6th International Scientific Conference
May 13–14, 2010, Vilnius, Lithuania
BUSINESS AND MANAGEMENT 2010
Selected papers. Vilnius, 2010

ISSN 2029-4441 print / ISSN 2029-428X CD
doi:10.3846/bm.2010.024
http://www.vgtu.lt/en/editions/proceedings
© Vilnius Gediminas Technical University, 2010

MODELLING OF THE HISTORY AND PREDICTIONS OF FINANCIAL MARKET TIME SERIES USING EVOLINO

Aleksandras Vytautas Rutkauskas[1], Nijolė Maknickienė[2], Algirdas Maknickas[3]

Vilnius Gediminas Technical University, Sauletekio ave. 11, LT-10223 Vilnius, Lithuania
E-mail: [1]aleksandras.rutkauskas@vgtu.lt; [2]nijole.maknickiene@vgtu.lt; [3]algirdas.maknickas@vgtu.lt

Abstract. Artificial neural networks and their systems are already capable of learning, to summarize, filter, and classify information. The increasing amount of authors are trying to teach them to approximate and predict chaotic, fractal processes. One of the greatest challenges of today's financial researches is forecasting of the commodities, stocks and currency markets. Variations in prices lead to economic indicators as result of investment process of investors and short time market players. Present article investigates recurrent neural network systems as mathematical tool for objective forecasts of fractal behaviour of financial markets by Evolino recurrent neural network learning algorithm.

Keywords: Evolino, recurrent neural networks, prediction of markets, chaos modelling, orthogonality.

1. Introduction

The objective of this paper is to research whether Evolino algorithm is good enough to investigate and predict the financial and currency market. Financial markets could be explained with profitability, reliability or risk investment models and analysis methods. Opposite of statistical analysis (Rutkauskas, Stasytytė 2008; Kucko 2007) could be sophisticated reinforcement learning agents (Ramanauskas; Rutkauskas 2009) or neural networks (Plykinas 2005).

Neural networks aids to monitor the progress of the chaotic processes in the learning process. The comparison of the various methods evaluates neural networks learning algorithms of chaotic processes and increase their prediction accuracy. (Schmidhuber *et al.* 2006) introduced a general framework of sequence learning algorithm, Evolution of recurrent systems with Linear outputs (Evolino) (Schmidhuber *et al.* 2005). Evolino uses evolution to discover good RNN hidden node weights, while using methods such as liner regression or quadratic programming to compute optimal liner mappings from hidden state to output.

When quadratic programming is using to maximize the margin, it is impossible to obtain the first evolutionary recurrent support vector machines. Evolino – based Long Short-Term Memory (LSTM) can solve tasks that Echo State nets cannot (Schmidhuber *et al.* 2006). Here was introduced a new class of recurrent, truly sequential SVM-like devices with internal adaptive states, trained by a novel method called Evolution of systems with Kernel-based outputs (Evoke), an instance of the recent Evolino class of methods.

Evoke evolves recurrent neural networks to detect and represent temporal dependencies while using quadratic programming/support vector regression and pseudoinverse regression. Evoke is the first SVM-based mechanism learning to classify a context-sensitive language. It also outperforms recent state-of-the-art gradient-based recurrent neural networks (RNNs) on various time series prediction tasks. NN learning is using for context- sensitive languages recognition and is a difficult and often increasable problem for standard recurrent neural networks (RNNs), because it require unlimited memory resources. For these array of problems Evolino based LSTM learns in approximately 3 min on average and it is able to generalize substantially better then gradient – based LSTM (Schmidhuber *et al.* 2005; Schmidhuber *et al.* 2006; Schrauwen *et al.* 2007; Wierstra *et al.* 2005) . With Evolino it makes impossible to learn functions composed of multiple superimposed oscillators such as double sine and triple sine. Network reached good learning and still makes very accurate predictions (Schmidhuber *et al.* 2005; Schmidhuber *et al.* 2006; Wierstra *et al.* 2005).

