International Journal of Bifurcation and Chaos, Vol. 16, No. 7 (2006) 2053–2062
© World Scientific Publishing Company

COMPUTATIONAL INTELLIGENCE METHODS FOR FINANCIAL TIME SERIES MODELING

N. G. PAVLIDIS*, D. K. TASOULIS†, V. P. PLAGIANAKOS‡,
and M. N. VRAHATIS§
Computational Intelligence Laboratory,
Department of Mathematics, University of Patras,
University of Patras Artificial Intelligence Research Center (UPAIRC),
GR-26110 Patras, Greece
**npav@math.upatras.gr*
†dtas@math.upatras.gr
‡vpp@math.upatras.gr
§vrahatis@math.upatras.gr

Received February 16, 2005; Revised April 11, 2005

In this paper, the combination of unsupervised clustering algorithms with feedforward neural networks in exchange rate time series forecasting is studied. Unsupervised clustering algorithms have the desirable property of deciding on the number of partitions required to accurately segment the input space during the clustering process, thus relieving the user from making this *ad hoc* choice. Combining this input space partitioning methodology with feedforward neural networks acting as local predictors for each identified cluster helps alleviate the problem of nonstationarity frequently encountered in real-life applications. An improvement in the one-step-ahead forecasting accuracy was achieved compared to a global feedforward neural network model for the time series of the exchange rate of the German Mark to the US Dollar.

Keywords: Time series modeling and prediction; unsupervised clustering; neural networks.

1. Introduction

System identification and time series prediction are embodiments of the old problem of function approximation [Principe *et al.*, 1998]. A discrete time series is a set of observations of a given variable $z(t)$ ordered according to the parameter time, and denoted as $z_1, z_2, \ldots, z_N$, where N is the size of the time series.

Conventional time series models rely on global approximation, employing techniques such as linear regression, polynomial fitting and artificial neural networks. Global models are well suited to problems with stationary dynamics. In the analysis of real-world systems, however, two of the key problems are nonstationarity (often in the form of switching between regimes) and overfitting (which is particularly serious for noisy processes) [Weigend *et al.*, 1995]. Nonstationarity implies that the statistical properties of the data generator vary through time. This leads to gradual changes in the dependency between the input and output variables.

Noise, on the other hand, refers to the unavailability of complete information from the past behavior of the time series to fully capture the dependency between the future and the past. Noise can be the source of overfitting, which implies that the performance of the forecasting model will be poor when applied to new data [Cao, 2003; Milidiu & Renteria, 1999]. Although global approximation methods can be applied to model and forecast time series having the aforementioned characteristics, it is reasonable to expect that the forecasting accuracy can be improved if regions of the input space exhibiting similar dynamics are identified and subsequently a local model is constructed for each of them. A number of researchers have proposed alternative methodologies to perform this task effectively [Cao, 2003; Milidiu & Renteria, 1999; Pavlidis *et al.*, 2003; Pavlidis *et al.*, 2005; Principe *et al.*, 1998; Sfetsos & Siriopoulos, 2004; Weigend *et al.*, 1995]. In principle, these methodologies are formed by the combination of two distinct approaches; an algorithm for the partitioning of the input space and a function approximation model. Evidently the partitioning of the input space is critical for the successful application of these methodologies.

In this paper we investigate the improvement in one-step-ahead forecasting accuracy that can be attained if the partitioning of the input space is performed through unsupervised clustering algorithms, while the function approximation model is a feedforward neural network. Clustering can be defined as the process of "grouping a collection of objects into subsets or clusters, such that those within one cluster are more closely related to one another than objects assigned to different clusters" [Hastie *et al.*, 2001]. Unsupervised clustering algorithms automatically approximate the number of clusters in the dataset during their execution. This property is important in the context of partitioning the input space for the purposes of time series forecasting, since the number of partitions corresponding to the different regimes is typically unknown *a priori*.

As a benchmark we consider the time series of the spot exchange rate of the German Mark against the US Dollar. Foreign exchange rates are among the most important economic indices in international monetary markets. Currently, foreign exchange markets are the most active of all financial markets with average daily trading volumes in traditional (nonelectronic broker) estimated at \$1.2 trillion [Bank of International Settlements, 2001]. Although the precise scale of speculative trading on spot markets is unknown it is estimated that only around 15% of the trading is driven by non-dealer/financial institution trading. Approximately, 90% of all foreign currency transactions involve the US Dollar [Bank of International Settlements, 2001]. Foreign exchange rates are affected by many highly correlated economic, political and psychological factors, the interaction of which is very complex. Thus, forecasting foreign exchange rates poses many theoretical and experimental challenges [Yao & Tan, 2000].

The remaining paper is organized as follows: in the next section we present the clustering and neural network algorithms employed in this study. In Sec. 3 experimental results regarding the spot exchange rate of the German Mark against the US Dollar are presented. The paper ends with a short discussion of the results and concluding remarks.

2. Methods

In this section we briefly describe the application of unsupervised clustering and neural networks in the context of time series modeling and prediction. Furthermore, three unsupervised clustering algorithms, as well as three neural network training algorithms are outlined.

2.1. *Unsupervised clustering algorithms*

A critical issue in the process of partitioning the input space for the purpose of time series modeling and forecasting is to obtain an appropriate estimation of the number of subsets. Over- or underestimation of this quantity can cause the appearance of clusters with little or no physical meaning, and/or clusters containing patterns from regions with different dynamics, and/or clusters with very few patterns that are insufficient for the training of a feedforward neural network.

