there are only testing patterns for a given cluster then it should be merged with the closest one. An alternative approach is only training patterns to be clustered and used for neural network training.

CLUSTERING OPTIMIZATIONS

Several optimizations of the general self-organizing map training algorithm [5] are proposed in the solution. The output layer of the self-organizing map has a fixed size that in not changed over the training. This allows accelerating the stage of searching neurons within the neighbourhood radius. If for example the output grid of the self-organizing map is rectangular with four neurons in x-axis, three neurons in y-axis (table 1) and every neuron is a square with size 10 units then sorted distances from element (w) with coordinates (2, 2) to the other 11 units are shown in table 2.

Table 1. Grid of output neurons in SOM.

d_{11}	d_{12}	d_{13}	d_{14}
d_{21}	**w**	d_{23}	d_{24}
d_{31}	d_{32}	d_{33}	d_{34}

Table 2. List of pairs “neuron - distance”

d_{12}	d_{21}	d_{23}	d_{32}	d_{11}	d_{13}	d_{31}	d_{33}	d_{24}	d_{14}	d_{34}
10	10	10	10	14	14	14	14	20	20	23

The distances between the neurons in the grid do not change and this fact can be used by calculating distances between every two neurons beforehand. Then a set of pairs for every output neuron is created. The first value in a pair is an index of a neuron in the grid and the second value is the distance to this neuron. The indices of neurons in the grid are set ascending from left to right and from top to bottom. The pairs should be sorted regarding to their second value (distances). In this way the set of first values in the pairs of a given neuron is a list of indices sorted by nearness to the given neuron. According to the original Kohonen algorithm when the best matching neuron is found, a modification should be

done of its weights and the weights of its neighbours in the neighbourhood radius. Using the procedure described here allows traversing of all output neurons in the grid to be reduced. Only the units in the neighbourhood radius are traversed by the following cycle:

```
for(i=0; pair[i].second_value < neighbourhood_radius; i++)
{
  neighbourhood_neuron_index = pair[i].first_value;
  adjust_weights_of_neighbourhood_neuron;
}
```

Moreover, another accelerating procedure can be applied when the neighbourhood radius is equal to the size of one neuron. In this case the new winning neuron is just the old winner (in about 95% of the epochs) because every input pattern modifies only its winner's weights toward pattern values. Using a simple boolean check and an array that shows where the pattern has been mapped in the previous epoch can reduce operations needed to find the best matching neuron.

```
if(neighbourhood_radius < neuron_size)
{
  winner_neuron_index = repr_position[current_pattern]
}
```

Every element of the array repr_position is an index (for indexing in the Kohonen grid) of the winning neuron for training pattern number equal to the index in the repr_position array. For example, if training pattern number one has had a winning neuron number 2 (that is d_{12} in table 1) in the last epoch then, with zero based indices in the array, the value of repr_position[0] is 2 (fig.4).

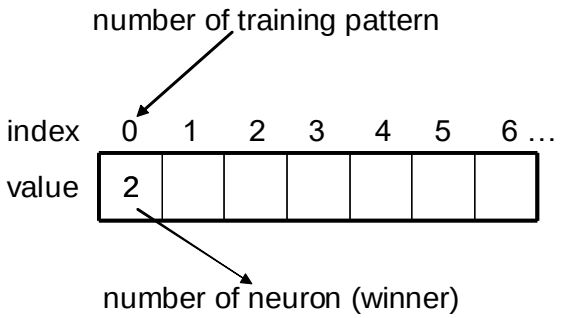

Figure 4. Array for saving the position of training patterns.

According to the performed tests, the two optimizations described above accelerate the general self-organizing map algorithm by 0 - 30% according to the number of input patterns and the number of output neurons.

EXPERIMENTAL RESULTS

The results of prediction of the time series for airline passenger data can be seen on fig.5. The prediction with clustering is shown with root mean squared error 113,46 and the prediction without clustering is with error 118,29.

