Integration of Neural Networks and Expert Systems for Time Series Prediction

Ventsislav Nikolov, Valeri Bogdanov

Abstract: *In this paper an approach for development of univariate time series prediction library in Java and its integration with an existing CLIPS related system is presented. A backpropagation neural network based forecaster is provided in the current version of the library. The connection between Java and CLIPS is realized by Java Native Interface (JNI) and CLIPS user defined functions. The expert system is used to construct the object structure, configure and fit the prediction model, thus reducing user expertise requirements and allowing for complete automation.*

Key words: *Neural Network, Expert System, Rule Based System, Time Series, Prediction, Java Native Interface, JNI, CLIPS.*

INTRODUCTION

Most existing applications' approach to univariate time series prediction using neural networks require the user to fully control the process of neural network setup and training. This prevents their wide spread use because of the lack of user expertise and unacceptable operation times. To reduce or remove this deficiency we adopt another approach, namely the development of a hybrid system involving neural networks integration with an expert system. The combination of the symbolic and sub-symbolic systems could provide superior knowledge and performance. The interest of using them together is of rising interest in artificial intelligence [4, 6, 8]. The current view is that connectionist and symbolic approaches are complementary, not competing [11]. In our approach the integration of these two AI approaches is realized by providing abstraction of neural network creation, structure and training process to the rule based system. The hybrid prediction system is intended for use in financial applications for prediction and simulation of economical indicators, shares, etc.

REALIZATION

The main application is existing financial software developed in C++ and it is based on production CLIPS rules. All logical and user interface operations are described in CLIPS scripts used as models for specific economical calculations. Including Java neural network library allows network to be created, configured, trained and used for prediction. In addition to the ordinary neural network operations as creating, training and prediction, the library should also provide some possibilities that facilitate analyzing and pre-processing initial data as trend calculation and different ways to remove it, estimation of indicators for time series predictability as Hurst exponent, autocorrelation and partial autocorrelation functions calculation, etc.

The approach taken is to provide such functions to the main application as if they are part of the CLIPS language [2]. There are several problems related to this task. First, there should be possibility to create, manipulate and delete a neural network as an individual object. So, there should also be provided a possibility to manipulate many parallel neural networks. For this task a Java adapter have to be included between the library and the main application. Second, JNI should be used in order to use Java functionality in the C++ application.

Finally, CLIPS user defined functions API should be used to describe available functions as CLIPS functions and thus they can be used as if they are part of the CLIPS language [3].

