Recurrent Self-Organising Maps and Local Support Vector Machine Models for Exchange Rate Prediction

He Ni and Hujun Yin

School of Electrical and Electronic Engineering, University of Manchester, Manchester, UK
he.ni@manchester.ac.uk, hujun.yin@manchester.ac.uk

Abstract. This paper considers the problem of predicting non-linear, non-stationary financial time sequence data, which is often difficult for traditional regressive models. The Self-Organising Map (SOM) is a vector quantisation method that represents statistical data sets in a topology preserving fashion. The method, which uses the Recurrent Self-Organising Map(RSOM) to partition the original data space into several disjointed regions and then uses Support Vector Machines (SVMs) to make the prediction as a regression method. It is model free and does not require a prior knowledge of the data. Experiments show that the method can make certain degree of profits and outperforms the GARCH method.

1 Introduction

The problem of financial time series prediction is of primary importance to modern economical world. Financial time series forecasting is a challenging problem since most conventional statistical algorithms are not naturally suited for time varying patterns. Sudden changes in policy, natural and man-made disasters, institutional changes, new discoveries and revisions among others cause occasional large forecast errors in the standard, constant-parameter models. The traditional way, for example autoregressive moving average (ARMA) [1], to approach the problem is to model the underlying time series structures globally. In order to improve the quality of regression models in economic data modeling, autoregressive conditional heteroskedastic (ARCH) models were introduced by Engle [2] and extended by Bollerslev to GARCH model [3].

However, statistical models still suffer from being distorted by the non-stationary nature of most financial time series. If data series are divided into different regions, the noise level may vary from region to region. Therefore, overfitting or under-fitting will inevitably exist in one single model. Alternatively, artificial neural networks have been proved with various success in modeling such non-stationary data [4]. In addition, inspired by a so-called divide-and-conquer principle, a discretisation function (clustering) and several local models that can be combined to solve a complicated problem [5, 6, 7]. Local models are then built up on the divided sub sets, which would be of relatively smaller size and smaller variation comparing to the global set.

The data partition is normally done by clustering methods, for example neural gas [8] and SOM [9]. Afterwards, the local models are set up by using the Support Vector Machines(SVMs) [10]. In this work, SVMs serve as regression methods to complete the prediction tasks.

The rest of the paper is organised as follows, Section 2 introduces the so-called divide-and-conquer architecture and the algorithm of tree type RSOM and SVM for regression task. Experimental results are provided in Section 3. Finally, a brief conclusion and future work is given.

2 Architecture and the Learning Algorithm

For time series prediction, one has to capture the dependency between the past and the future behaviour of the time sequence. A typical one step ahead prediction can be formulated as

$$x(n) = [\gamma_1 x(n-1) + \gamma_2 x(n-2) + \ldots + \gamma_i x(n-i)] \ . \tag{1}$$

$x(n)$ is the value of time series at time step n, where $n = 1, 2, \ldots, i$. During the learning process, provided the actual process can be described by such a linear model, the coefficients $\gamma_k, k = 1, 2, \ldots, i$ should gradually converge under the least square method. However, in realities, γ_k may not always converge [11].

Global models give the best results with stationary time series. Whilst local models can often overcome some problems of global models and have gained growing interests during the last decade. By using local models, the possibility to converge in each smaller local region is highly enhanced. Thereafter, predictions are then done on local models.

2.1 Tree Type Recurrent Self-Organising Map

The Self-Organising Map (SOM) is a vector quantisation method to map patterns from a high dimensional input space ϕ to a lower dimensional prototype space φ while preserving the topological relationships among the nodes [9]. The network consists of a lattice of nodes, and in this case it is a tree. Starting from a main stem, then it gradually grows to branches, then the secondary branches, etc. Such a tree structure has been used in document management [12]. Defined by its position r, each node of the tree has a reference weight w_r of the same dimension as the input. In order to preserve the topology during the training as well as keep the training processing from being trapped in local minima, a proper neighbourhood function h has to be used.