This is a fundamental and unresolved problem in cluster analysis, independent of the clustering technique applied. For instance, well-known and widely used iterative techniques, such as Self-Organizing Maps (SOMs) [Kohonen, 1997], the k-means algorithm [Hartigan & Wong, 1979], as well as, the Fuzzy c-means algorithm [Bezdek, 1981], require from the user to specify the number of clusters present in the dataset prior to the execution of the algorithm.

On the other hand, algorithms that have the ability to approximate the number of clusters present in a dataset belong to the category of *unsupervised* clustering algorithms. In this study we consider only unsupervised clustering algorithms. In particular, we employ the Growing Neural Gas [Fritzke, 1995], the DBSCAN [Ester *et al.*, 1996], and the unsupervised k-windows [Tasoulis & Vrahatis, 2004; Vrahatis *et al.*, 2002] clustering algorithms. Next, the aforementioned unsupervised algorithms are briefly presented.

2.1.1. *Growing Neural Gas clustering algorithm*

The Growing Neural Gas (GNG) clustering algorithm [Fritzke, 1995] is an incremental neural network. It can be described as a graph consisting of k nodes, each of which has an associated weight vector, defining the node's position in the data space and a set of edges between the node and its neighbors. During the clustering procedure, new nodes are added to the network until a maximal number of nodes is reached. GNG starts with two nodes, randomly positioned in the data space, connected by an edge. Adaptation of weights, i.e. the nodes' positions, is performed iteratively. For each data object the closest node (winner), s_1, and the closest neighbor of the winner node, s_2, are identified. These two nodes are connected by an edge. An age variable is associated with each edge. When the edge between s_1 and s_2 is created its age is set to zero. At each learning step the age variable of all edges emanating from the winner node are increased by one. By tracing the changes of the age variable it is possible to detect inactive nodes. Edges exceeding a maximal age, R, and any nodes having no emanating edges are removed. The neighborhood of the winner is limited to its topological neighbors. The winner and its topological neighbors are moved in the data space toward the presented object by a constant fraction of the distance, defined separately for the winner and its topological neighbors. There is no neighborhood function, or ranking concept and thus, all topological neighbors are updated in the same manner.

2.1.2. *The DBSCAN clustering algorithm*

The DBSCAN clustering algorithm [Sander *et al.*, 1998] relies on a density-based notion of clusters and is designed to discover clusters of arbitrary shape and to distinguish noise. More specifically, the algorithm relies on the idea that for each point in a cluster at least a minimum number of objects, $MinPts$, should be contained in a neighborhood of a given radius, *Eps*, around it. Thus, by iteratively scanning all the points in the dataset DBSCAN forms clusters of points that are connected through chains of *Eps*-neighborhoods of at least $MinPts$ points each.

2.1.3. *Unsupervised k-windows*

The unsupervised k-windows clustering algorithm [Tasoulis & Vrahatis, 2004; Vrahatis *et al.*, 2002] uses a windowing technique to discover the clusters present in a dataset. More specifically, if we suppose that the dataset lies in d dimensions, it initializes a number of d-dimensional windows over the dataset. At a next step it iteratively moves and enlarges these windows to enclose all the patterns that belong to one cluster in a window. The movement and enlargement procedures are guided by the points that lie within a window at each iteration. As soon as the movement and enlargement procedures do not alter significantly the number of points within a window they terminate. The final set of windows defines the clustering result of the algorithm. The unsupervised k-windows algorithm (UKW) applies the k-windows algorithm using a "sufficiently" large number of initial windows. The windowing technique of the k-windows algorithm allows for a large number of initial windows to be examined without any significant overhead in time complexity. At a final step the windows that contain a high percentage of common points from the dataset are considered to belong to the same cluster. Thus the number of clusters can be determined [Alevizos *et al.*, 2002; Alevizos *et al.*, 2004; Tasoulis & Vrahatis, 2004].

2.2. ***Feedforward Neural Networks***

Artificial Neural Networks (ANNs) have been widely employed in numerous fields and have shown their strengths in solving real-world problems. ANNs are parallel computational models comprised of interconnected adaptive processing units (neurons), characterized by an inherent propensity for storing experiential knowledge. They resemble the human brain in two fundamental respects; firstly, knowledge is acquired by the network from its environment through a learning process, and secondly, interneuron connection strengths (known as weights) are employed to store the acquired knowledge [Haykin, 1999].

Numerous neural network models have been proposed, but multilayered Feedforward Neural Networks (FNNs) are the most common. In FNNs neurons are arranged in layers and there are connections between neurons in one layer to the neurons of the following layer. The learning rule typically used for FNNs is supervised training. Two critical parameters for the successful application of FNNs are the appropriate selection of the network architecture and the training algorithm. For the general problem of function approximation, the *universal* *approximation theorem*, proved in [White, 1990] states that:

Theorem 2.1. *Standard Feedforward Networks with only a single hidden layer can approximate any continuous function uniformly on any compact set and any measurable function to any desired degree of accuracy.*

An immediate implication of the above theorem is that any lack of success in applications must arise from inadequate learning and/or an insufficient number of hidden units and/or the lack of a deterministic relationship between the input patterns and the desired response (target).