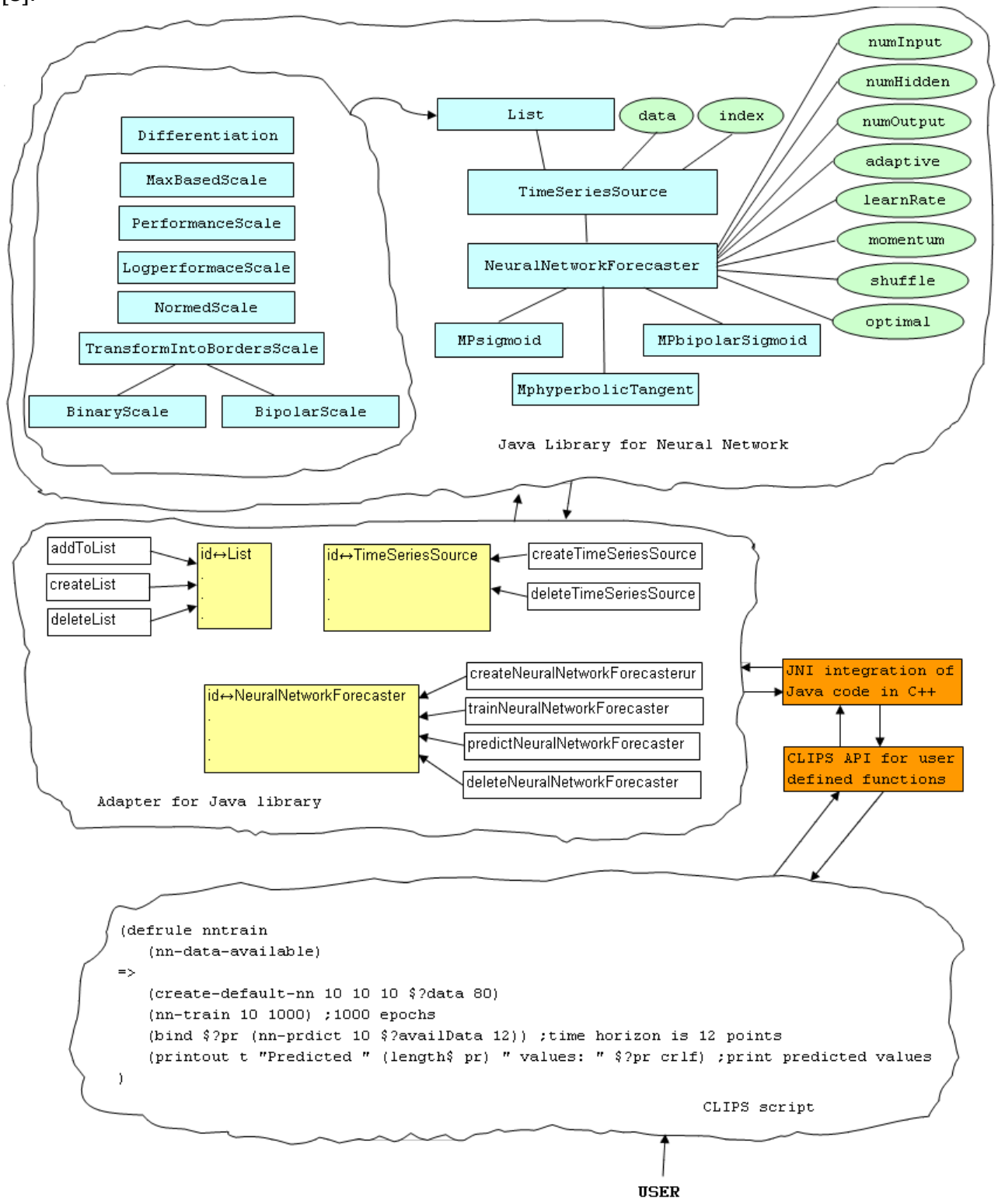

Figure.1 Architecture of the system

The information transferred from C++ to Java is needed to develop the neural network and to train it. Reverse data transfer is also needed from Java to C++ in order to propagate events (for example certain training epoch reached) during continues processing.
It is obvious that providing access from C++ to the Java library is easier task than to rewrite the whole Java library in C++. Moreover, the Java library can be further developed and used in Java applications. The logical structure of the objects in the Java library and the connection to the main application is shown on fig.1. The Java Library for Neural Network (at the top) should be built from CLIPS using the user defined functions provided in the CLIPS script. In order to build a proper and effective structure, the characteristics of the time series must be analyzed. Such functionality is made available in CLIPS through user defined functions referring to methods in the Java library. Nevertheless, time series processing is based on try and error approach which means that often some changes to the structure should be made and the process of building forecasting model should be repeated several times until satisfying model is found.
The neural network used in Neural Network Forecaster class is domain independent. Over it a domain dependent adapter is created satisfying common interface for time series forecasters.
The possibilities realized for pre-processing the existing time series are required for the neural network and a pre-processing list is used according to the series characteristics [7]. Realized methods for normalization are listed below.

Normed scale	$y_t = \frac{x_t - \mu}{\sigma}$	(1)
Differentiation	$y_t = x_t - x_{t-1}$	(2)
Performance scale	$y_t = \frac{x_t - x_{t-1}}{x_{t-1}}$	(3)
Log performance scale	$y_t = ln(\frac{x_t}{x_{t-1}})$	(4)
Max based scale	$y_t = \frac{x_t}{max(x)}$	(5)
Binary/Bipolar scale	$y_t = d + \frac{x_t - min(x)}{max(x) - min(x)} p$	(6)

where *d* is the lowest value of the series x, *p* is range (1 for normalization in (0 – 1) or 2 for normalization in (-1 – 1)).
Performance and log performance scale can be used only in series which do not contain zero as a value.
The other classes in the Java library are realized as a couple of modules that allow different extensions with new time series prediction approaches. It is designed as a programming interface which comprises several common methods used by the different type of forecasting approaches like AR, MA, ARMA, ARIMA [1, 5] and some other filters. A common technique used by many of these approaches is time series windowing. Also, certain pre-processing of the data is required. For example, removing trend and scaling. This functionality is separated in the pre-processing module (fig.2).
To allow the backward processing of series when prediction is made a stack is used. It contains the series before each processing. Some of the processings (like scaling) also require a state, which is initialized based on the data during the first processing.

Decoupling of pre-processing and the forecaster implementation allows for rather good extensibility of the Java library for both processing and forecast. While with the current implementation of the time series pre-processor if the time series does not contain all values, interpolation techniques can be used, alternative implementation can be made, skipping missing data (by not generating windows containing it), which can improve results for series with relatively small number of missing values and behaviour rather low value of Hurst exponent. Also, additional technique for integration of clustering into the training process [12] is taken into account when designing the library structure. Thus, it is possible to develop so called local prediction techniques or different “agents” that are many neural networks trained only by data in the corresponding cluster.

