TECHNIQUES FOR PREDICTION IN CHAOS – A COMPARATIVE STUDY ON FINANCIAL DATA

Laurenţiu BUCUR[1], Adina FLOREA[2]

Tehnicile de predicţie a seriilor de timp utilizând reţele neuronale sau elemente de geometrie analitică reprezintă două clase specializate de metode aplicabile semnalelor caracterizate de haos determinist. În paralel cu dezvoltarea acestor metode, progresele în domeniul învaţării statistice au impus metodele kernel de regresie neliniară ca metode de ultimă generaţie. În acest articol sunt comparate performanţele de predicţie ale celor mai reprezentativi algoritmi din cele trei clase de metode, cu accent pe maşinile kernel rare. Sunt efectuate experimente de predicţie pentru: Volatility Index (VIX), Put-to-Call ratio şi preţul zilnic de închidere al unor perechi valutare.

Time series prediction techniques using neural networks or elements of analytic geometry represent two specialized classes of methods applicable for signals which exhibit deterministic chaos. In parralel with the development of these methods, progress in statistical learning theory imposed kernel methods for nonlinear regression as the state of the art. In this paper the prediction performance of the most representative algorithms from the three classes are analysed, with focus on sparse kernel machines. The following time series are analysed:The Volatility Index (VIX), Put-to-Call ratio the daily closing price series of a basket of currencies.

Keywords: time series prediction, chaos, nearest neighbours, nearest trajectory, kernel methods, sparse kernel machines, VIX, put-to-call ratio, foreign exchange

1. Introduction

Time series prediction techniques using elements of deterministic chaos and analytic geometry constitute a special class of algorithms which base their predictions on local linear estimates of the vector field in the phase space of an unknown generating signal source. A second class of algorithms use the same elements of chaos theory for data pre-processing and artificial neural networks for nonlinear prediction of the same unknown evolution function in chaos. The third class of methods in statistical learning theory is constituted by kernel methods, as described in [5], [6] and [8]. This paper aims at comparing the predictive

[1] PhD student, Faculty of Automatics and Computers, University POLITEHNICA of Bucharest, e-mail: laur.bucur@gmail.com
[2] Prof., Faculty of Automatics and Computers, University POLITEHNICA of Bucharest, Romania, e-mail: adina.florea@cs.pub.ro

performance of each of the three classes of methods, by using the most representative algorithms in each category, on a large financial dataset. The contribution of the paper results from a lack of comparative study in the literature of the methods used herein and a lack of analysis on the predictability of the time series under investigation using a significant body of prediction methods. The main task set forth in this study is to examine the accuracy of each prediction method for time series with a strong stochastic component (noise), understood as variability not explained by the lagged values of the time series as endogenous variables.

From the first category of algorithms which use methods of deterministic chaos for pre-processing and analytic geometry for prediction, we used the Nearest Neighbours approach of Hannias and Karras [3] (herein reffered to as *Nearest -Neighbours*) with linear estimates and the nearest trajectory algorithm of McNames [4] with an exponential weighted Mallahobis metric (herein reffered to as *Nearest-Trajectory*).

From the second category of algorithms which use deterministic chaos and neural networks, we used the Multiple Backpropagation Feedforward Neural Networks for global modelling in conjunction with the approach of Mori and Urano ([1]) (herein reffered to as *MBP-Global*) and with the nearest neighbour algorithm of Hannias and Karras ([3]) (herein reffered to as *MBP-Local*).

From the class of kernel methods for nonlinear time series prediction we applied kernel ridge regression [5] (herein reffered to as *Kernel-Ridge),* Sparse Kernel Machines using attractors as described in [6] (herein reffered to as *SKM-Attractors)* and the Support Vector Machine for Regression as described in [8] (herein reffered to as ε-*SVR*). All experiments were performed on the minimum delay embedding phase space of each time series.

The following time series were analysed:

- the Volatility Index (VIX) [10] (Feb.1st 2004 – December 21st 2010) and the put-to-call ratio [11] (October 17th 2003 – December 21st 2010). Both indexes indicate volatility and the ratio between put and call options traded on the Chicago Board of Options Exchange (CBOE).

- a basket of commonly traded currency pairs [12] on the foreign exchange interbank market (FOREX).

The peformance of the algorithms was measured in terms of directional predictive accuracy and root mean squared error (RMSE).

2. Experimental setup

For each time series, the following data sets were generated in the pre-processing stage:

Iteration 1: no detrending, application of the prediction methods on the raw delay-embedded time series.

Iteration 2: stochastic detrending (1) of order 1, application of the False Nearest Neighbours algorithm and delay embedding on the difference time series.

Iteration 1 was used for testing the hypothesis that the generating signal source is stationary and its dynamics (vector field) can be approximated using nonlinear tools.