In the context of time series modeling the inputs to the FNN typically consist of a number of delayed observations, while the target is the next value of the series. The *universal myopic mapping theorem* [Sandberg & Xu, 1997a, 1997b] states that any shift-invariant map can be approximated arbitrarily well by a structure consisting of a bank of linear filters feeding an FNN. An implication of this theorem is that, in practice, FNNs alone can be insufficient to capture the dynamics of a nonstationary system [Haykin, 1999]. This is also verified by the results presented in this paper.

The selection of the optimal network architecture for a specific task remains up to date an open problem. An upper bound on the architecture of an FNN designed to approximate a continuous function defined on the unit cube in $\mathbb{R}^n$ is given by the following Theorem [Pinkus, 1999]:

Theorem 2.2. *On the unit cube in $\mathbb{R}^n$ any continuous function can be uniformly approximated, to within any error by using a two hidden layer network having $2n+1$ units in the first layer and $4n+3$ units in the second layer.*

2.3. *Supervised training of neural networks*

The supervised training process is an incremental adaptation of the weights that propagate information between the neurons. Learning in FNNs is achieved by minimizing the network error using a *batch*, also called *offline*, or a *stochastic*, also called *online*, training algorithm.

Batch training is considered as the classical machine learning approach. In time series applications, a set of patterns is used for modeling the system, before the network is actually used for prediction. In this case, the goal is to find a minimizer $w^* = (w_1^*, w_2^*, \ldots, w_n^*) \in \mathbb{R}^n$, such that:

$$w^* = \min_{w \in \mathbb{R}^n} E(w),$$

where E is the batch error measure of the FNN, whose lth layer $(l = 1, \ldots, M)$ contains N_l neurons:

$$E = \frac{1}{2} \sum_{p=1}^{P} \sum_{j=1}^{N_M} (y_{j,p}^M - t_{j,p})^2 = \sum_{p=1}^{P} E_p. \qquad (1)$$

In the above relation, the error function is based on the squared difference between the actual output value at the jth output layer neuron for pattern p, $y_{j,p}^M$, and the target output value, $t_{j,p}$. E_p is the error of the pth pattern and p is the index over the input–output pairs. To predict the next value of the time series, there is only one output neuron ($N_M = 1$). On the other hand, when the problem is formulated as a classification task the value of N_M can vary according to the number of classes.

Supervised training is a difficult task since, in general, the dimension of the weight space is very high and the function E generates a complicated surface, characterized by multiple local minima and broad flat regions adjoined to narrow steep ones.

In online training, the FNN weights are updated after the presentation of each training pattern. Online training may be the appropriate choice for learning a task either because of the very large (or even redundant) training set, or because of the slowly time-varying nature of the task. Although batch training seems faster for small-size training sets and networks, online training is probably more efficient for large training sets and FNNs. It often helps to avoid local minima and provides a more natural approach for learning nonstationary tasks, such as time series modeling and prediction. Online methods seem to be more robust than batch methods as errors, omissions, or redundant data in the training set can be corrected, or ejected during the training phase.

In this paper we have employed and compared four algorithms for batch training and one online training algorithm. The batch training algorithms were the well-known Resilient Propagation (RPROP) [Riedmiller & Braun, 1993], a Scaled Conjugate Gradient (SCG) [Møller, 1993] and two population based algorithms, namely the Differential Evolution algorithm (DE) [Storn & Price, 1997] and the Particle Swarm Optimization (PSO) [Eberhart *et al.*, 1996]. We also implemented the recently proposed Adaptive Online BackPropagation training algorithm (AOBP) [Magoulas *et al.*, 2001; Plagianakos *et al.*, 2000]. Next, we briefly describe the AOBP, the DE, as well as, the PSO algorithms.

2.3.1. *The Online Neural Network training algorithm*

Despite the abundance of methods for learning from examples, there are only a few that can be used effectively for online learning. For example, the classic batch training algorithms cannot straightforwardly handle nonstationary data. Even when some of them are used in online training the problem of "catastrophic interference" appears, in which training on new examples interferes excessively with previously learned examples, leading to saturation and slow convergence [Sutton & Whitehead, 1993].

Methods suited to online learning are those that can handle time-varying data, while at the same time, require relatively little additional memory and computation in order to process an additional example. The AOBP method proposed in [Magoulas *et al.*, 2001; Plagianakos *et al.*, 2000] belongs to this class of methods.

The key features of this method are the low storage requirements and the inexpensive computations. At each iteration, the d-dimensional weight vector is evaluated using the following update formula:

$$w^{g+1} = w^g - \eta^g \nabla E(w^g).$$

To calculate the learning rate for the next iteration, η^{g+1}, AOBP uses information from the current and the previous iteration. In detail, the new learning rate is calculated through the following relation:

$$\eta^{g+1} = \eta^g + K \langle \nabla E(w^{g-1}), \nabla E(w^g) \rangle,$$

where η is the learning rate, K is the meta-learning rate constant (typically $K = 0.5$), and $\langle \cdot, \cdot \rangle$ stands for the usual inner product in $\mathbb{R}^d$. This approach stabilizes the learning rate adaptation process, and previous experiments [Magoulas *et al.*, 2001; Plagianakos *et al.*, 2000] have shown that it allows the method to exhibit good generalization and high convergence rate.