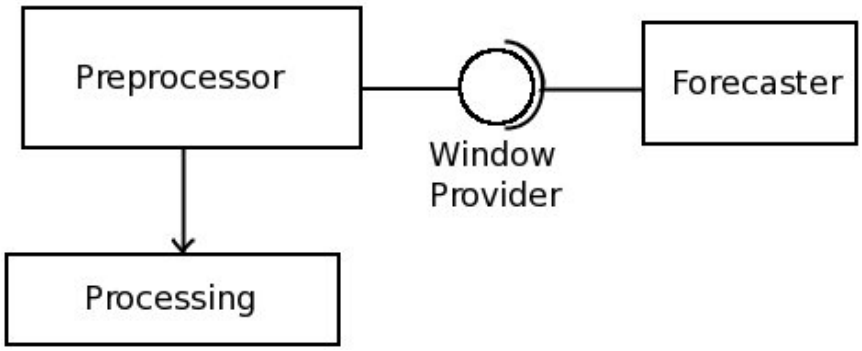

Figure 2. Modules of the system

Specific for the neural network based forecaster is the long time required for training. To make this manageable different stopping criteria for the neural network training stage are provided (given number of epochs, physical training time, minimal error changes for a given number of epochs, etc.). Also, events are raised during training that allows the training process to be observed, including current state - different type of errors, validation, etc.

To allow the usage of the java library in CLIPS about 50 functions are implemented for creating, setting, training and getting results of neural network adapted that can be used in both the antecedent and consequent of the rules. Such usage provides several advantages:

1) Neural network settings can be automated by simulation expert knowledge. Many of the neural network parameters are naturally set by trial and error approach that is convenient to be realized by production IF-THEN rules. Thus, by data pre-processing and neural network architecture changes can be checked whether the built model is adequate to ex ante performance and also comparison with other known time series prediction approaches can be done and generalization ability can be checked.

The following rule tests whether the trend of the time series is non-stationary and if so then differentiation is used:

```
(defrule time-series-preprocessing
        (neural-network ?id)
        (> (get-trend-slope (get-timeseries-source-from-nn ?id)) 0.3)
=>
        (bind $?preprocessings (get-nn-ts-preprocessings ?id))
        (bind ?tsid (get-preproc-from-ts (get-timeseries-source-from-nn ?id)))
        (bind ?pos (length$ $?preprocessings))
        (add-preproc-to-position ?tsid (+ ?pos 1) "differentiation")
)
```

Similar rules can be used for the development of the neural network and training with default settings that is high level of automation in time series processing.

2) This is important for the investors' decision making. If something cannot be controlled, it should be predicted in order to take the proper action pursuing the investor's objective. Based on the predicted economic indicators the inference engine can be used as a decision-making system that performs inference operations by IF-THEN rules in the rule based system to propose a proper behaviour of the investors. In this way, also another advantage of the expert systems can be used that is they can explain how they derive their results. Such an approach is used as hybrid network architecture developed in [10], which combines a neural network with an expert system in which the neural network is used to predict future stock prices and generate trading signals.

3) Suggestions can be provided for the investment decisions based on flexible techniques of additional optimizations that can be used in conjunction with the prediction. The neural network can be trained only for a local space of the available data that is expected to be more specific and leading to important economical consequences. At such critical time horizon an additional neural network can be created and trained with specific settings in order to turn user attention into the critical economic situation. The values around immediate trend change or some local or global peak can be detected using IF-THEN rules and a procedure to be started for more precise analysis of the local time series interval. Moreover, a weighted window approach can be used [9] with different weighting schemes to define weights for observations in the training patters. The weighting approach can be not only linear but also non-linear.