The Mackey-Glass system is a standard benchmark for chaotic time series prediction. Authors (Schmidhuber *et al.* 2005) show deviation between the curves of Evolino generated and Mackey-glass system. Evolino is capable of making precise (0.0019) prediction in tasks like the Mackey-Glass benchmark.

2. Description of Evolino learning for recurrent neural networks

The Evolino algorithm (Schrauwen *et al.* 2007) is easy to use and very clearly shows the neural network training and forecasting. Evolino learning model uses an evolutionary approach of hidden the chosen weights of optimal feedback recurrent neural networks. For this purpose, programming techniques of linear regression and the square error between NN output and initial time series in learning process is using.

As it is known, Evolino works particularly well in the three most critical problem areas (Schmidhuber *et al.* 2005): 1) context-sensitive language (Goodman *et al.* 2008) 2) combination, difficult sine wave (Schmidhuber *et al.* 2006) 3) Mackey – Glass system of training and forecasting. The goal of this work is to understand how it works for modelling and prediction of the financial markets, their behavioural analysis and paying attention to the acceptance of chosen method for the anticipation.

Evolino systems are based on two cascaded modules: model 1 – a recurrent neural network which receives the sequence of external inputs, and model 2 – a parametric function which maps the internal activations of the first module to a set of outputs. In particular, an Evoke network is governed by the following formulas:

$$\varphi(t) = f\left(W, u(t), u(t-1), \ldots, u(0)\right), \tag{1}$$

$$y(t) = w_0 + \sum_{i=1}^{k}\sum_{j=0}^{l_i} w_{ij} K\left(\varphi(t), \varphi^i(j)\right), \tag{2}$$

where: $\varphi(t) \in R_n$ is the activation at time t of the n units of the RNN, $f(.)$, given the sequence of input vectors $(u(0)...u(t))$, and weight matrix W. Note that, because the networks are recurrent, $f(.)$ is a function of the entire input history. The output $y(t) \in R$ of the combined system can be interpreted as a class label, in classification tasks, or as a prediction of the next input $(u(t+1))$, in time-series prediction. In order to find a W that minimizes the error between $y(t)$ and the correct output, Evolino use artificial evolution. Starting with random population of real-numbered strings or chromosomes representing candidate weight matrices, it evaluates each candidate through the following two-phase procedure.

In the first phase, the aforementioned training set of sequence pairs $(u_i, d_i, i = 1...k)$, each of length l_i , is presented to the network. For each input sequence u_i ,starting at time t = 0, each pattern $u_i(t)$ is successively propagated through the RNN to produce k a vector of activations $\varphi_i(t)$ that is stored as a row in a $n \times \sum_i^k l_i$ matrix φ. Associated with each input sequence is a target d_i in D containing the correct output values for each time step. Once all k sequences have been seen, the weights w_{ij} of the kernel model (2) are computed using support vector regression/classification from φ and D, with φ_i, d_i as training set.

In the second phase, a validation set is presented to the network, but now the inputs are propagated through the RNN and the newly computed output connections to produce $y(t)$. The error in the classification-prediction or the residual error, possibly combined with the error on the training set, is then used as the fitness measure to be minimized by evolution. By measuring error on the validation set rather that just the training set, RNNs will receive better fitness for being able to generalize. Those RNNs that are most fit are then selected for reproduction where new candidate RNNs are created by exchanging elements between chromosomes and possibly mutating them. New individuals replace the worst old ones and the cycle repeats until a sufficiently good solution is found.

Evoke is instantiated using Enforced Sub-Populations (ESP) to evolve Long Short-Term Memory (LSTM) networks. ESP differs from standard neuro-evolution methods in that, instead of evolving complete networks, it co-evolves separate sub-populations of network components or neurons

LSTM is an RNN purposely designed to learn long-term dependencies via gradient descent. The unique feature of the LSTM architecture is the memory cell that is capable of maintaining its activation indefinitely. Memory cells consist of a linear unit which holds the state of the cell, and three gates that can open or close over time. The Input gate "protects" a neuron from its input: only when the gate is open, can inputs affect the internal state of the neuron. The Output gate lets the internal state out to other parts of the network, and the Forget gate enables the state to "leak" activity when it is no longer useful. The gates also receive inputs from neurons, and a function over their input (usually the sigmoid function) decides whether they open or close. Hereafter, the term gradient-based LSTM (G-LSTM) will be used to refer to LSTM when it is trained in the conventional way using gradient-descent.