2.3.2. *Differential Evolution training algorithm*

DE [Storn & Price, 1997] is a novel minimization method designed to handle nondifferentiable, nonlinear and multimodal objective functions, by exploiting a *population* of *NP* potential solutions, that is d-dimensional vectors, to probe the search space. At each iteration of the algorithm, called *generation*, g, three steps, *mutation*, *recombination* and *selection*, are performed to obtain more accurate approximations [Plagianakos & Vrahatis, 2002]. Initially, all weight vectors are initialized by using a random number generator. At the mutation step, for each $i = 1, \ldots, NP$ a new mutant weight vector v^i_{g+1} is generated by combining weight vectors, randomly chosen from the population, and exploiting the following variation operator:

$$v^i_{g+1} = \omega^i_g + \mu(\omega^{\text{best}}_g - \omega^i_g + \omega^{r1}_g - \omega^{r2}_g), \qquad (2)$$

where ω^{r1}_g and ω^{r2}_g are randomly selected vectors, different from ω^i_g, and ω^{best}_g is the member of the current generation that yielded the lowest error function value. Finally, the positive mutation constant μ, controls the magnification of the difference between two weight vectors (typically $\mu = 0.8$).

The resulting mutant vectors are mixed with a predetermined weight vector, called *target* vector. This operation is called *recombination*, and it gives rise to the *trial* vector. At the recombination step, for each component $j = 1, 2, \ldots, d$ of the mutant weight vector a random number $r \in [0, 1]$ is generated. If r is smaller than the predefined recombination constant p (typically $p = 0.9$), the jth component of the mutant vector v^i_{g+1} becomes the jth component of the trial vector. Otherwise, the jth component of the target vector, ω^i_g, is selected as the jth component of the trial vector. Finally, at the selection step, the trial weight vector obtained after the recombination step is accepted for the next generation, if and only if, it yields a reduction of the value of the error function relative to the previous weight vector; otherwise, the previous weight vector is retained.

2.3.3. *Particle Swarm Optimization training algorithm*

PSO is a swarm-intelligence optimization algorithm capable of minimizing nondifferentiable, nonlinear and multimodal objective functions. Each member of the swarm, called *particle*, moves with an adaptable velocity within the search space, and retains in its memory the best position it ever encountered. At each iteration, the best position ever attained by the swarm is communicated among the particles [Eberhart *et al.*, 1996].

Assume a d-dimensional search space, $\mathcal{S} \subset \mathbb{R}^d$, and a swarm of NP particles. Both the position and the velocity of the ith particle are d-dimensional vectors, $x_i \in \mathcal{S}$ and $v_i \in \mathbb{R}^d$, respectively. The best previous position ever encountered by the ith particle is denoted by p_i, while the best previous position attained by the swarm is denoted by p_g. The velocity [Clerc & Kennedy, 2002] of the ith particle at the $(g+1)$-th iteration is obtained through Eq. (3). The new position of this particle is determined by simply adding the velocity vector to the previous position vector, Eq. (4).

$$v_i^{(g+1)} = \chi(v_i^{(g)} + c_1 r_1 (p_i^{(g)} - x_i^{(g)}) + c_2 r_2 (p_g^{(g)} - x_i^{(g)})), \tag{3}$$

$$x_i^{(g+1)} = x_i^{(g)} + v_i^{(g+1)}, \tag{4}$$

where $i = 1, \ldots, NP$; c_1 and c_2 are positive constants (typically $c_1 = c_2 = 2.05$); r_1, r_2 are random numbers uniformly distributed in $[0, 1]$; and χ is the constriction factor (typically $\chi = 0.729$). In general, PSO has proved to be very efficient and effective in tackling various difficult problems [Parsopoulos & Vrahatis, 2002].

3. Presentation of Experimental Results

The time series considered was that of the daily spot prices of the exchange rate of the German Mark relative to the US Dollar [Keogh & Folias, 2002]. The time period considered extends from 10/9/1986 to 8/9/1996, covering approximately ten years. The total number of observations was 2567. The first 2317 were used to evaluate the parameters of the predictive models, while the remaining 250, covering approximately the final year of the dataset, were used to evaluate their performance.

The first step in the analysis and prediction of time series originating from real-world systems is the choice of an appropriate time delay, T, and the determination of the embedding dimension, D. To select T an established approach is to use the value that yields the first minimum of the mutual information function [Fraser, 1989]. For the considered time series no minimum occurs for $T = 1, \ldots, 20$, as illustrated in Fig. 1. In this case a time delay of one is typically selected. To determine the minimum embedding dimension for state space reconstruction we applied the method of "False Nearest Neighbors" [Hegger *et al.*, 1999; Kennel *et al.*, 1992]. As illustrated in Fig. 1 the proportion of false nearest neighbors as a function of D drops sharply to the value of 0.006 for D equal to five, which is the embedding dimension that we selected, and it becomes zero for dimensions higher than seven. With this embedding dimension the number of patterns used to evaluate the parameters of the predictive models was 2312 while the performance of the models was evaluated on the last 250 patterns.

