International Journal of Computer and Information Technology (ISSN: 2279 – 0764)
Volume 03 – Issue 05, September 2014

Currency Exchange Rate Forecasting Using Machine Learning Techniques

Denada Xhaja, Ana Ktona, Gazmira Brahushi
Department of Computer Science
Faculty of Natural Science
Tirana, Albania

Abstract—**Many economic decisions are made based on exchange rate of currencies making thus the forecasting of it an importance task. Furthermore, forecasting models/methods have become even a challenging task due to the fact that currency exchange rate is a highly volatile one. This paper deals with the application of Support Vector Machine (SVM) and Artificial Neural Network (ANN) in one step forecasting of EUR/ALL exchange rate based on the past behavior and patterns. EUR/ALL exchange rate on working days for a period of four years is used in this study. Several experiments are conducted applying different parameters to both techniques. Root Mean Square Error (RMSE) is used as a measure to evaluate and compare the predictive power of SVM and ANN.**

Keywords-Support Vector Machine; Artificial Neural Network; forecasting; exchange rate;

I. INTRODUCTION

Many economic decisions are made based on exchange rate of currencies making thus the forecasting of it an importance task. Furthermore, forecasting models/methods have become even a challenging task due to the fact that currency exchange rates are influenced by many political and economical factors.

Historically, exchange rates have operated under three different types of regimes. Albania is one of the countries that apply the flexible exchange rate regime, in which the exchange rate between two currencies fluctuates freely in the foreign exchange market. The advantage of a flexible Exchange rate system is that the Exchange rate is a market-determined price that reflects economic fundamentals at each point in time. Governments do not intervene to defend some exchange rate level, so there is no inventive to “speculate” against them. The disadvantage is that exchange rate can be quite volatile. Due to the volatile nature of the flexible regime, forecasting exchange rate is a difficult task.

This paper provides insight into two machine learning techniques used to forecast the behavior of the foreign exchange market especially on short-run fluctuations. These techniques are Support Vector Machine and Artificial Neural Network. They will be used for the daily forecasting of EUR/ALL exchange rate based on the past behavior and patterns.

II. MACHINE LEARNING TECHNIQUES FOR FINANCIAL TIME SERIES PREDICTION

A time series is a time-ordered sequence of observation values of a physical or financial variable made at equally spaced time intervals Δt, represented as a set of discrete values x_1, x_2, x_3, ... , etc. [1]. Time series analysis comprises methods for data compression, explanatory, signal processing and prediction. In this paper we will focus on the prediction. Prediction has to do with forecasting future values based on previously observed values. Time series forecasting has important applications in various fields because good decisions are made based on the forecasted results.

In the last two decades, machine learning models have drawn attention and have established themselves as serious contenders to classical statistical models in the forecasting community [2, 1, 3, 4]. Machine Learning Techniques used to predict the value of a numerical class based on one or more numerical attributes are regression techniques. These techniques build a model, usually a mathematical equation, which measures the value of the numerical dependent variable (class) based on a set of numerical independent variables (attributes). Regression algorithms can build linear or nonlinear models. A linear model assumes that the relation between the dependent variable and independent variables is linear (the class is represented as a linear combination of the attributes). Linear regression is one of the simplest algorithms that build a linear model [5]. It is a mathematical procedure that evaluates the "most appropriate" line through data. The “Most Appropriate” line is the line that is described by the linear equation where the weights and bias are defined by minimizing the mean of square errors. Linear regression is useful to predict the dependent variable when the tendency of this variable in time is linear. In terms of time series when is an upward or downward trend in the data over time. However, if the tendency of the dependent variable in time is not linear then the regression would not understand the relationship. Linear regression fails to capture seasonal trends or cycles in time series data. Also linear regression cannot distinguish the effects of changes in time series direction nor the degree of change over time. When we use the linear regression in time series is important to keep a part of the data at any time and examine it for possible non-linear trends. Due to the non-linearity of EUR/ALL exchange rate we cannot use linear Machine

Learning models in daily forecasting of it. Linear Machine Learning algorithms can be extended to represent nonlinear future decisions. Two of the most used techniques in financial time series forecasting are Support Vector Regression and Artificial Neural Network.

A. Support Vector Machines

Support vector machines are based on the algorithm maximum-margin hyperplane. This algorithm finds, in a two linearly separable class datasets, a linear discriminate model that separates the two classes as much as possible. Maximum-margin hyperplane is represented by a function which is build from the instances of the two classes that have the shortest path between them and perpendicular to the hyperplane. These instances (at least two: one of each class) are called the support vectors. If we have the support vectors we can build the maximum-margin hyperplane. The hyperplane may be determined by a function of the following form:

$$f(x) = b + w \bullet x$$

Where x represents the test instance and w and b represent the values that must be learned by the algorithm.