ESP and LSTM are combined by co-evolving sub-populations of memory cells instead of standard recurrent neurons. Each chromosome is a string containing the external input weights and the Input, Output, and Forget gate weights, for a total of $4 \times (I + H)$ weights in each memory cell chromosome, where I is the number of external inputs and H is the number of memory cells in the network.

There are four sets of $I+H$ weights because the three gates and the cell itself receive input from outside the cell and the other cells. ESP normally uses crossover to recombine neurons. However, for Evoke, where fine local search is desirable, ESP uses only mutation. The top quarter of the chromosomes in each sub-population are duplicated and the copies are mutated by adding Cauchy distributed noise to all of their weight values.

The support vector method used to compute the weights (w_{ij} in equation 2) is a large scale approximation of the quadratic constrained optimization, as implemented . For continuous function generation, back-projection (or teacher forcing in standard RNN terminology) is used, where the predicted outputs are fed back as inputs in the next time step, such that 1 becomes:

$$\varphi(t) = f\big(W, u(t), y(t-1), \ldots, u(1), y(0)\big) \tag{3}$$

During training and validation, the correct target values are back-projected, in effect "clamping" the network's outputs to the right values. During testing, the network back-projects are its own predictions.

3. Orthogonal functions and time series approximations

3.1. Orthogonal functions

It is common to use the following inner product for two functions f and g:

$$\langle f, g\rangle = \int f(x)g(x)w(x)dx \tag{4}$$

where:

$w(x)$ is a non-negative weight function of the definition of inner product. These functions are orthogonal if that inner product is zero:

$$\int f(x)g(x)w(x)dx = 0. \tag{5}$$

We write the norms with respect to this inner product and the weight function as

$$\|f\|_w = \sqrt{\langle f_i, f_j\rangle_w} \tag{6}$$

The members of a sequence $(f_i, i = 1,2,3,\ldots)$ are orthogonal on the interval $[a,b]$ if

$$\langle f_i, f_j\rangle = \int f_i(x)f_j(x)w(x)dx \tag{7}$$

orthonormal on the interval $[a,b]$ if

$$\langle f_i, f_j\rangle = \delta_{ij} \tag{8}$$

where:

$$\delta_{ij} = \begin{cases} 1 & if \quad i = j \\ 0 & if \quad i \neq j \end{cases} \tag{9}$$

is the Kronecker delta. In other words, any two of them are orthogonal, and the norm of each is 1 in the case of the of each is 1 in the case of the orthonormal sequence.

3.2. Galerkin method

Choose a subspace $V_n \subset V$, dimension n and solve the projected problem: Find $U_n \subseteq V_n$ such that for all $v \subseteq V_n$

$$a(u_n, v_n) = f(v_n) \tag{10}$$

We call this the Galerkin equation. Notice that the equation has remained unchanged and only the spaces have changed. The key property of the Galerkin approach is that the error is orthogonal to the chosen subspaces. Since $V_n \subset V$, we can use v_n as a test vector in the original equation. Subtracting the two, we get the Galerkin orthogonality relation for the error, $e_n = u - u_n$ which is the error between the solution of the original problem, u, and the solution of the Galerkin equation, u_n is

$$\begin{aligned} a(e_n, v_n) &= a(u, v_n) - a(u_n, v_n) \\ &= f(v_n) - f(v_n) = 0 \end{aligned} \tag{11}$$