Iteration 2 was used to reduce possible Auto-Regressive AR(1) components and to eliminate the effects of possible non-stationaruty and spurious regression in the original samples, as suggested in [13].

Table 1 summarizes the datasets and their stochastic detrended variants used in this comparative study.

Table 1

Datasets

Data set no.	Original series (symbol and time frame)	Order of stochastic detrending	Minimum Embedding Dimension	Training / Testing samples
1	VIX Daily	0	5	871/872
2		1	6	870 / 871
3	Put-To-Call ratio Daily	0	6	897 / 898
4		1	5	897 / 898
5	EUR/USD 4h	0	6	6728 / 6728
6		1	8	6726 / 6727
7	GBP/USD daily	0	6	2600 / 2601
8		1	6	2600 / 2600
9	GBP/USD 4h	0	7	6954 / 6954
10		1	8	6953 / 6953
11	USD/CAD Daily	0	5	1596 / 1597
12		1	6	1595 / 1596
13	USD/CAD 4h	0	7	8251 / 8252
14		1	8	8250 / 8251
15	AUD/USD Daily	0	6	2729 / 2729
16		1	6	2728 / 2729
17	GBP/CHF Daily	0	4	2057 / 2057
18		1	7	2055 / 2055
19	GBP/CHF 4h	0	6	8115 / 8115
20		1	8	8113 / 8114

For each data set:

- the minimum delay embedding was calculated with the False Nearest Neighbours method in [9] for reconstructing the attractor of the deterministic component.

- the samples were delay embedded with the minimum embedding dimension using (2) and each delay embedding sample was assigned the desired output (3). This approach is used for integrated prediction, as suggested by McNames in [4].
- the resulting data set was split in 50% training data and 50% test data.
- the full set of time series prediction methods described in section 1 was used. The performance of each method was measured in terms of RMSE and directional sign prediction of each sample label (3).

In determining the minimum embedding dimension, the last value m for which the fraction of false nearest neighbours in the m-dimensional delay embedding space was greater than 0 was chosen, as suggested in [9].

Stochastic detrending (1) of a non-stationary time series is a method proposed in [13] to eliminate the effect of spurious regression. In [14] we use it to eliminate possible non-stationarity and increase the statistical support of patterns in the phase space of the difference time series by using stochastic detrending of order 1 (τ=1). In general, for continuous time systems:

$$z(t) = x(t) - \frac{1}{\tau}\sum_{j=1}^{\tau} x(t-\tau) \tag{1}$$

where τ is the order of stochastic detrending.

This paper uses (1) by taking the finite sampling of x(t) as x[n]=x(nθ) as in (2) to obtain z[n]. An m-dimensional delay embedding of a continuous time signal x(t) is the m-dimensional vector:

$$Y(t) = \{\, x(t), x(t-\theta), x(t-2\theta), x(t-3\theta), \ldots x(t-(m-1)\theta) \,\} \tag{2}$$

where θ is the sampling period. In this study, θ corresponds to a daily or a four hour period for the financial instruments in Table 1. For each time series x(t) (Table 1- column 1) and each m-dimensional (Table 1- column 6) delay embedding sample (2), its corresponding desired output was assigned to

$$DesiredOutput(t) = \{\, x(t+\theta) - x(t) \,\} \tag{3}$$

which corresponds to the one-step ahead differential in the original time series.

For each time series $x_i[n]$, n=0..N_i (Table 1-column 1), the set PredictionSet(i)={(Y[n],DesiredOutput[n])}$_{n=1..N}$ was created, where:

$$Y[n] = Y(n\theta) \tag{4}$$

$$DesiredOutput[n] = DesiredOutput(n\theta) \tag{5}$$

The complete data sets used in this study are published at [15].

Supervised training was performed on the first 50% samples of the set and performance was measured against the remaining half of the data.

3. Class 1 algorithms performance

In the first experiment we tested the *Nearest-Neighbours* and the *Nearest-Trajectory* algorithms for time series prediction. They perform a weighted approximation (8) of the evolution of the current state in phase space using the biweight function (9). For the *Nearest-Neighbours* method, the number of nearest neighbours k was optimized over the training data. Trial values for k were between 2 and 10. The same optimization was performed for the *Local-Trajectory* algorithm together with the λ parameter (values between 0.1 and 0.9 in 0.1 increments) . The λ parameter (0<λ<1) defines an exponential weighted Mallahobis metric ([4]):

$$d_{\Lambda}(Q_t, X_t) = (Q_t - X_t)^T \Lambda (Q_t, X_t) \tag{6}$$

where:

$$\Lambda_{i,j} = \begin{cases} \lambda^{i-1}, i = j \\ 0 \end{cases} \tag{7}$$

Both methods use iterated prediction in the form:

$$\text{PredictedOutput} = \frac{\sum_{i=1}^{k} q_i DesiredOutput_i}{\sum_{i=1}^{k} q_i} \tag{8}$$

where:

$$q_i = (1 - \frac{d_i^2}{d_{\max}^2})^2 \tag{9}$$

d_i – the distance between the current observation in the minimum delay embedding phase space and its i-th nearest neighbour using (6)

d_{max} – the maximum distance to any of the k nearest neighbours of the current observation in the minimum delay embedding phase space.