The dot product in a dataset with k attributes can be written as:

$$\sum_{i=1}^{k} w_i * x_i$$

As a result the function that determined the hyperplane can be written as:

$$f(x) = b + \sum_{i=1}^{k} w_i * x_i$$

We can represent the class value as 1 for one class and -1 for the other class in a two class datasets. We can extended the representation of class values in datasets with more than two classes by using one of the classes as a positive one and labeled 1 and the other classes negative ones and labeled -1. The parameter w is learned by the support vectors and can be written as:

$$\sum_{j \text{ index of the support vector}} \alpha_j * x_j * y_j$$

As a result the function that determined the hyperplane can be written as:

$$f(x) = b + \sum_{j \text{ index of the support vector}} \alpha_j * y_j * \sum_{i=1}^{k} x_{ji} * x_i$$

Now, the parameters that must be learned by the algorithm and determined the hyperplane are b and α_j.

Until now we described the case where the datasets were linearly separable. The function describing the hyperplane could be extended for datasets that are not linearly separable. This can be done by finding the function Ω that map the instances of the dataset into another space of n dimensions where the classes of the data are linearly separable and can represented by a linear equation. So if we found Ω that map X (the training dataset) to the high-dimensional space L we can replace it to the function that described the hyperplane where the instances of the dataset appear. So x_{ji} is replaced by Ω (x_{ji}) and x_i is replaced by Ω (x_i) and the function defining the hyperplane in the high-dimensional space L is:

$$f(x) = b + \sum_{j \text{ index of the support vector}} \alpha_j * y_j * \sum_{i=1}^{k} \Omega(x_{ji}) * \Omega(x_i)$$

Where the parameters b and α_j has to be learned by the algorithm to find the maximum margin hyperplane. The algorithm the concept of which we explained until now applies to classification. This algorithm could be extended to apply also in regression. The algorithm will try to find a function which will minimize the error of prediction while approximates the class' value of the training instances. Here a parameter e is introduced which is user specified. The function that will be found by the algorithm will use only the instances falling outside or on the border of the tube with the diameter 2e. These instances are the support vectors. The version of support vector machines for regression is called support vector regression. When the class of data can be represented by a linear equation the function learned by the application of support vector regression algorithm can be written as:

$$f(x) = b + \sum_{j \text{ index of the support vector}} \alpha_j * \sum_{i=1}^{k} x_{ji} * x_i$$

And as explained above for the case of classification the instances could be replaced by their mapping function in the high-dimensional space.

$$f(x) = b + \sum_{j \text{ index of the support vector}} \alpha_j * \sum_{i=1}^{k} \Omega(x_{ji}) * \Omega(x_i)$$

Another parameter that is used in Support Vector Regression is the C parameter. This parameter controls the tradeoff between the prediction error and the complexity of the model. The dot product $\sum_{i=1}^{k} \Omega(x_{ji}) * \Omega(x_i)$ is called the kernel function. The kernel could be linear or non-linear. Some of the most used kernels of non-linear category for numerical data are polynomial kernel of d degree where d >1, Radial Basis Function kernel known also as Gaussian kernel and Sigmoid kernel. Another kernel used in Support Vector Regression is Pearson VII Universal Kernel function based kernel [6]. Polynomial kernel of degree d can be written as:

$$K(x_{ji}, x_i) = (x_{ji} \bullet x_i + c)^d$$

The parameter c is a constant value. When d=1 the kernel is linear.

The function of Gaussian RBF kernel can be written as:

$$K(x_{ji}, x_i) = \exp\left(\frac{\|x_{ji} - x_j\|_2^2}{-2\sigma^2}\right)$$

The normalized Polynomial kernel of degree d can be written as:

$$K(x_{ji}, x_i) = \frac{(Xji \cdot Xi + c)^d}{\sqrt[2]{(Xi \cdot Xi + c2)^d (Xji \cdot Xji + c1)^d}}$$

The PUK kernel can be written as:

$$K(x_{ji}, x_i) = \frac{H}{\left[1 + \left(\frac{2 \cdot \sqrt{2^{\frac{1}{\omega}} - 1}\sqrt{\|x_{ji} - x_i\|^2}}{\sigma}\right)^2\right]^{\omega}}$$

B. *Artificial Neural Network*

An artificial neural network (ANN) is an information processing system inspired by the biological neural system. It consists of several layers each of them containing a number of neurons. Although in the meantime the variety of proposed neural network structures has grown, the multilayered perceptron (MLP) has remained the prevailing one and also the most widespread network structure [1]. Figure 1 shows the architecture of a Multilayer Perceptron.