Since the aim of Galerkin's method is the production of a linear system of equations, we build its matrix form, which can be used to compute the solution by a computer program. Let $e_1, e_2, e_3, \ldots, e_n$ be a basis for V_n. Then, it is sufficient to use these in turn for testing the Galerkin equation, i.e.: find $u_n \subseteq V_n$ such that

$$a(u_n \cdot e_i) = f(e_i) \tag{12}$$

where:

$i = 1,2,\ldots,n$ We expand u_n in respect to this basis,

$$u_n = \sum u_j \cdot e_j \tag{13}$$

and insert it into the equation above, to obtain

$$a\Big(\sum u_j e_j e_i\Big) = \sum u_j \cdot a(e_j, e_i) = f(e_i) \tag{14}$$

where $i = 1,2,\ldots,n$.

This previous equation is actually a linear system of equations $Au = f$, where

$$a_{i,j} = a(e_i, e_j) \tag{15}$$

$$f_i = f(e_i) \tag{16}$$

3.3. Basic input time series of RNN's

Above chosen mathematical method is using for modelling of continuous functions. We use this method for finding of best input bases in Evolino learning process. Other authors (for example McNelis 2005) which exploring financial prognoses are searching dependences between time series of financial indicators or similar patterns in the same time series. Our proposal is to find time series of best orthogonality as bases for modelling of history of chaotic financial time series and their future behaviour. In this case orthogonality could be described as finite sum opposite to integration in equation 7. Now we must make assumption that there is equation 10 which could describe time series of financial market. Our goal was finding of this equation. We were used for this approach recurrent neural network Evolino learning algorithm described above.

4. Numerical investigation

Pybrain framework (Gudman *at al.* 2008) was used for researching and forecasting of the time series of the financial market. It is very convenient python based framework of RNN including and Evolino learning algorithm. It was adopted to output not the last but the elected best result of the learning RMSE chaotic time series in the transformation process of neural networks.

Time series orthogonality and Evolino neural network learning algorithm was investigated numerically. Two orthogonal bases were formed on stock market indexes and Gold prices and currencies exchange pairs and Gold prices. New founded bases were as inputs for RNN time series modelling and prediction. Finally, it was created these databases: DJIA (The Dow Jones Industrial Average), NasdaQ (National Association of Securities Dealers Automated Quotations), GOLD which is historically the basis for all financial resources.. For approving of proposed NN prediction model with Evolino learning and orthogonal inputs daily ticks of DJIA, NasdaQ, and GOLD were used. The target of all market predictions was their dynamic aspect. Each base input pair was learned several times with the best orthogonality by shifting of base ticks. Resulting RMSE was used for monitoring of influence of given bases for learning and prediction of the recurrent neural network. At first, we also were examined dynamic of the currencies exchange market. First training input was Gold daily price time sequence for prediction of the USD/JPY (U.S. dollar and Japanese Yen), the EUR/USD (euro and US dollar) and the GBP/USD (Great Britain pound and US dollar) exchange pairs.

Selecting input ranges shows better neural network learning and prediction results when the better or thogonality were used. For these purpose it was created new python script. This script calculates ranges for better orthogonality. The closer to zero value indicated higher orthogonality of the input base pairs.

For the taken NasdaQ and Gold base, DJIA and Gold base, NasdaQ and DJIA bases we found time intervals with different orthogonality values from 0.02155 to 0.000023. It was used 64 neural networks and 220 epochs. It was studied RMSE dependence from orthogonality in neural network learnings and obtained learning and prediction results in each couple of stock indexes.

We were examined dynamic of the currencies exchange market. First training input was Gold daily price time sequence for prediction of the USD/JPY (U.S. dollar and Japanese Yen), the EUR/USD (euro and US dollar) and the GBP/USD (Great Britain pound and US dollar) exchange pairs. For the taken USD/JPY and Gold base we found time intervals with different orthogonality values from 0.021272 to 0.00000351. It was used 64 neural networks and 220 epochs. It were studied RMSE in neural network learning and obtained forecasting, when was used the best orthogonal intervals. The best forecasting is shown in Fig. 1.