Table 2 contains the out-of-sample prediction performance for *Nearest-Neighbours* and *Nearest-Trajectory*. Column 3 contains the optimal value for k determined over the training set in terms of minimization of the RMSE. Column 4 contains the optimal value of λ for the *Nearest-Trajectory* algorithm. Columns 5,6,7 and 8 contain the prediction performance of the algorithms over the test sets.

Table 2

Class 1 prediction algorithms performance: Nearest-Neighbours and Nearest-Trajectory

Data set no.	Original time series	Optimal k	Optimal λ	Nearest-Neigbours		Nearest-Trajectory	
				RMSE	Directional accuracy (%)	RMSE	Directional accuracy (%)
1	VIX	6	0.6	8.46	46.31	**7.29**	**46.94**
2		4	0.7	3.44	46.97	**3.23**	**50.2**
3	Put-to-call ratio	4	0.9	0.65	**48.36**	**0.60**	46.97
4		3	0.8	0.78	**52.26**	**0.69**	47.61
5	EUR 4h	9	0.9	**0.004**	49.57	**0.004**	**50.22**
6		5	0.5	**0.0043**	**50.45**	**0.0043**	50.03
7	GBP/USD Daily	7	0.8	0.0104	49.4	**0.0101**	**51.55**
8		5	0.5	0.0123	50.79	**0.0114**	**51.16**
9	GBP/USD 4h	10	0.9	0.0055	**50.6**	**0.0054**	49.4
10		5	0.1	**0.0058**	49.11	0.0060	**50.8**
11	USD/CAD Daily	6	0.2	**0.0062**	**51.54**	0.0068	48.98
12		4	0.6	0.0091	49.37	**0.0086**	**51.01**
13	USD/CAD 4h	10	0.9	0.0037	**49.93**	**0.0036**	49.38
14		5	0.5	**0.0039**	**50.25**	0.0040	49.49
15	AUD/USD Daily	7	0.8	0.0046	**50.78**	**0.0045**	49.81
16		5	0.1	0.0074	**52.37**	**0.0071**	51.75
17	GBP/CHF Daily	6	0.7	0.0115	49.33	**0.0112**	**51.08**
18		4	0.5	**0.0120**	**51.40**	0.0121	50
19	GBP/CHF 4h	9	0.9	0.0045	49.05	**0.0041**	**50.52**
20		5	0.8	**0.0060**	48.74	0.0061	**49.52**

Both the *Nearest-Neighbours* and *Nearest-Trajectory* algorithms fail to provide a statistically significant effect size for the prediction of directional change (columns 5 and 7). However, the *Nearest-Trajectory* algorithm is superior to the *Nearest-Neighbours* approach in terms of RMSE in 15 out of 20 trials while in terms of accuracy in only outperforms the *Nearest-Neighbours* algorithm in 10 out of 20 trials.

4. Class 2 algorithms performance

In [1], Mori and Urano use a different approach for time series forecasting. In the preprocessing phase the minimum embedding dimension is chosen the value *d* for which the largest Lyapunov value of the d-dimensional delay embedding reaches a maximum. They use delay embedding and Multi-Layer Perceptrons without stochastic detrending for predicting the short-term load forecasting in power systems. In [3] Hannias and Karras assume a more general form of (8):

$$\text{PredictedOutput} = f(q_i, DesiredOutput_i, q'_j\ DesiredOutput_k) \quad (10)$$

where:

- q_i, i=1..k are the normalized weighted distances (9) to the k nearest neighbours and $DesiredOutput_i$ are their attached labels
- q_j, j=1..k' are the normalized weighted distances (9) to the k *furthest* neighbours and $DesiredOutput_j$ are their attached labels
- f is an unknown deterministic map as function of the distances to the k nearest and k' furthest neighbours and their attached labels.

In [3] Hannias and Karras contribute by taking into account the distances to the k' furthest neighbours, eliminating non-related trajectories from selection and using a Multiple Layer Perceptron (MLP) for learning the unknown function (10) instead of the linear form (8) which uses only the k nearest neighbours. They use the MLP approach for multi-step nonlinear prediction of voltage in a diode resonator circuit.