Having selected an embedding dimension, we tested numerous FNNs with different architectures and training algorithms, but no FNN was capable of producing a satisfactory test set prediction accuracy. In fact, the forecasts resembled a time-lagged version of the original series. Next, the three unsupervised clustering algorithms, namely GNG,

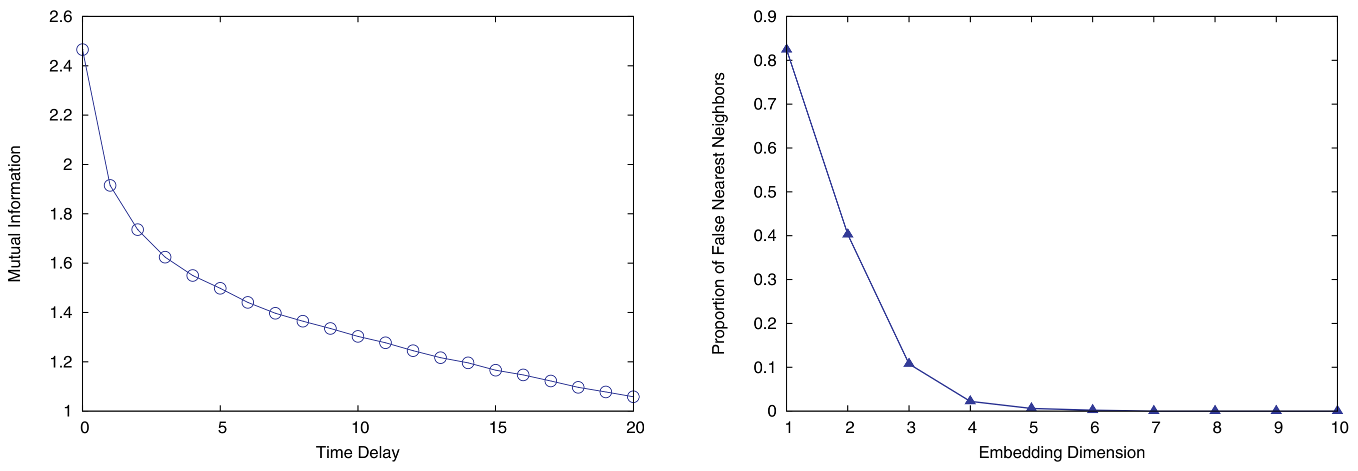

Fig. 1. Mutual information as a function of T (left) and proportion of "false nearest neighbors" as a function of D (right).

DBSCAN and UKW, were applied on the patterns of the training set to obtain a partition of the input space. Note that the value to be predicted (target value) by the FNNs acting as local approximators, was also included in the patterns comprising the dataset supplied to the clustering algorithms. Our experience suggests that this approach slightly improves the overall forecasting performance. Once the clusters present in the training set are identified, each pattern from the test set is assigned to one of the clusters. Since the target value for patterns in the test set is unknown the assignment is performed by not taking into consideration the additional dimension that corresponds to the target component. A test set pattern is assigned to the cluster to which the nearest (in terms of Euclidean distance) node, pattern, window center, belongs for the GNG, DBSCAN, and UKW algorithms, respectively.

We evaluate the accuracy of the FNNs by the percentage of correct *sign* prediction [de Bodt *et al.*, 2001; Giles *et al.*, 2001; Walczak, 2001]. This measure captures the percentage of forecasts in the test set for which the following inequality is satisfied:

$$(\widehat{x_{t+d}} - x_{t+d-1}) \cdot (x_{t+d} - x_{t+d-1}) > 0, \tag{5}$$

where, $\widehat{x_{t+d}}$ represents the prediction generated by the FNN, x_{t+d} refers to the true value of the exchange rate at period $t + d$ and, finally, x_{t+d-1} stands for the value of the exchange rate at the current period, $t + d - 1$. Correct sign prediction in effect captures the percentage of profitable trades enabled by the forecasting system. To successfully train FNNs capable of forecasting the direction of change of the time series, a modified, nondifferentiable, error function was implemented:

$$E_k = \begin{cases} 0.5 \cdot |x_{t+d} - \widehat{x_{t+d}}|, & \text{if } (\widehat{x_{t+d}} - x_{t+d-1}) \\ & \quad \cdot (x_{t+d} - x_{t+d-1}) \\ & \quad > 0 \\ |x_{t+d} - \widehat{x_{t+d}}|, & \text{otherwise.} \end{cases} \tag{6}$$

Since RPROP, SCG and AOBP are gradient based algorithms, this function is employed only when the FNNs are trained through the DE and PSO algorithms.

Numerical experiments were performed using a Clustering and a Neural Network C++ Interface, built under the Fedora Core Linux 3.0 operating system using the GNU compiler collection (gcc) version 3.4.2. The results obtained are reported in Tables 1–3 and the accompanying figures. Each table reports the total number of clusters identified in the training set. Furthermore, it reports the number of clusters to which test set patterns were assigned. For each such cluster the number of patterns from the training set and the test set assigned to this cluster are also reported. Notice that irrespective of

Table 1. UKW: Results.

	Patterns in the	
	Train Set	Test Set
Cluster 1	84	33
Cluster 2	82	57
Cluster 3	65	3
Cluster 4	239	67
Cluster 5	210	90

Total number of clusters: 13.
Clusters used in test set: 5.