5. Conclusions

By observing of RNN learning on financial market indexes DJIA, NasdaQ and GOLD was found that the result is highly dependent on choice of the input and output ranges. The study of the standard deviation's dependence of learned RNN on orthogonality in selected ranges got much better results with higher data orthogonality.

Different pairs of indexes: DJIA and NasdaQ, DJIA and GOLD, NasdaQ and GOLD were used for studies of RNN's learning and prediction. The best neural network learning and prediction results were reached for DJIA and NasdaQ indexes. The RMSE = -0.002535 of learned neural network was obtained for NasdaQ index with 2 predictable days. Other results with the similar RMSE indicate just the right moving directions of investigated indexes.

We were examined each of three input bases of currency market: daily ticks of USD/JPY and GOLD, daily ticks of EUR/USD and GOLD, daily ticks of GBP/USD and GOLD for learning and testing of USD/JPY, EUR/USD, GBP/USD. The study of the standard deviation's dependence of learned neural networks on orthogonality of selected ranges got much more better results for higher orthogonality.

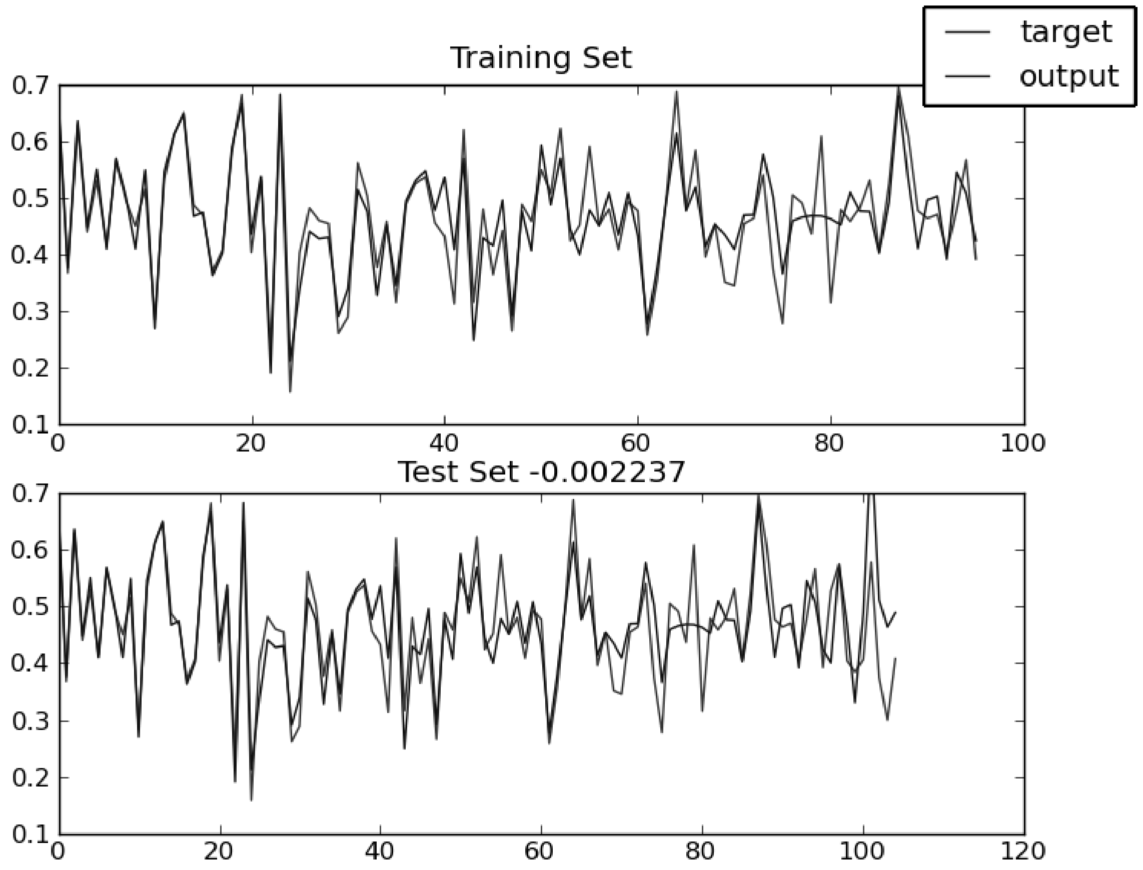

Fig. 1. RMSE in neural network learning and obtained forecasting

By searching of best intervals of the time series corresponding to the received orthogonality, we obtained the best results for USD/JPY prediction. It approves a good known observation of market players on dependences of USD/JPY and GOLD market prices. The best prediction with RMSE = -0.002237 was found for USD/JPY with 3 predictable days. Other results show just predictable moving directions of EUR/USD and GOLD input bases and EUR/USD output. Meanwhile, the GBP/USD and GOLD base input and GBP/USD output learning RMSE = -0.005997 was to low for using in forecasts.

In the summary, the Evolino algorithm is good enough to investigate and predict the financial and currency markets, if properly selected orthogonal bases of time series are using in training of chaotic financial data.

References

Goodman, D.; Brette, R. 2008. Brian: a Simulator for Spiking Neural Networks in Python, *Frontiers in Neuroinformatics* 2: 5. doi:10.3389/neuro.11.005.2008