The approach of Mori and Urano does not assume the form (10) of the unknown map of the deterministic system and use the neural networks directly in the delay embedding phase space of the raw time series. The use of MLPs in [1] in the present study is herein reffered to as *MBP-Global*.

The approach of Hannias and Karras assumes the specific form (10) using local phase space information. The approach in [3] is herein reffered to as *MBP-Local*.

In applying the *MBP-Global* and *MBP-Local* prediction algorithms in the present study, we used the Multiple-Layer Backpropagation Neural Network model [16] (MBP) with selective activation.

We trained each model on the first 50% of the samples and tested their performance on the second half of the delay embedding samples, identically to the experiments on the first class of algorithms.

The topology of each neural network used was chosen as follows:

- one input layer with m inputs, m being the minimum embedding dimension of the time series
- one hidden layer with 2m + 1 units
- one hidden layer with 4m+3 units
- one output layer with one unit

where m is the minimum embedding dimension of the time series.

The topology was chosen based on a theorem introduced by Pinkus [17] and used by Pavdilis and Tassoulis in [9] for prediction of foreign exchange using k-windows clustering and artificial neural networks. The theorem is included here for completeness.

Theorem (Pinkus): *On the unit cube in R^n any continuous function can be uniformly approximated, to within any error by using a two hidden layer neural network having 2n+1 units in the first layer and 4n+3 units in the second layer.*

Without loss of generality, the initial formulation of the theorem reffers to the unit cube only for convenience. In [18], Pinkus and Maiorov state the results hold for any compact subset in R^n.

In assessing the performance of *MBP-Global* and *MBP-Local* algorithms we used the datasets in table 1, their minimum embedding dimension m together with the MBP neural network model and the topology given by the Pinkus theorem, as mentioned above. For the *MBP-Local* approach we used k'=0 and k as specified in Table 2 – column 3.

- Inputs for the neural network using *MBP-Global* are the delay embedding samples of the time series, identical to the class 1 algorithms.
- Inputs for the neural network using *MBP-Local* are the weights q_i, i=1..k as specified in (10) and the attached labels $DesiredOutput_i$.

In all cases, the desired output from the neural network is (3).

Table 3 summarizes the performance of *MBP-Global* and *MBP-Local* in terms of RMSE and directional predictive accuracy over the test sets.

Table 3

Class 2 prediction algorithms performance: MBP-Global and MBP-Local

Data set no.	Original time series	MBP-Global		MBP-Local	
		RMSE	Directional accuracy (%)	RMSE	Directional accuracy (%)
1	VIX	3.299	**56.12**	**2.763**	52.31
2		**2.535**	**54.12**	2.779	44.80
3	Put-to-call ratio	**0.302**	**63.80**	0.383	48.72
4		**0.269**	**68.81**	0.385	48.25
5	EUR 4h	**0.0037**	48.65	0.0051	**50.94**
6		**0.0036**	**50.36**	0.0047	48.63
7	GBP/USD Daily	**0.0098**	**49.75**	0.0102	48.53
8		**0.0097**	**51.51**	0.0109	51.14
9	GBP/USD 4h	**0.0049**	**50.72**	0.0060	48.56
10		**0.0049**	**50.97**	0.0051	48.28
11	USD/CAD Daily	**0.0074**	**50.97**	0.0075	49.11
12		**0.0073**	**52.75**	0.0077	49.21
13	USD/CAD 4h	**0.0034**	**48.88**	0.0037	48.10
14		**0.0034**	**49.02**	0.0036	48.05
15	AUD/USD Daily	**0.0060**	**47.43**	0.0085	46.06
16		**0.0060**	**47.12**	0.0062	46.60
17	GBP/CHF Daily	**0.0114**	**50.53**	0.0130	48.63
18		**0.0115**	49.80	0.0121	**50.32**
19	GBP/CHF 4h	**0.0051**	48.86	0.0105	**49.60**
20		**0.0051**	**50.35**	0.0061	48.99

Table 3 shows superior performance of *MBP-Global* in comparison to *MBP-Local* in terms of RMSE and directional accuracy. *MBP-Global* - the approach of Mori and Urano ([1]) proves superior to *MBP-Local* - the approach suggested by Hannias and Karras ([3]) because the latter loses phase space location information. In the former case, the location in phase space is used to estimate the geometry of the vector field, while in the latter only normalized distances from a point in phase space in relation to its k nearest neighbours are provided as inputs to the neural network, while the actual position in phase space is lost. *MBP-Global* outperforms *MBP-Local* in 19 out of 20 trials in terms of RMSE and in 18 out of 20 trials in terms of directional accuracy. *MBP-Global* detects weak directional predictability in phase space for the VIX data series and strong directional predictability for the put-to-call ratio while all previous methods fail. For the currency time series 5…20 both *MBP-Global* and *MBP-Local* fail to provide a statistically significant directional predictive advantage. However, the consistent superior performance of *MBP-Global* over *MBP-Local* implies that price motion is not random and has a deterministic component, otherwise in the random walk hypothesis a consistent superior performance of one method over another would not have been possible.