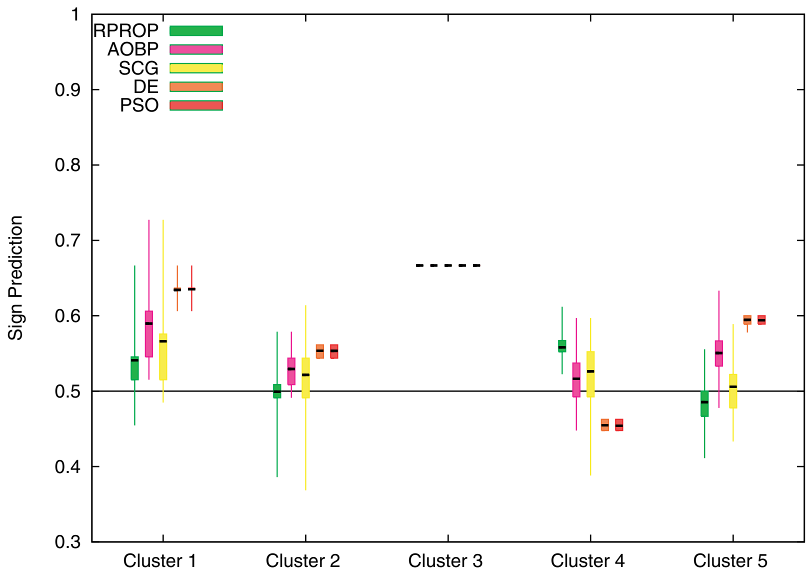

Fig. 2. Proportion of correct sign prediction based on the clustering of the input space using the UKW algorithm.

Table 2. DBSCAN: Results.

	Patterns in the	
	Train Set	Test Set
Outliers	1353	123
Cluster 1	95	59
Cluster 2	81	53
Cluster 3	4	4
Cluster 4	11	11

Total number of clusters: 12.
Clusters used in test set: 5.

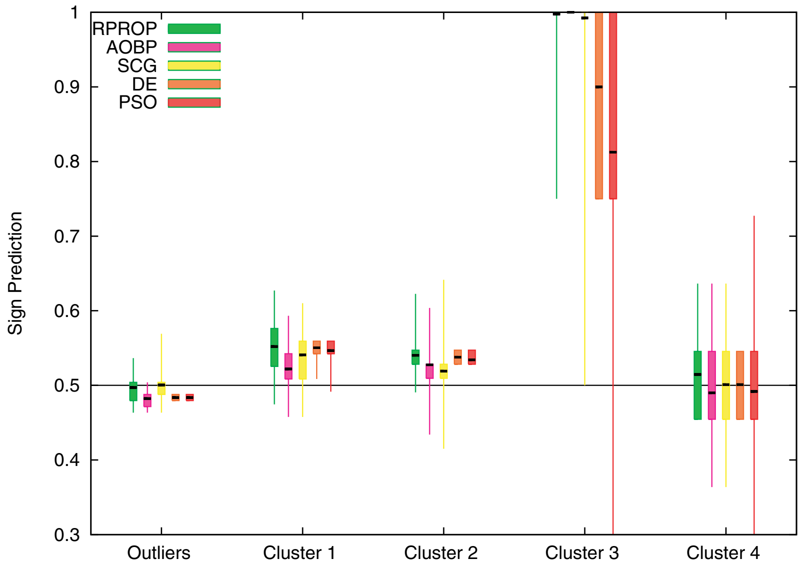

Fig. 3. Proportion of correct sign prediction based on the clustering of the input space using the DBSCAN algorithm.

Table 3. GNG: Results.

	Patterns in the	
	Train Set	Test Set
Cluster 1	90	57
Cluster 2	61	29
Cluster 3	94	6
Cluster 4	496	158

Total number of clusters: 9.
Clusters used in test set: 4.

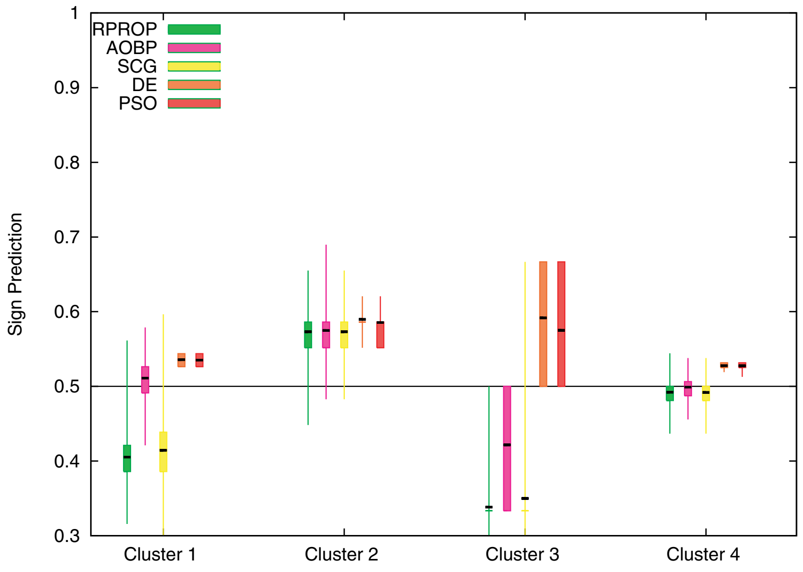

Fig. 4. Proportion of correct sign prediction based on the clustering of the input space using the GNG algorithm.

the clustering algorithm, a relatively small proportion of the patterns contained in the training set was actually used to generate the predictions, since training only the FNNs corresponding to the particular clusters is necessary. The accompanying figures provide candlestick plots. Each candlestick depicts for a cluster and a training algorithm the forecasting accuracy with respect sign prediction, obtained over 100 experiments. A filled box is plotted between the first and third quartile of the data. The lines extending from each end of the box (whiskers) show the range of the data. The black line inside the box stands for the mean value of the measurements. An immediate observation from the inspection of the figures is that there are significant differences in the predictability of the different clusters, irrespective of the clustering algorithm. Moreover, within the same cluster, different training algorithms produced FNNs yielding different predictive accuracy.