Kucko, I. 2007. Investment Fund Portfolio Selection Strategy, *Verslas:teorija ir praktika* [Business: Theory and Practice] 8(4): 214–220.

McNelis, P. D. 2005. *Neural Networks in Finance: Gaining Predictive Edge in the Market.* London: Elsevier Academic Press. 262 p. ISBN 0-12-485967-4.

Plikynas, D. 2005. Explaining International Investment Patterns: a Neural Network Approach, *Information technology and control* 34(1): 42–50.

Ramanauskas, T.; Rutkauskas A. V. 2009. Empirical Version of an Artificalt Stock Market Model, *Pinigų studijos* [Study of Money] 2(1):5–21.

Rutkauskas, A. V.; Ramanauskas, T. 2009. Building an Artificial Stock Market Populated by Reinforcement-Learning Agents, *Journal of Business Economics and Management* 9(4): 329–341. doi:10.3846/1611-1699.2009.10.329-341

Rutkauskas, A. V.; Stasytytė V.; Borisova J. 2009. Adequate Portfolio as a Conceptual Model of Investment Profitability, Risk and Reliability Adjustment to Investor's Interests, *Ekonomika ir vadyba* [Economics and Management] 14: 1170–1174.

Rutkauskas, A. V.; Stasytytė V. 2008 . Stratification of Stock Profitabilities – the Framework for Investors' Possibilities Research in the Market, *Intelektualinė ekonomika* [Intelectual Economics] 21(3): 65–72.

Schmidhuber, J.; Gagliolo, M.; Wierstra, D.; Gomez, F. 2006. Evolino for Recurrent Support Vector Machines, in *ESANN'2006 proceedings. European Symposium on Artificial Neural Networks.* Bruges, Belgium, 26–28 April 2006. ISBN 2-930307-06-4.

Schmidhuber, J.; Wierstra, D.; Faustino Gomez, F. 2005. Evolution: Hybrid Neuroevolution / Optimal Linear Search for Sequence Learning, in *Proceedings of the 19th International Joint Conference on Artificial Intelligence.* TU Munich, Boltzmannstr. 3, 85748 Garching, Munchen, Germany IDSIA, Galleria 2, 6928 Lugano, Switzerland.

Schrauwen, B.; Verstraeten, D.; Campenhout, J. V. 2007. An Overview of Reservoir Computing: Theory, Applications and Implementations, in *Symposium on Artificial Neural Networks.* Bruges, Belgium, 25–27 April 2007. 471–482. ISBN 2-930307-07-2.

Wierstra, D.; Gomez, F. J; Schmidhuber, J. 2005. Modeling Systems with Internal State using Evolino, in *Conference on Genetic and Evolutionary Computation GECCO*. Washington, USA, New York: ACM Press, 1795–1802.