5. Class 3 algorithms performance

From the third class of algorithms for prediction - kernel methods, the following algorithms were evaluated:

- Kernel ridge regression [5] (*Kernel-Ridge)*
- Sparse kernel machines using attractors [6] (*SKM-Attractors*)
- The Support Vector Machine for Regression using the ε-insensitive Loss Function [8] (ε-SVR).

Kernel-Ridge

Given a labeled training set S =$\{ (x_1,y_1), (x_2, y_2),\ldots (x_N,y_N)\}$ Kernel Ridge Regression uses pattern functions of the form:

$$f(x) = \sum_{i=1}^{N} \alpha_i K(x_i, x) \tag{11}$$

where:

- K is a kernel function. In this study we used the isotropic Gaussian kernel:

$$K(x, x_i) = e^{-\frac{\|x-x_i\|^2}{2\sigma^2}} \tag{12}$$

- $\sigma > 0$ – the kernel parameter
- N – the number of training samples
- α_i, i=1..N – the dual weights of the pattern function

Given a positive real constant λ called the regularization constant, the weight vector $\alpha = (\alpha_1, \alpha_{2,\ldots} \alpha_N)^T$ is calculated using:

$$\alpha = (Kern + \lambda I_N)^{-1} \tag{13}$$

where *Kern* is the N x N kernel matrix calculated using the set S:

$$Kern_{i,j} = K(x_i, x_j) \tag{14}$$

In calculating the weight vector α, the right hand side of (13) is a positive definite matrix. In this study we calculated the inverse using the Conjugate-Gradient algorithm.

In determining the optimal values of λ and σ we used the following partition of the data sets:

- 25% of the samples – the training set - are used to define (11)
- 25% of the samples – the cross-validation set – used to optimize λ
- 50% of the samples are used as the test set, identical to the test sets in the class 1 and class 2 experiments.

Using the cross-validation set the isotropic kernel parameter σ was optimized from 0.0001 to 0.1 using 20 increments on a log-linear scale. The optimal values in terms of number of correct directional predictions over the test and cross-validation sets combined were used in testing the kernel machine (11) out-of-sample (Table 4 - columns 3,4 and 5).

Support Vector Machines for Regression(ε-SVR)

The Support Vector Machine for Regression [8] is a function of the form (11) with the additional property of sparseness. The weights are represented in the dual form:

$$\alpha_i = a_i - \hat{a}_i \tag{15}$$

Such as to maximize :

$$L(a, \hat{a}) = -\frac{1}{2}\sum_{i=1}^{N}\sum_{i=1}^{N}(a_n - \hat{a}_n)(a_m - \hat{a}_m)K(x_n - x_m) + \sum_{i=1}^{N}(a_n - \hat{a}_n)t_n \tag{16}$$

The optimum satisfies the following constraints:

$$0 \le a_n \le C / N \tag{17}$$

$$0 \le \hat{a}_n \le C / N \tag{18}$$

$$\sum_{n=1}^{N} a_n - \hat{a}_n = 0 \tag{19}$$

$$\sum_{n=1}^{N} (a_n + \hat{a}_n) \le \nu C \tag{20}$$

where C is a weight bounding constant chosen by the user and $\nu \in [0,1]$ is a fraction of samples allowed to lie outside the ε-tube. In [8] the Support Vector Machines for Regression are shown to possess the sparsity property. The property of sparsity is ensured by the ε-insensitive loss:

$$L(f(x_i) - y_i) = \begin{cases} 0, if \mid f(x_i) - y_i \mid < \varepsilon \\ \mid f(x_i) - y_i \mid - \varepsilon \end{cases} \tag{21}$$

All points x_i which are in the ε-tube $| f(x_i) - y_i | < \varepsilon$ have zero dual weights, while the remaining set are support vectors of (11).

In training the Support Vector Machine for Regression using the ε-insensitive loss we used the LIBSVM [20] package by training and testing the model on the same datasets as the class 1 and class 2 algorithms. The performance of the Support Vector Machine for Regression is shown in Table 4 - columns 9 and 10.

