Time Series Forecasting Methodology for Multiple–Step–Ahead Prediction

N. G. Pavlidis, D. K. Tasoulis, M. N. Vrahatis
Department of Mathematics,
University of Patras Artificial Intelligence Research Center (UPAIRC),
University of Patras, GR–26110 Patras, Greece.
{npav,dtas,vrahatis}@math.upatras.gr

ABSTRACT
This paper presents a time series forecasting methodology and applies it to generate multiple–step–ahead predictions for the direction of change of the daily exchange rate of the Japanese Yen against the US Dollar. The proposed methodology draws from the disciplines of chaotic time series analysis, clustering, and artificial neural networks. In brief, clustering is applied to identify neighborhoods in the reconstructed state space of the system; and subsequently neural networks are trained to model the dynamics of each neighborhood separately. The results obtained through this approach are promising.

KEY WORDS
Computational Intelligence, Forecasting, Clustering, Neural Networks

1 Introduction

System identification and time–series prediction are embodiments of the old problem of function approximation. The classic approach is to build an explanatory model from first principles and measure initial data [3]. Unfortunately, this approach is not always feasible. Here we assume knowledge of the scalar time series only. The most common approach consists of two steps:

- identify a model capable of performing one–step–ahead predictions of the time series, and
- generate a long time–series by iterated prediction.

Principe et al. [14] report that in many cases, this approach fails. The reason is that the selected model has not learned the chaotic attractor despite the fact that it is capable of performing accurate one–step–ahead prediction. In the work of Principe et al. [15] a self–organizing map is used to partition the input space. This is a step toward a model that makes accurate short–term predictions and learns the outlines of the chaotic attractor.

In this paper we propose a time series forecasting methodology that draws from the disciplines of chaotic time series analysis, clustering, and artificial neural networks, and apply it to perform multiple–step–ahead predictions of the time series of the daily exchange rate of the Japanese Yen against the US Dollar. The proposed methodology is related to the notion of local approximation [3] and has been previously applied to generate one–step–ahead predictions for two financial time–series [12].

The remaining paper is organized as follows: Section 2 describes analytically the proposed forecasting methodology; Section 3 is devoted to implementation details and the numerical results obtained. Conclusions and ideas for future research are provided in Section 4.

2 Proposed Methodology

Instead of constructing a global model for a chaotic time series, Farmer and Sidorowich [3] proposed to construct models for neighborhoods of the state space, an approach known as *local approximation*. In brief, to predict $x(t+T)$ primarily, the m nearest neighbors of the state vector $\mathbf{x}(\mathbf{t})$, i.e. the m states $\mathbf{x}(\mathbf{t}')$ that minimize the distance $\|\mathbf{x}(\mathbf{t}) - \mathbf{x}(\mathbf{t}')\|$, are found. Then, a *local predictor* is constructed using $\mathbf{x}(\mathbf{t}')$ as points of the domain and $x(t'+T)$ as points of the range. Finally, the local predictor is used to forecast $x(t+T)$. The technique of local linear models is appealing for modeling complex time–series due to the weak assumptions required and its intrinsic simplicity. This approach is closely related to differential topology and it is more general than the global approach, in the sense that fewer statistical and geometric assumptions about the data are required. Computational intelligence methods have been used both as means of partitioning the input space, and as local predictors [4, 10, 15, 19].

Our approach is based on partitioning the input space through the unsupervised k–windows clustering algorithm [17]. This algorithm has the ability to endogenously determine the number of clusters present in the dataset. Once the clustering process is complete, a feed-forward neural network acts as the local predictor for each cluster. In brief, the proposed methodology consists of the following steps:

1. determine the minimum embedding dimension for phase–space reconstruction [7],
2. identify the clusters present in the training set,

3. for each cluster in the training set train a different feedforward neural network using for training patterns, patterns from that cluster solely.

4. To perform multiple–step–ahead prediction on the test set:

   (a) assign the input pattern to the appropriate cluster,

   (b) use the corresponding trained neural network to generate the prediction,

   (c) use the predicted value to formulate the next pattern.

2.1 Unsupervised k–windows Algorithm

For completeness purposes we briefly outline the workings of the unsupervised k–windows (UKW) clustering algorithm [17].

Intuitively, the k-windows algorithm tries to place a d-dimensional window containing all patterns that belong to a single cluster; for all clusters present in the dataset. At first, k points are selected (possibly in a random manner). The k initial d–ranges (windows), of size a, have as centers these points. Subsequently, the patterns that lie within each d-range are identified. Next, the mean of the patterns that lie within each d–range (i.e. the mean value of the d–dimensional points) is calculated. The new position of the d–range is such that its center coincides with the previously computed mean value. The last two steps are repeatedly executed as long as the increase in the number of patterns included in the d–range that results from this motion satisfies a stopping criterion. The stopping criterion is determined by a variability threshold θ_v that corresponds to the least change in the center of a d–range that is acceptable to recenter the d–range.

Once movement is terminated, the d–ranges are enlarged in order to capture as many patterns as possible from the cluster. Enlargement takes place at each dimension separately. The d–ranges are enlarged by θ_e/l percent at each dimension, where θ_e is user defined, and l stands for the number of previous successful enlargements. After the enlargement in one dimension is performed, the window is moved, as described above. Once movement terminates, the proportional increase in the number of patterns included in the window is calculated. If this proportion does not exceed the user–defined coverage threshold, θ_c, the enlargement and movement steps are rejected and the position and size of the d–range are reverted to their prior to enlargement values. Otherwise, the new size and position are accepted. If enlargement is accepted for dimension $d' \geqslant 2$, then for all dimensions d'', such that $d'' < d'$, the enlargement process is performed again assuming as initial position the current position of the window. This process terminates if enlargement in any dimension does not result in a proportional increase in the number of patterns included in the window beyond the threshold θ_c.