The following CLIPS fragment creates a neural network with default settings followed by training and the results of the prediction are graphically shown.

```
(create-default-nn ?id ?sid ?prid $?training-data ?index)
(nn-train ?id 400 ?notification)        ;train NN with id ?id to perform 400 epochs
(bind $?pr (nn-predict ?id $?available-data 20))   ; prediction 20 values ahead
;plot the predicted values ($?pr)
```

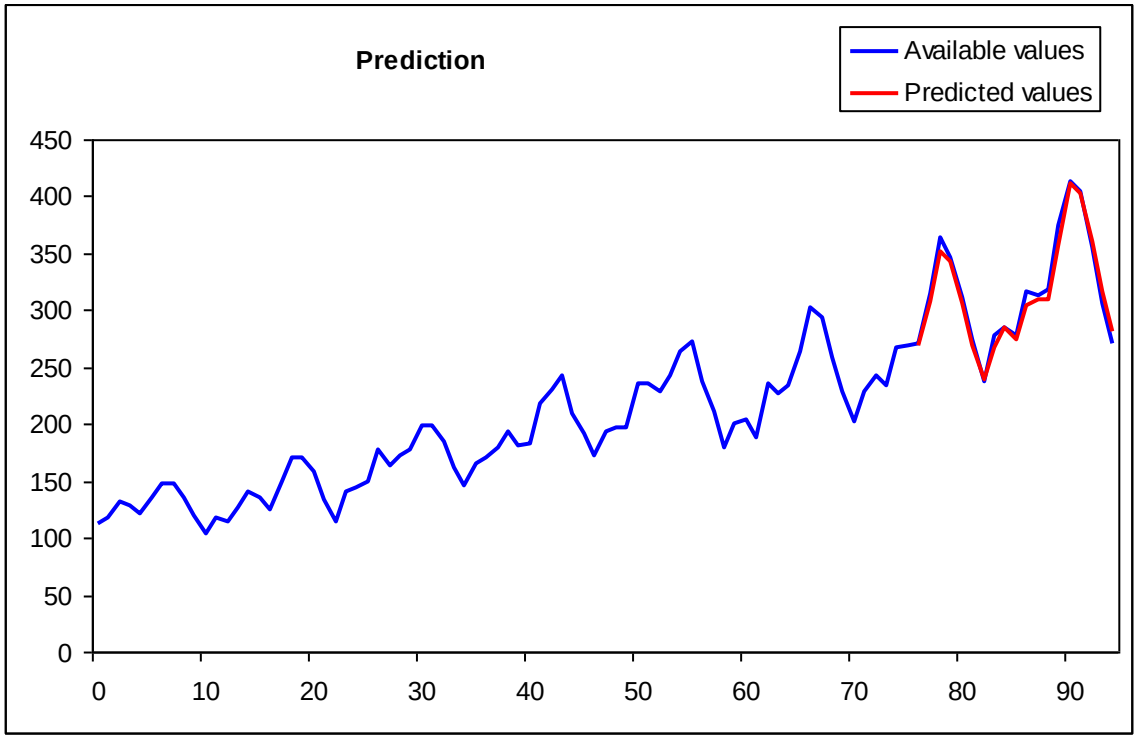

Figure 3. Time series prediction results

CONCLUSION AND FUTURE WORK

The realized hybrid system is currently used with a low level of automation. It is successfully developed and tested for standard univariate time series prediction. The future development of the systems includes integration of alternative prediction approaches such as ARMA/ARIMA, development of CLIPS rules for high level of automation and decision making, validation etc. Also, integration of clustering is to be made in the Java library to facilitate analysis of local time series data.

REFERENCES

[1] Box G. E. P., Jenkins G. M. Time Series Analysis: Forecasting and Control, San Francisco, Holden-Day, 1970.

[2] CLIPS Reference Manual, Volume I, Basic Programming Guide, 2002

[3] CLIPS Reference Manual, Volume II, Advanced Programming Guide, 2002

[4] Giles, C.L. Sun, R. Zurada, J.M. Neural Networks and Hybrid Intelligent Models: Foundations, Theory, And Applications. Neural Networks, IEEE Transactions, 1998.

[5] Hamilton, J. Time Series Analysis, Princeton University Press, 1994.

[6] Krishnamoorthy, C. Rajeev, S. Artificial Intelligence and Expert Systems for Engineers. CRC Press, 1996.

[7] Klevecka, I. Lelis, J. Pre-Processing of Input Data of Neural Networks: The Case of Forecasting Telecommunication Network Traffic. Telektronik 3/4, 2008.

[8] Larry R. Medsker. Hybrid Neural Network and Expert Systems. Kluwer Academic Publishers, 1994.

[9] Peter G. Zhang. Neural Networks in Business Forecasting. Idea Group,2004.

[10] Ramon Lawrence. Using Neural Networks to Forecast Stock Market Prices. Department of Computer Science University of Manitoba, December 12, 1997

[11] Russell, S. Norvig, P. Artificial Intelligence - A Modern Approach, 2 Ed, Ph, 2003

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

ABOUT THE AUTHORS

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

Valeri Bogdanov
Senior Software Developer
Eurorisk Systems Ltd.
31, General Kiselov Str., 9002 Varna, Bulgaria