Sparse Kernel Machines using attractors

From the third class of algorithms we used the Sparse Kernel Machine model of Lee, Jung and Lee [6]. Training starts with the fully featured kernel machine (11) trained with the Ridge Regression algorithm. The aim is to obtain a simplified version:

$$g(x) = \sum_{n=1}^{M} \beta_i K(z_i, x) + c \tag{22}$$

where:

- c, $\beta_1 \in R$ and $x \in X$ a compact set, $X \subseteq R^n$, M<<N and $z_i \in X$, i=1..m
- $\{z_i\}_{i=1,M}$ is the Reduced Set (RS) of g(x)

The Reduced Set is the set of M attractors of the dynamical system:

$$\rho(x) = \| f(x) - \Phi(x) \|^2 = \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j K(x_i, x_j) + K(x,x) - 2\sum_{j=1}^{N} \alpha_i K(x_i, x) \tag{23}$$

$$\frac{dx}{dt} = -\nabla \rho(x) \tag{24}$$

where f (11) is specified and previously trained with Kernel Ridge Regression.

The sparse kernel machine (22) weights are:

$$\beta = (K^{zz})^{-1} K^{zx} \alpha \tag{25}$$

where:

- $\alpha = (\alpha_1, \alpha_2, ..., \alpha_N)^T$ - the weight of (11) except the bias term
- K^{zz} is an M x M matrix, $(K^{zz})_{ij} = K(z_i, z_j)$
- K^{zx} is an M x N matrix , $(K^{zx})_{ij} = K(z_i, x_j)$
- $S=\{(x_i,y_i)\}_{i=1..N} \subseteq X$ is the training set and $\{x_i\}_{i=1..N}$ are the training points of *f* in (11)

In [6] the authors do not provide an objective method for identifying attractors. They also do not provide any error bounds after the kernel machine simplification relative to the error bounds of the original kernel machine *f* (11).
In this study we used the following procedure:

- Kernel ridge regression was performed using the test and cross-validation sets using the kernel parameter σ in Table 4- column 5.
- Each kernel was descent toward its basin of attraction
- Kernel ridge regression was performed again using the same sets.
- ε-insensitive kernel elimination was performed for kernels clustered around attractors:
    - For each kernel k:
        - if the desired output at the center of k was predicted by the kernel machine within ε-tolerance then it was marked for deletion since it did not represent a support vector for regression.
- after the loop all kernels marked for deletion are removed to reduce the Rademacher complexity [5] of the kernel machine.
- The remaining kernels form the Reduced Set and represent support vectors for regression
- The kernel machine is retrained using Kernel Ridge Regression
-

The performance of the resulting sparse kernel machine was tested on the test set. In the above procedure, instead of setting or optimizing ε at each iteration we used ε = 10% of the best RMSE obtained by other methods as an objective error measure. The reduction rate in the number of kernels following this procedure is shown in Table 4 – column 8 and is rather low due to the presence of noise in data and the small neighbourhood size given by the kernel parameter σ. Columns 6 and 7 illustrate the performance of the resulting sparse kernel machines.

Table 4

Class 3 prediction algorithms performance

Data set no.	Original time series	Kernel-Ridge			SKM-Attractors			ε-SVR	
		RMSE	Directional accuracy (%)	Best σ	RMSE	Directional accuracy (%)	Reduction rate (%)	RMSE	Directional accuracy (%)
1	VIX	**1.247**	49.45	0.069	1.294	54.32	12.66	2.593	**55.55**
2		**1.625**	53.11	0.1	**1.625**	51.66	21.51	2.569	**55.04**
3	Put-to-call ratio	0.334	62.54	0.1	0.336	50.62	10.05	**0.273**	**67.26**
4		0.335	59.81	0.1	0.336	59.47	10.06	**0.268**	**68.59**
5	EUR 4h	**0.003**	**48.84**	0.004	n/a	n/a	15	0.0036	48.29
6		**0.003**	**50.56**	0.005	0.0036	49.89	19	0.0036	48.29
7	GBP/USD Daily	0.012	50.13	0.011	**0.0097**	**50.86**	6.91	0.0149	48.75
8		**0.009**	**49.71**	0.002	0.0097	48.20	9.34	0.0149	48.75
9	GBP/USD 4h	0.007	**50.58**	0.003	0.0098	49.20	7.08	**0.0060**	50.29
10		**0.005**	51.02	0.001	0.0096	**51.09**	4.75	0.0060	50.28
11	USD/CAD Daily	n/a*	n/a	0.008	**0.0078**	**50.91**	27.68	0.0081	48.65
12		0.008	50.88	0.003	**0.0073**	**51.32**	27.88	0.0081	48.68
13	USD/CAD 4h	n/a*	n/a	0.004	n/a	n/a	19.73	**0.0037**	**48.88**
14		**0.003**	**50.9**	0.007	0.3676	49.52	0	0.0037	48.88
15	AUD/USD Daily	0.007	49.19	0.004	0.0065	**51.07**	25.83	**0.0060**	46.48
16		**0.006**	**49.62**	0.006	**0.006**	49.31	21.79	**0.0060**	46.46
17	GBP/CHF Daily	0.014	49.84	0.008	**0.0125**	48.13	13.22	0.0140	**50.58**
18		0.017	50.41	0.011	**0.0113**	**50.8**	14.88	0.0140	50.63
19	GBP/CHF 4h	0.007	**50.26**	0.008	**0.0038**	50.09	13.58	0.0136	49.27
20		**0.005**	**50.77**	0.001	**0.005**	50.02	1.65	0.0136	49.26