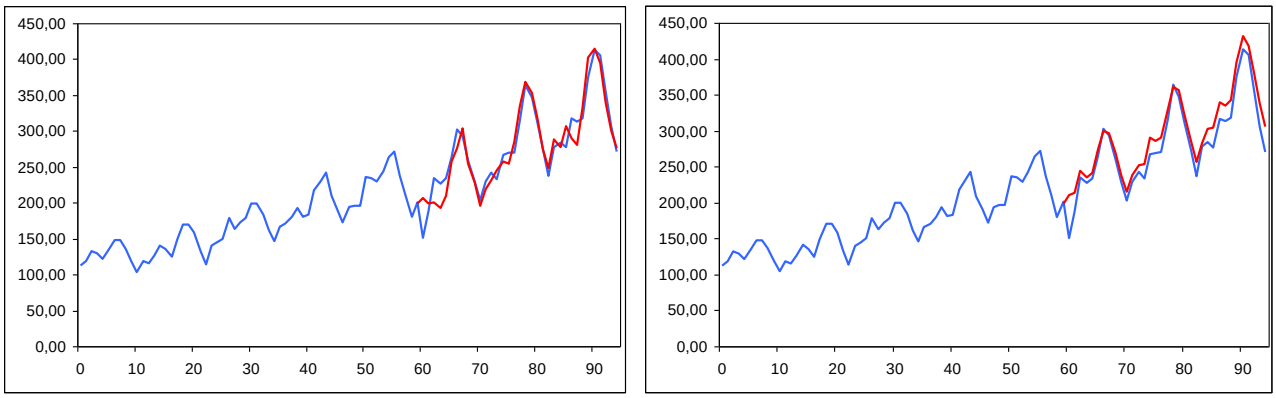

Figure 5. Prediction of the last 30 values with clustering (left) and without clustering (right)

Four clusters are used for the prediction with clustering. Data pre-processing (trend removing, data normalization etc.) are performed before clustering. The multilayer perceptrons work directly with the data obtained from clusters. The following steps are performed after data pre-processing and clustering.

1) Training:
   - For every cluster a multilayer perceptron is created and trained.
   - Individual settings are used for every multilayer perceptron (adaptive/non-adaptive learning rate, momentum factor, number of hidden neurons etc.).
2) Prediction:
   - The vector obtained by the last data window is fed to the self-organizing map and the winning neuron (and respectively cluster) is determined.
   - The vector is fed to the multilayer perceptron corresponding to the determined cluster and a prediction value is generated.
   - The procedure is repeated, and recursive prediction is performed.

Every cluster comprises both training and testing patterns. For every cluster the training patters are selected to be the last 10% of the data. Every multilayer perceptron created for a cluster is trained as if it should learn the whole series. Moreover, the training patterns in every cluster can be shuffled in every epoch.

CONCLUSIONS AND FUTURE WORK

There is a variety of prediction methods that can be used in combination with the clustering of input-output training patterns obtained by the sliding window. They can be easier adjusted using the homogeneous structure of the cluster data. For example, ARIMA, neural network, even simple regression can be used similarly to the described approach. If the time series has too many observations, then a more complex structure of different hierarchical local models can also be investigated. Another direction for further development can also be investigation of approaches for assigning of different priorities of the training patterns according to their position in the initial time series.

REFERENCES

[1] Antonov, A. Nikolov, V. 2009. Decision Making System for Clustering of Spread Curves. International conference Automatics and Informatics'09 Bulgaria, Sofia 29.09 - 4.14.2009, V-9 - V-12.

[2] Baretto, G. Time Series Prediction with the Self-Organizing Map: A Review. Department of Teleinformatics Engineering, Federal University of Cear´a Av. Mister Hull.

[3] Crone S. Business Forecasting with Artificial Neural Networks. IBF Tutorial – Institute of Business Forecasting, Boston, 2004.

[4] Dave Touretzky. Neural Networks for Time Series Prediction. Course 15-486/782: Artificial Neural Networks Fall, 2006.

[5] Kohonen, T. Self-Organizing Maps. Springer, 2001.

[6] Kovacheva, Ts. Cluster Analysis. Information technologies and control, Sofia, No.3, 2004, 24-30.

[7] Nikolov, V. Clustering and Prediction of Risk Spread Curves. CompSysTech'09, Ruse, 2009.

[8] Steve Lawrence, Ah Chung Tsoi, Andrew D. Back. Function Approximation with Neural Networks and Local Methods: Bias, Variance and Smoothness. Australian Conference on Neural Networks, ACNN'96. Australian National University, pp.16-21, 1996.]

[9] Thome, A. Tenorio, M. A Selective Committee Architecture for Time Series Prediction and Pattern Classification. Purdue University, 1993.

[10] Xu, R. Wunsch, D. Clustering (IEEE Press Series on Computational Intelligence), 2009.

[11] Zhang, G. Neural Networks in Business Forecasting. Idea Group Publishing, 2004.

ABOUT THE AUTHOR

Ventsislav Nikolov
Senior Software Developer
Eurorisk Systems Ltd.
31, General Kiselov Str., 9002 Varna, Bulgaria
E-mail: vnikolov at eurorisksystems dot com