For clusters 1, 3, 5 identified by the UKW algorithm and having the corresponding FNNs trained by the DE and PSO algorithms, a mean predictive accuracy exceeding 60% was achieved. These three clusters together comprise more than 50% of the test set. However, the predictability of cluster 4 (26.8% of the test set) is rather low. As previously mentioned, DBSCAN has the ability to identify outliers in a dataset. In this case close to 50% of the patterns of the test set were characterized as outliers. The FNN trained on these patterns produced a poor performance. On the other hand, the mean predictability for clusters 1 and 2 was around 55%. Note that cluster 3 (to which four test patterns were assigned) exhibited extremely high predictability. The GNG algorithm distinguished cluster 2 for which the corresponding FNN produced a mean accuracy close to 60% irrespective of the training algorithm used. For cluster 4, PSO and DE exhibited good performance, but the other three algorithms yielded the worst performance witnessed in this study.

4. Discussion and Concluding Remarks

In this study, we report results from the application of unsupervised clustering algorithms, combined with feedforward neural networks in exchange rate time series forecasting. The desirable property of unsupervised clustering algorithms is that they can automatically approximate the number of partitions (clusters), thus relieving the user from making this critical, problem-specific, choice.

Combining this input space partitioning methodology with feedforward neural networks acting as local predictors for each identified cluster helps alleviate the problem of nonstationarity frequently encountered in real-life applications. Feedforward neural networks are selected as local approximation models due to their ability to cope with noise. Through this approach an improvement, compared to global feedforward neural networks, in the one-step-ahead prediction of the direction of change of the daily spot exchange rate of the German Mark to the US Dollar was achieved.

Among the unsupervised clustering algorithms considered, UKW's performance is more robust. Both the DBSCAN and the GNG algorithms however, were capable of identifying meaningful clusters that yielded increased predictability in the test set. From the training algorithms considered, FNNs trained using the AOBP training algorithm exhibited the highest maximum performance. The performance of the population based algorithms, DE and PSO, exhibited wide variations.

Future work will include the synthesis of the results of the different clustering algorithms to improve the forecasting performance in larger regions of the input space, and also the examination of other real-life time series.

Acknowledgments

The authors thank the European Social Fund (ESF), Operational Program for Educational and Vocational Training II (EPEAEK II) and particularly the Program PYTHAGORAS, for funding this work. This work was supported in part by the "Karatheodoris" research grant awarded by the Research Committee of the University of Patras.

References

Alevizos, P., Boutsinas, B., Tasoulis, D. K. & Vrahatis, M. N. [2002] "Improving the orthogonal range search *k*-windows clustering algorithm," *Proc. 14th IEEE Int. Conf. Tools with Artificial Intelligence*, Washington, D.C., pp. 239–245.

Alevizos, P., Tasoulis, D. K. & Vrahatis, M. N. [2004] *Parallelizing the Unsupervised k-Windows Clustering Algorithm*, Lecture Notes in Computer Science, Vol. 3019, pp. 225–232.

Bank of International Settlements [2001] "Central bank survey of foreign exchange and derivative market Activity in April 2001," Bank of International Settlements.

Bezdek, J. C. [1981] *Pattern Recognition with Fuzzy Objective Function Algorithms* (Kluwer Academic Publishers).

Cao, L. [2003] "Support vector machines experts for time series forecasting," *Neurocomputing* **51**, 321–329.

Clerc, M. & Kennedy, J. [2002] "The particle swarm-explosion, stability, and convergence in a multidimensional complex space," *IEEE Trans. Evol. Comput.* **6**, 58–73.

de Bodt, E., Rynkiewicz, J. & Cottrell, M. [2001] "Some known facts about financial data," *European Symp. Artificial Neural Networks* (*ESANN'2001*), pp. 223–236.

Eberhart, R. C., Simpson, P. & Dobbins, R. [1996] *Computational Intelligence PC Tools* (Academic Press).

Ester, M., Kriegel, H. P., Sander, J. & Xu, X. [1996] "A density-based algorithm for discovering clusters in large spatial databases with noise," *Proc. 2nd Int. Conf. Knowledge Discovery and Data Mining*, pp. 226–231.

Fraser, A. M. [1989] "Information and entropy in strange attractors," *IEEE Trans. Inform. Th.* **35**, 245–262.

Fritzke, B. [1995] "A growing neural gas network learns topologies," *Advances in Neural Information Processing Systems*, eds. Tesauro, G., Touretzky, D. S. & Leen, T. K. (MIT Press, Cambridge MA), pp. 625–632.

Giles, L. C., Lawrence, S. & Tsoi, A. H. [2001] "Noisy time series prediction using a recurrent neural network and grammatical inference," *Mach. Learn.* **44**, 161–183.

Hartigan, J. A. & Wong, M. A. [1979] "A *k*-means clustering algorithm," *Appl. Stat.* **28**, 100–108.