* iteration statistically insignificant – 0 activations over the test set (low global mixing, \ nonstationary series)

6. Final performance comparison

Table 5 summarizes the performance of the three classes of algorithms for which the performacne is outlined in tables 2, 3 and 4. In this section we identify the best algorithm and its performance for each data set in terms of RMSE (columns 3 and 4) and directional predictive accuracy (columns 5 and 6).

Table 5

Performance summary – best prediction algorithms

Data set no.	Original series (symbol and time frame)	Best RMSE algorithm	Best RMSE	Best directional predictor	Best directional accuracy (%)

1	VIX Daily	Kernel-Ridge	1.247	MBP-Global	**56.12***
2		Kernel-Ridge, SKM-Attractors	1.625	ε-SVR	**55.04***
3	Put-To-Call ratio Daily	ε-SVR	0.273	ε-SVR	**67.26***
4		ε-SVR	0.268	MBP-Global	**68.81***
5	EUR/USD 4h	Kernel-Ridge	0.003	MBP-Local	50.94
6		Kernel-Ridge	0.003	Kernel-Ridge	50.56
7	GBP/USD Daily	SKM-Attractors	0.0097	SKM-Attractors	50.86
8		Kernel-Ridge	0.009	MBP-Global	**51.51****
9	GBP/USD 4h	MBP-Global	0.0049	MBP-Global	50.78
10		MBP-Global	0.0049	SKM-Attractors	**51.09****
11	USD/CAD Daily	Nearest-Neighbours	0.0062	Nearest-Neighbours	**51.548****
12		SKM-Attractors, MBP-Global	0.0073	MBP-Global	**52.75****
13	USD/CAD 4h	MBP-Global	0.0034	Nearest-Neighbour	49.93
14		ε-SVR, Kernel-Ridge	0.003	ε-SVR	50.9
15	AUD/USD Daily	ε-SVR	0.0045	SKM-Attractors	51.07
16		All class 3 algorithms, MBP-Global	0.006	Nearest-Neighbours	**52.37****
17	GBP/CHF Daily	Nearest-Trajectory	0.0112	Nearest-Trajectory	51.08
18		SKM-Attractors	0.0113	SKM-Attractors	50.8
19	GBP/CHF 4h	SKM-Attractors	0.0038	Nearest-Trajectory	50.52
20		SKM-Attractors, Kernel Ridge	0.005	Kernel-Ridge	50.77

* - strong predictability, greater or equal to 5% above chance

** - weak predictability: statistically significant effect size greater or equal to 1.4% above random chance, less than 3%

7. Conclusions

In this study we analysed the performance of three classes of nonlinear time series prediction methods developed over the past 20 years, from neural networks to kernel methods, with focus on sparse kernel machines (Lee, Jung,

Lee, 2009)[6]. The author provided his own implementation of:the *Nearest-Neighbours*, *Nearest-Trajectory, Kernel-Ridge* and *SKM-Attractors* and compared their performance to the MBP neural networks [16] and Support Vector Machines for Regression [20].

For the first class of algorithms, the Nearest-Trajectory algorithm proves superior to the linear Nearest-Neighbours algorithm in terms of RMSE and matches it in terms of directional prediction accuracy.

From the second class of algorithms, the approach of Mori and Urano [1] (*MBP-Global*) is consistently superior to the approach of Hannias and Karras [3] (*MBP-Local*) and provides a statistically significant predictive effect size for the VIX and put-to-call ratio time series. The consistent advantage obtained using *MBP-Global* over *MBP-Loca*l on a large body of data implies price motion has a deterministic and statistically stable component, otherwise both methods would have been comparable in performance. *MBP-Global* suggests location in phase space provides more predictive information than the normalized k nearest neighbour distances, as used by *MBP-Local*.

From the third class of algorithms, Kernel Ridge Regression, ε-SVR and Sparse Kernel Machines using attractors are comparable in terms of performance.

The best performance in column 6 of Table 5 illustrates a high degree of predictability of the directional change of the VIX and put-to-call ratio time series. For the rest of the time series, a very weak statistical effect size can be detected higher than 1.5% above chance only in some situations by sparse kernel machines using attractors, MBP-Global and Nearest-Neighbours algorithms.

For the strong deterministic components found in the VIX and put-to-call ratio time series, the ε-SVR sparse kernel model is comparable to the MBP neural network approach *MBP-Global* using the results of the Pinkus theorem.