The training algorithm of the SOM is: randomly choose one input vector $x(n)$ from an input space ϕ and compare it with the reference vector w_r of the neuron r on the output space φ. The best matching unit v to the input is the one having its weight w_v closest to it. Reference vectors (weights) are randomly set at the beginning, and then are updated by

$$w_r(n+1) = w_r(n) + \alpha(n) h_{rv}(n)(x(n) - w_r(n)) \ . \tag{2}$$

where $0 < \alpha(n) < 1$ is the learning rate, as n increases, $\alpha(n)$ decreases, with the constraints $\lim_{n\to\infty} \int_0^n \alpha(n)dn = \infty, \lim_{n\to\infty} \int_0^n \alpha(n)^2 dn < \infty$.

In order to store the temporal context in the time series, Chappell and Tayor [13] proposed a modification to the original SOM. This modified SOM, termed Temporal Kohonen Map (TKM), can not only function like a normal SOM but is also able to give context to vectors appearing in sequences. The main different between the TKM and the SOM is that the outputs of SOM are reset to zero after feeding of each input vector, while in the TKM the original outputs are replaced by leaky integrator outputs, which once activated, gradually lose their activity. The original TKM was then extended by Koskela, et al. [14] to a modified version called Recurrent Self-Organising Map(RSOM). RSOM moves the leaky integrator from the output into the input gives,

$$y_i(n) = (1-\beta)y_i(n-1) + \beta(x(n) - w_i(n)) \ . \tag{3}$$

$y_i(n)$ is the temporally leaked difference vector at map unit i, time n. Above $0 < \beta < 1$ is the leaking coefficient while $x(n)$ and $w_i(n)$ have their previous meanings. Large β corresponds to short memory while small β corresponds to longer memory. When $\beta = 1$, RSOM behaves like a normal SOM and on the other extreme when $\beta = 0$ all $w_i(n)$ equal to the mean of input vectors. After certain mathematical reformulation, $y_(n)$ end up with a similar form as an exponentially weighted linear FIR filter,

$$y(n) = \beta \sum_{k=1}^{n} (1-\beta)^{(n-k)} (x(k) - w) \ . \tag{4}$$

The updating rule of the map should include part of the history of the input, as the impulse response of the recurrent filter decay below 5% of its initial value after a few steps. For further explanation, please see [14]. The length of episode is 3 and $\beta = 0.95$ in the experiments.

The tree type RSOM has the similar mechanism as the Neural Gas [8]. Firstly the map starts with a main stem, an one dimensional RSOM chain. After certain iterations, the stem is trained to go through the backbone of the "density". According to the number of input vectors attracted by each neuron, the main stem starts to split into sub-chains at the neurons that attract more input vectors than a preset threshold. Also the length of sub-chain depends on how much the number of input vectors.

2.2 Support Vector Machine in Regression

SVM [10] is a novel regressor with three distinct advantages. First, SVM solves a risk minimisation problem by balancing the empirical error and a regularisation term, where the risk is measured by Vapnik's ε-insensitive loss function. Second, SVM usually estimates a set of linear functions defined in a high dimensional feature space. Third, SVM only uses a limited number of the support vectors.

Suppose we are given a training data set $\{(x1, y1), \ldots, (x_l, y_l)\} \subset \chi \times \mathbb{R}$, where χ denotes the space of the input patterns (e.g. $\chi = \mathbb{R}^d$). We try to do the prediction by estimating a linear function f

$$f(x) = w * \phi(x) + b \ . \tag{5}$$

$\phi(x)$ represents a nonlinear mapping of x in a high-dimensional feature space. The function f can be estimated by minimising a regularised risk function

$$\text{minimise} \quad \frac{1}{2} \parallel w \parallel^2 + C\frac{1}{l}\sum_{i=1}^{l} L_\varepsilon \ . \tag{6}$$

$$L_\varepsilon = \begin{cases} |y_i - w * \phi(x_i) - b| - \varepsilon, & |y_i - w * \phi(x_i) - b| \geq \varepsilon \\ 0, & \text{otherwise} \end{cases} \ . \tag{7}$$