The single layer MLP with one hidden layer is the most basic and commonly used neural network in economic and financial applications [7]. The input layer of MLP is the layer where the past observations of data series are received, while the output layer is the one that produces the future predicted value. Each layer in the MLP is fully connected to the next one. The connections are associated with a parameter called weights. Except for the neurons in the input layer, on each neuron of the other layers is applied a nonlinear activation function. The mainly used activation functions are the hyperbolic tangent (equation 1) that ranges from -1 to 1 and the logistic function (equation 2) that ranges from 0 to 1.

$$Y_j = \tanh(V_j) \tag{1}$$

$$Y_j = 1/(1+e^{-V}j) \tag{2}$$

$$V_j = w_{0j}x_{0j} + w_{1j}x_{1j} + ... + w_{ij}x_{ij} + ... + w_{lj}x_{lj} \tag{3}$$

Y_j is the output of node j, V_j is the weighted sum of the input nodes to node j, x_{ij} represents the *i*th input to node j, w_{ij} represents the weight associated with the *i*th input to node j. There are l+1 inputs to node j.

These networks learn through a supervised learning method which is the back propagation algorithm. In the forward path during training the last s observations (input values) from time t are propagated through the single hidden layer into the one output neuron which produces the predicted observation at the time t+1. S is the size of input window.

The error performed is computed by comparing the output of MLP with the actual value of time series at time t+1. This error is propagated through network in a backward direction from output layer to the hidden layer and then from hidden layer to input layer, computing the contribution of each connection between neurons in this error and also adjusting the corresponding weights in order to minimize the error. The weights are updated as follows:

$$w_{ij,new} = w_{ij,current} + \eta \delta_j x_{ij}$$

where η represents the learning rate which is a constant that helps to move network weights toward a global minimum for the error, δ_j represent the responsibility for a particular error belonging to node j, and x_{ij} is the *i*th input to node j.

δ_j for nodes on output layer is computed as:

$$\delta_j = \text{output}_j(1-\text{output}_j)(\text{actual}_j-\text{output}_j)$$

While for nodes on the hidden layers δ_j is computed as:

$$\delta_j = output_j(1 - output_j) \sum_{downstream} w_{jk}\delta_j$$

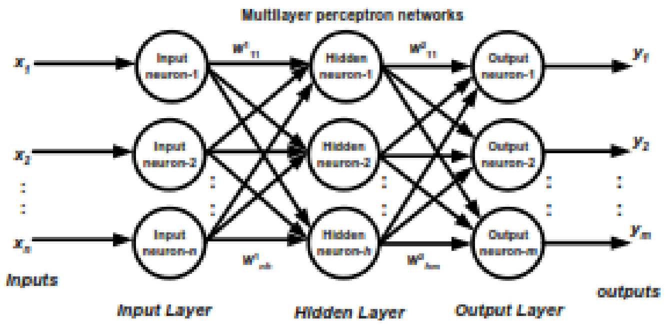

Figure 1. MLP architecture Source [1]

At the end of training process weights of connections between neurons represent the knowledge learned. These weights will be used to buid a model that gives us valid results when fed with new data.

III. EXPERIMENTS

The dataset studied in this paper consist of EUR/ALL daily rate for a period between 1 January 2010 and 31 December 2013. The data series was obtained from official website of Bank of Albania. All experiments are conducted using Weka machine learning algorithms [8]. For the experiments involving ANNs the algorithm is applied directly to the dataset, while for the experiments involving SMOReg the algorithms are called from our Java code.

The sample size is the total dataset period of 4 years for the EUR/ALL dataset, a total 1041 values. The dataset does not include an exchange rate for weekends and holidays. We considered as missing values only holidays and computed the values for these days as an average of previous and the consecutive day of that day (holiday). The dataset is normalized to [0,1] using linear transformation:

$x_n=(x_0-x_{min})/(x_{max}-x_{min})$

The dataset is divided into two subsets, 70% for the training set and 30% for testing set. The experiment was conducted with various window sizes. Root Mean Square Error (RMSE) is used as a measure to evaluate and compare the predictive power of Support Vector Regression and ANN. RMSE error is square root of the MSE (Mean Squared Error) which measures the variability in prediction errors. The difference between the predicted and actual value is squared before average is applied, as a result when RMSE is high shows that a lot of large errors are generated from the model. Comparison of different model variations in terms of parameters and structure and comparison of the two techniques is based on comparing the performance on the test set.

A. *Experiments using SVM for regression*

The SMOReg algorithm implements Support Vector Machine for regression. This algorithm is executed using Poly Kernel, PUK, RBF Kernel and Normalized Poly Kernel. For each kernel type several experiments are conducted with lag number equal to 3, 5, 7 and 9.The errors generated on train and test set from the application of support vector regression algorithm with the different appropriate kernels and different lags are shown on table I and II.