R E F E R E N C E S

[1] *H. Mori, S. Urano,* "Short Term Load Forecasting with Chaos Time Series Analysis", in Proceedings of the International Conference on Intelligent Systems Applications to Power Systems, 1996 (ISAP '96)

[2]. *P. Cvitanovi, C.R. Artuso, P. Dahlqvist, R. Mainieri, G. Tanner, G. Vattay, N. Whelan, A. Wirzba*, "Classical and Quantum Chaos", [Available online at]: http://chaosbook.org/] [Last accessed: 06.11.2010]

[3] *M.P. Hanias, D.A. Karras,* "Improved Multistep Nonlinear Time Series Prediction by applying Deterministic Chaos and Neural Network Techniques in Diode Resonator Circuits", in: IEEE International Symposium on Intelligent Signal Processing, 2007. WISP 2007

[4] *J. McNames,* "A Nearest Trajectory Strategy for Time Series Prediction", in: Proceedings of the International Workshop on Advanced Black-Box Techniques for Nonlinear Modeling, pages 112–128, Katholieke Universiteit Leuven, Belgium, July 1998

[5] *J.S. Taylor, N. Cristianini*, "Kernel Methods for Pattern Analysis", Cambridge University, Press, 2004, ISBN 978-0-521-81396-6

[6] *D. Lee, K.H. Jung, J. Lee*. "Constructing Sparse Kernel Machines using Attractors", in: IEEE, Transactions on Neural Networks, **Vol. 20**, No.4, April 2009
[7] *J.M. Nese,* "Quantifying local predictability in phase space", in: Physica D., Nonlinear phenomena, Volume 35, Issues 1-2, April 1989, Pages 237-250
[8] *C.M. Bishop,* "Pattern Recognition and Machine Learning"; Springer Science and Bussiness Media, 2006, ISBN-10: 0-387-31073-8
[9] *N.G. Pavdilis, D.K. Tasoulis, M.N. Vrahatis*, "Financial Forecasting Through Unsupervised Clustering and Evolutionary Trained Neural Networks", In: Proceedings of the 2003 Congress on Evolutionary Computation, **Vol.4**, 2003, p. 2314-2321
[10] CBOE VIX Historical price data. [Available online at: http://www.cboe.com/publish/ScheduledTask/MktData/datahouse/vixcurrent.csv] [Last accessed: 10.12.2010]
[11] CBOE exchange volume and put-to-call ratio. [Available online at: http://www.cboe.com/publish/ScheduledTask/MktData/datahouse/totalpc.csv]
[12] *L. Bucur*. Experimental currency data sets.[Online] .Available at: https://docs.google.com/leaf?id=0B7VYFkQ0d6D-N2ZjMmJlYjctNzVjZC00NDE1LTkwZDgtYzg0NDdmZTYwNzVh&hl=en. [Last accessed: 10.12.2010]
[13] *W.E. Ferson, S. Sarkissian, T. Simin*, "Is Stock Predictability Spurious?", In: Journal of Investment Management, **Vol.1**, No.3, (2003), p. 1-10. [Available online - at: http://timsimin.net/Papers/PINT.PDF] [Last accessed: 11.11.2010]
[14] *L. Bucur, A.M. Florea,* "Exploring Chaos with Sparse Kernel Machines", Proc. of SYNASC 2010, the12th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, Timisoara, Romania, September 23-26, 2010
[15] *L. Bucur*, Datasets for the present study. [Available online at: https://docs.google.com/leaf?id=0B7VYFkQ0d6D-ODIzMzBlY2EtMDA2Yi00M2NiLTg4YWQtY2NiZGUzNDNkNTBh&hl=en] [Last accessed: 25.12.2010]
[16] *N.Lopes, B. Ribeiro*, "An efficient gradient-based learning algorithm applied to neural networks with selective actuation neurons", In: Neural, Parralel & Scientific Computations, **Vol.11**, Issue 3, pp. 253-272
[17] *A. Pinkus*. "Approximation theory of the mlp model in neural networks", in Acta Numerica (1999), 143-195
[18] *V. Maiorov, A. Pinkus,* "Lower Bounds for Approximation by MLP Neural Networks", [Available online at: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.6.5161] [Last accessed: 31.12.2010]
[19] *L.A. Purcina, S.F.P. Saramago*. "Differential Evolution Applied to the Solution of Large Linear Systems". [Available online at: http://www.engopt.org/nukleo/pdfs/0146_saramago.pdf] [Last accessed: 24.11.2010]
[20] LIBSVM (Library for Support Vector Machines). [Available online at: http://www.csie.ntu.edu.tw/~cjlin/libsvm/] [Last accessed: 9.1.2011].