The term $\parallel w \parallel^2$ is called the regularised term, which we want to make as flat as possible. The second term is the empirical error measured by Vapnik's ε-insensitive loss function. C is the regularisation constant. Transfer above two equations to the primal objective function as follows by introducing slack variables ξ_i^*, ξ_i

$$\text{minimise} \quad \frac{1}{2} \parallel w \parallel^2 + C\sum_{i=1}^{l}(\xi_i + \xi_i^*) \ . \tag{8}$$

$$\text{subject to} \quad \begin{cases} y_i - w * \phi(x) - b \leq \varepsilon + \xi_i \\ w * \phi(x) + b - y_i \leq \varepsilon + \xi_i^* \\ \xi_i, \ \xi_i^* \geq 0 \end{cases} \ . \tag{9}$$

Finally, by introducing Lagrange multipliers α_i, α_i^*, we can solve a linearly constrained quadratic problem. The decision function then has a explicit form

$$f(x) = \sum_{i=1}^{l}(\alpha_i - \alpha_i^*)K(x_i, x) + b \ . \tag{10}$$

where $K(x_i, x)$ is defined as the kernel function. The elegance of using the kernel function is that one can deal with arbitrary dimensional feature space without having to compute the nonlinear mapping function explicitly. Any function that satisfies Mercer's condition [10] can be used as the kernel function. This technique has also been implicitly used in SOMs, as the neighbourhood functions are equivalent to Gaussian kernel functions [15]. Since no prior knowledge of the data distribution is available, we use Gaussian kernel in SVM regressor.

Based on Karush-Kuhn-Tucker(KKT) conditions [16], only a number of coefficients $(\alpha_i - \alpha_i^*)$ will be nonzero, which corresponding to training data points referred to as support vectors. Approximation errors at the point of support vectors are larger than or equal to ε.

2.3 Two-Stages Architecture

The idea underlying the two-stage architecture is to firstly partition the whole input space into several non-overlapped regions and then use regression models to produce forecasting results on those partitioned regions. These regression models

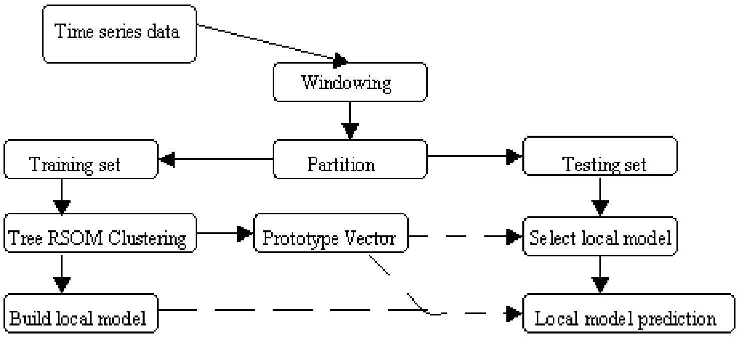

Fig. 1. A generalised scheme for constructing local models and their predictions

are called local models. A generalised scheme for constructing and evaluation the local models is depicted in Fig. 1.

The learning algorithm for the proposed two-stage architecture is outlined as follows:

1. Window the time series data into vectors. Separate into training, validation, testing parts.
2. Create a one dimensional chain with a number of terminal nodes and initialise it with random weights.
3. Recursively [1] training the Recurrent SOMs with the training data. Use the trained weight to partition input space into different regions.
4. Calculate the number of training data points in each regions. If the number of data points large than the preset threshold, expand the terminal node to a branch(a sub-RSOM chain) with at least two nodes on it. Otherwise, stop.
5. Repeat (3)-(4) until the partition cannot be preceded in any region.
6. Train SVM for the partitioned regions and find out the most adequate SVM based on the training data points in each partitioned region.