TABLE I. RMSE GENERATED BY SMOREG ON TRAIN SET

Kernel Type	Lag =3	Lag=5	Lag=7	Lag=9
Poly Kernel E = 1	0.0269	0.0268	0.0268	0.0267
PUK	0.0583	0.0725	0.081	0.0891
RBF Kernel	0.0877	0.0914	0.0923	0.0929
Normalized Poly Kernel	0.0479	0.0463	0.0493	0.05

TABLE II. RMSE GENERATED BY SMOREG ON TEST SET

Kernel Type	Lag =3	Lag=5	Lag=7	Lag=9
Poly Kernel E = 1	0.0139	0.0138	0.0149	0.0152
PUK	0.1416	0.1442	0.1472	0.1513
RBF Kernel	0.0518	0.0276	0.0225	0.0231
Normalized Poly Kernel	0.0476	0.0297	0.0298	0.049

We notice the error for the PUK kernel is larger in test data than error for PUK kernel in training data. The application of Support Vector Regression with PUK kernel in our data shows over fitting. As a result this kernel will not be considered in the performance comparison. When the algorithm is executed using PolyKernel and NormalizedPolyKernel the lowest error value is obtained for number of lags equals to 5. In the case of RBF kernel the lowest error value is when the number of lags is 7. The lowest error value is RMSE = 0.0138. This value is generated when the algorithm is used with the PolyKernel kernel and when the number of lags is 5.

B. *Experiments using ANN*

The ANN model consists of only one hidden layer. We conducted several experiments with different number of nodes in the hidden layer. The transfer function used in our experiments is the sigmoid function. The number of input nodes (lag or window size) will be 3, 5, 7 and 9. The number of output node is one in all experiments, and it corresponds to the next predicted value. For each window size or lag, we performed several experiments with different number of hidden nodes.

The best neural network within each window size, defined by the network performance on test set is shown on table III:

TABLE III. RMSE GENERATED BY ANN ON TRAIN AND TEST SET

Lag/number of input neurons	Number of hidden neurons	RMSE on train set	RMSE on test set
3	2	0.0261	0.0158
5	3	0.025	0.0195
7	3	0.0248	0.0227
9	3	0.0247	0.024

These results show that in all cases RMSE on test set is slightly better than that on train set. For lag equal to 3, RMSE on test set is better than RMSE on test set for other lags, while RMSE on train set is slightly worse than RMSE on train set for other lags.

C. *Results from both experiments*

As can be noted on TableI, II and III the best model of SMOReg (kernel = PolyKernel and lags = 5) has a lower error on test set than the best ANN model (lag/input neurons = 3 and hidden neurons = 2), but the error they generated on the train set is almost the same. The 3-2-1 ANN model performs better than other SMOReg models on both train set and test set.

CONCLUSIONS

Many economic decisions are made based on exchange rate of currencies making thus the forecasting of it an importance task. In this paper we applied Support Vector Machine and Artificial Neural Network in one step forecasting of EUR/ALL exchange rate based on the past behaviour and patterns. The dataset consists of EUR/ALL exchange rate on working days for a period from January 2010 to December 2013. Several experiments are conducted applying different parameters to both techniques. Root Mean Square Error is used as a measure to evaluate and compare the predictive power of Support Vector Regression and ANN. The experiments conducted showed that the best model of SMOReg is the one that uses PolyKernel and 5 as number of lags. It performs better than 3-2-1 ANN model on the test data, but on train data they perform almost the same.

REFERENCES

[1] A. K. Palit and D. Popovic, "Computational Intelligence in Time series Forecasting", ISBN 1852339489 2005.

[2] Ahmed, Nesreen K. , Atiya, Amir F. , Gayar, Neamat El and El-Shishiny, Hisham(2010) 'An Empirical Comparison of Machine Learning Models for Time Series Forecasting', Econometric Reviews, 29: 5, 594 — 621

[3] G. Zhang, B. Eddy Patuwo, and M.Y. Hu, Forecasting with artificial neural networks: The state of the art. International Journal of Forecasting 14(1), 35–62 (1998).

[4] Gianluca Bontempi, Souhaib Ben Taieb, and Yann-A¨el Le Borgne, Machine Learning Strategies for Time Series Forecasting.

[5] C. L. Lawson, and R. J. Hanson, Solving least squares problems. Philadelphia: SIAM Publications, 1995.

[6] B. Üstün, W.J. Melssen, and L.M.C. Buydens, Facilitating the application of Support Vector Regression by using a universal Pearson VII function based kernel. Chemometrics and Intelligent Laboratory Systems 81: 29-40 (2006)

[7] Paul D. McNelis, Neural Network in Finance: Gaining Predictive Edge in the Market, Elsevier Academic Press, 2005

[8] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, Ian H. Witten (2009); The WEKA Data Mining Software: An Update; SIGKDD Explorations, Volume 11, Issue 1.