Hastie, T., Tibshirani, R. & Friedman, J. [2001] *The Elements of Statistical Learning* (Springer-Verlag).

Haykin, S. [1999] *Neural Networks*: *A Comprehensive Foundation* (Macmillan College Publishing Company, NY).

Hegger, R., Kantz, H. & Schreiber, T. [1999] "Practical implementation of nonlinear time series methods: The TISEAN package," *Chaos* **9**, 413–435.

Kennel, M. B., Brown, R. & Abarbanel, H. D. [1992] "Determining embedding dimension for phase–space reconstruction using a geometrical construction," *Phys. Rev. A* **45**, 3403–3411.

Keogh, E. & Folias, T. [2002] *The UCR Time Series Data Mining Archive.*

Kohonen, T. [1997] *Self-Organized Maps* (Berlin, Springer).

Magoulas, G. D., Plagianakos, V. P. & Vrahatis, M. N. [2001] "Adaptive stepsize algorithms for online training of neural networks," *Nonlin. Anal. T.M.A.* **47**, 3425–3430.

Møller, M. [1993] "A scaled conjugate gradient algorithm for fast supervised learning," *Neural Networks* **6**, 525–533.

Parsopoulos, K. E. & Vrahatis, M. N. [2002] "Recent approaches to global optimization problems through particle swarm optimization," *Natural Comput.* **1**, 235–306.

Pavlidis, N. G., Tasoulis, D. K. & Vrahatis, M. N. [2003] "Financial forecasting through unsupervised Clustering and evolutionary trained neural networks," *Proc. Congress on Evolutionary Computation* (*CEC 2003*), pp. 2314–2321.

Pavlidis, N. G., Tasoulis, D. K. & Vrahatis, M. N. [2005] "Time series forecasting methodology for multiple–step–ahead prediction," *The IASTED Int. Conf. Computational Intelligence* (*CI 2005*), pp. 456–461.

Pinkus, A. [1999] "Approximation theory of the mlp model in neural networks," *Acta Numerica*, 143–195.

Plagianakos, V. P., Magoulas, G. D. & Vrahatis, M. N. [2000] "Global learning rate adaptation in online neural network training," *Proc. Second Int. ICSC Symp. Neural Computation* (*NC'2000*).

Plagianakos, V. P. & Vrahatis, M. N. [2002] "Parallel evolutionary training algorithms for 'hardware–friendly' neural networks," *Natural Comput.* **1**, 307–322.

Principe, J. C., Wang, L. & Motter, M. A. [1998] "Local dynamic modeling with self–organizing maps and applications to nonlinear system identification and control," *Proc. IEEE* **86**, 2240–2258.

Milidiu, R. L., Machado, R. J. & Renteria, R. P. [1999] "Time-series forecasting through wavelets transformation and a mixture of expert models," *Neurocomputing* **28**, 145–156.

Riedmiller, M. & Braun, H. [1993] "A direct adaptive method for faster backpropagation learning: The RPROP algorithm," *Proc. IEEE Int. Conf. Neural Networks* (San Francisco, CA), pp. 586–591.

Sandberg, I. W. & Xu, L. [1997a] "Uniform approximation and gamma networks," *Neural Networks* **10**, 781–784.

Sandberg, I. W. & Xu, L. [1997b] "Uniform approximation of multidimensional myopic maps," *IEEE Trans. Circuits Syst. I*: *Fund. Th. Appl.* **44**, 477–485.

Sander, J., Ester, M., Kriegel, H.-P. & Xu, X. [1998] "Density-based clustering in spatial databases: The algorithm GDBSCAN and its applications," *Data Min. Knowl. Discov.* **2**, 169–194.

Sfetsos, A. & Siriopoulos, C. [2004] "Time series forecasting with a hybrid clustering scheme and pattern recognition," *IEEE Trans. Syst.*, *Man Cybern. Part A*: *Syst. Humans* **34**, 399–405.

Storn, R. & Price, K. [1997] "Differential evolution — A simple and efficient adaptive scheme for global optimization over continuous spaces," *J. Glob. Optim.* **11**, 341–359.

Sutton, R. S. & Whitehead, S. D. [1993] "Online learning with random representations," *Proc. Tenth Int. Conf. Machine Learning* (Morgan Kaufmann), pp. 314–321.

Tasoulis, D. K. & Vrahatis, M. N. [2004] "Unsupervised distributed clustering," *IASTED Int. Conf. Parallel and Distributed Computing and Networks* (Innsbruck, Austria), pp. 347–351.

Vrahatis, M. N., Boutsinas, B., Alevizos, P. & Pavlides, G. [2002] "The new *k*-windows algorithm for improving the *k*-means clustering algorithm," *J. Compl.* **18**, 375–391.

Walczak, S. [2001] "An empirical analysis of data requirements for financial forecasting with neural networks," *J. Manag. Inform. Syst.* **17**, 203–222.

Weigend, A. S., Mangeas, M. & Srivastava, A. N. [1995] "Nonlinear gated experts for time series: Discovering regimes and avoiding overfitting," *Int. J. Neural Syst.* **6**, 373–399.

White, H. [1990] "Connectionist nonparametric regression: Multilayer feedforward networks can learn arbitrary mappings," *Neural Networks* **3**, 535–549.

Yao, J. & Tan, C. L. [2000] "A case study on using neural networks to perform technical forecasting of forex," *Neurocomput.* **34**, 79–98.