A validation set is used to early stop the RSOM in step (3) when the quantisation error of validation data starts to increase.

3 Experimental Results

We evaluate the proposed approach using real market data (exchange rate of U.S. Dollar to British Pound). The data is retrieved from the PACIFIC Exchange Rate Service provided by Prof. Werner Antwiler at UBCs Sauder School of Business. There are around 15 years' daily data excluding weekends and bank

[1] Only the initial weights are set randomly. The final weights of the first training phrase is brought to the second training phrase as its initial weights. Same as the learning rate, it is a continues variable rather than reset at the each time as a new submap is created.

holidays when currency markets were closed. We applied the price-return convert(i.e. $x'_n = \ln \frac{x_{n+1}}{x_n}$ here the x_n is the scalar values of the original data at the time n) to the original data. Considering the highly varying nature of the exchange rate, we train the neural networks with 3000 consecutive data points and we test the performance of the prediction method on the following 100 data points. Then, we window both of the training and testing sets with a window length of 13 to form two vector spaces. The window length is chosen empirically according to relatively good performances in SVM prediction.

We compare the various modeling and prediction methods on three measurements. The first is the Mean Square Error between target time series and predicted ones. The second measurement is the correct prediction percentage, which is a criterion to check whether the prediction is made on the right direction (i.e. we calculate how many percents predicted returns have the same sign as their corresponding actual returns). The third measurement is called profit earned, which is a program based on a virtual real trader's reaction in a real market. A set of rules have to be pre-set, such as if the increasing percentage is larger than a threshold, we invest certain amount of money to buy. If either increase or decrease is less than another threshold, no action is taken.

Fig. 2 shows the results of predictions of difference methods. The first sub-figure shows the prediction of the proposed local model method combining Recurrent SOM and SVM. The second sub-figure shows the local model method

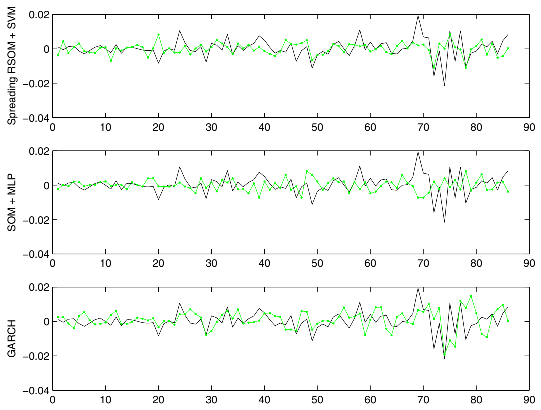

Fig. 2. The prediction results of three different approaches, the solid lines represent the original time series and the dot lines represent the predicted curves

Table 1. Total errors of various methods on the test set

	RSOM+SVM	SOM+MLP	GARCH
Mean Square Error (e-005)	3.0886	3.0999	3.9857
Correct Prediction (%)	67.5275	62.7907	46.6667
Profit earned (%)	7.20	4.29	5.56

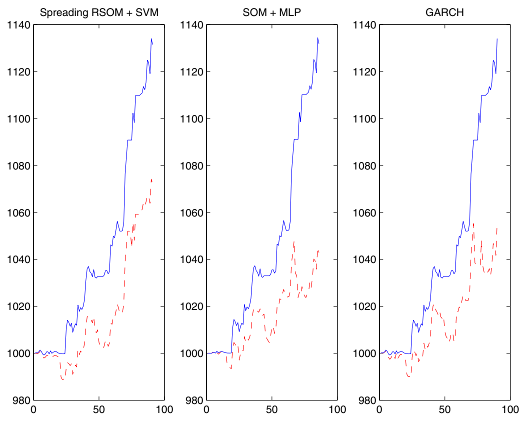

Fig. 3. The profits made based on three approaches. The closer the dash line is to the solid line, the better the performance. The solid line is the best profit curve, which means in case that prediction is 100% correct, the best performance a trader can achieve based on the strategy. The dash lines are the profits actually achieved.

combing SOM and MLP [2], and the third shows GARCH model prediction. Fig. 3 shows the profits made on these predictions of different approaches. Table 1 shows the measurements of three methods. The results of the first two methods are average of several experiments, because of the random initially weights and randomly feed in input vectors sequence.

4 Conclusion

Based on the principal of "divide-and-conquer", the proposed two-stage hybrid system outperforms other traditionary modeling techniques. There are a few inherent advantages of this approach. Firstly, since the prediction is based on smaller sets of data points in local models, it is thus possible to choose the best SVM for each individual model. Secondly, SOM is inherently an efficient clustering technique as it has a neighbourhood structure which can prevent the clustering being trapped to local minima. Thirdly, the convergence speed of SVMs can be greatly increased as the number of support vector decreases in smaller data regions (local models).

The future work includes alternative validation methods such as entropy [12] and nongaussian measurement [17] which can be used as a self-validation method. In this paper, we use Gaussian kernels as we know little about the distribution of exchange rate data. We will try to use more appropriate kernels if we can explore more on the data distribution in the future. Furthermore, we will also extend this work to multi-step prediction and consider bid-offer margins in profit estimation.

[2] The MLP has linear activation function and a number of inputs equals to the window size of the RSOM with one output.

References

1. Hamilton, J.: Time Series Analysis. Princeton University Press (1994)
2. Engle, R.: Autoregressive Conditional Heteroskedasticity with Estimates of The Variance of United Kingdom Inflation. Econometrics 50 (1982) 987-1007
3. Bollerslev, T.: Generalized Autoregressive Conditional Heteroskedasticity. Journal of Econometrics 31 (1986) 307-327
4. Kim, T.Y.,Oh, K.J., Kim, C., Do, J.D.: Artificial Neural Networks for Non-stationary Time Series. Neurocomputing 61 (2004) 439-447
5. Vesanto, J.: Using The SOM and Local Models in Time-Series Prediction. Helsinki University of Technology. (1997) B15
6. Cao, L.J.: Support Vector Machines Experts for Time Series Forecasting. Neurocomputing 51 (2002) 321-339
7. Koskela, T.: Time Series Prediction Using Recurrent SOM with Local Linear Models. Helsinki University of Technology (2001)
8. Martinetz, T.M., Berkovich, S.G., Schulten, K.J.: "Neural-Gas" Network for Vector Quantization and Its Application to Time Series Prediction. IEEE Transactions on Neural Networks 4 (1993) 558-569
9. Kohonen, T.: Self-Organizing Maps. Springer, Berlin (1995)
10. Vapnik, V.: The Nature of Statistical Learning Theory. Springer, New York (1995)
11. Walter, J., Ritter, H., Schulten, K.: Nonlinear Prediction with Self-Organizing Maps. Beckman-Institute and Department of Physics, UIUC, Urbana.
12. Freeman, R.T., Yin, H.: Adaptive Topological Tree Structure for Document Organisation and Visualisation. Neural Networks 17 (2004) 1255-1271
13. Chappell, G.J., Taylor, J.G.: The Temporal Kohonen Map. Neural Networks 4 (1993) 441-445
14. Koskela, T., Varsta, M., Heikkonen, J., Kaski, K.: Time series Prediciton Using Recurrent SOM with Local Linear Model. Report B15, Lab. of Computational Engineering, Helsinki University of Technology, Oct (1997)
15. Yin, H.: Self-Organising Map as A Natural Kernel Method. Proc. ICNN&B'05, Beijing, China, 13-15 Oct. (2005)
16. Kuhn, H.W., Tucker, A.W.: Nonlinear Programming. Proc. 2nd Berkeley Symposium on Mathematical Statistics and Probabilistics. 481C492, (1951) University of California Press, Berkeley
17. Hyvarinen, A., Oja, E.: Independent Component Analysis: Algorithms and Applications. Neural Networks 13(4-5) (2000) 411-430