UKW generalizes the original algorithm. The key idea to automatically determine the number of clusters, is to apply 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. Once all the processes of movement and enlargement for all windows are terminates, all overlapping windows are considered for merging. The merge operation is guided by a merge threshold θ_m. Having identified two overlapping windows, the number of patterns that lie in their intersection is calculated. Next the proportion of this number to the total patterns included in each window is calculated. If the mean of these two proportions exceeds θ_m, then the windows are considered to belong to a single cluster and are merged, otherwise not.

The output of the algorithm is a number of sets that define the final clusters discovered in the original dataset.

2.2 Artificial Neural Networks

Artificial Feedforward Neural Networks (FNNs) are parallel computational models comprised of densely interconnected, simple, adaptive processing units, characterized by an inherent propensity for storing experiential knowledge and rendering it available for use. Two critical parameters for the successful application of FNNs are the appropriate selection of network architecture and training algorithm. The problem of identifying the optimal network architecture for a specific task remains up to date an open and challenging problem. For the general problem of function approximation, the *universal approximation theorem* proved in [5, 20] 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, insufficient number of hidden units, or the lack of a deterministic relationship between the input and the target. A second theorem proved in [13] provides an upper bound for the architecture of an FNN destined to approximate a continuous function defined on the hypercube in $\mathbb{R}^n$.

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.*

The efficient supervised training of FNNs is the subject of considerable ongoing research and numerous algorithms have been proposed to this end. Supervised training amounts to the global minimization of the network error function E. The rapid computation of a set of weights that

minimizes this error is a rather difficult task since, in general, the number of network weights is large and the resulting error function generates a complex surface in the weight space, characterized by multiple local minima and broad flat regions adjoined to narrow steep ones. Next, a brief exposition of the training algorithms considered is provided.

3 Numerical Results

We have applied the previously described methodology to the daily (interbank rate) time–series of the Japanese Yen against the U.S. Dollar. The series consists of 1827 observations spanning a period of five years, from the 1st of January 1998 until the 1st of January of 2003. The series is freely available from `www.oanda.com`. The training set contained the first 1500 patterns, while the remaining patterns, covering approximately the final year of data, were assigned to the test set. Numerical experiments were performed using a Clustering C++ and a Neural Network C++ Interface built under the Fedora Linux 1.0 operating system using the GNU compiler collection (gcc) version $3.3.2$.

Applying the method of "False Nearest Neighbors" [7] on the training set we observed that the proportion of false nearest neighbors drops sharply to the value of 0.334% for an embedding dimension of $d = 5$. For larger values of d the proportion of false nearest neighbors lies in the neighborhood of 0.067%, up to $d = 19$ for which the number of false nearest neighbors drops to zero. The embedding dimension chosen for this series was 5.

Having identified the appropriate embedding dimension, the UKW algorithm is employed to compute the clusters present in the training set. Pattern n is of the form $p_n = [x_n, x_{n+1}, \ldots, x_{n+d-1}, x_{n+d}, \ldots, x_{n+d+h-1}], n = 1, \ldots, 1500$, and $h = 2, 5$ represents the forecasting horizon. In other words, the values to be predicted $[x_{n+d}, \ldots, x_{n+d+h-1}]$, are components of the pattern vectors employed by the UKW algorithm. For the two–step–ahead prediction problem, a total of 15 clusters were identified in the training set, while for the five–step–ahead task, UKW detected 28 clusters in the training set. To identify the cluster to which a pattern from the test set belongs, it is first necessary to find the window whose center is closest (in terms of Euclidean distance) to that pattern. The pattern is then assigned to the cluster to which this window belongs. Since the future values of the series are unknown for the patterns of the test set, distances from window centers are computed by excluding the components $[x_{n+d}, \ldots, x_{n+d+h-1}]$ of the window center vector from the computation of distances.

As previously mentioned, the issue of selecting the optimal network architecture for a particular task, remains up to date an open and challenging problem. After experimentation with networks with one and more hidden layers, we concluded that 5–5–4–1 constitutes an appropriate architecture for the FNNs used as local predictors. The FNNs associated with each cluster detected in the training set were trained to minimize the mean squared error of one–step–ahead prediction. Four training algorithms were considered:

- Adaptive On–Line Back Propagation (AOBP) [8].
- Scaled Conjugate Gradient Descent (SCG) [11],
- Improved Resilient Back Propagation (iRPROP) [6],
- Resilient Back Propagation (RPROP) [16], and
- Back Propagation with Variable Stepsize (BPVS) [9].

As an additional evaluation criterion, the performance of the FNNs on the task of two– and five–step–ahead prediction on the training set was monitored. The accuracy of the multiple–step–ahead forecasts was assessed by the percentage of correct *sign* prediction [4, 18]. This measure captures the percentage of forecasts in the test set for which the following inequality is satisfied:

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

where, $\widehat{x_{t+d+h-1}}$ represents the prediction generated by the FNN, $x_{t+d+h-1}$ refers to the true value of the exchange rate at period $t + d + h - 1$ 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 employed [18].

Having trained all the FNNs for 100 epochs, their performance on the task of two– and five–step–ahead prediction was evaluated on the test set. For the clusters to which patterns from the test set were assigned, Tables 1, 2 and 3 report the minimum ($\min$), mean, maximum ($\max$) performance with respect to correct sign prediction. Also the standard deviation (st.dev), as well as, the performance of the FNN that managed the highest multiple–step–ahead sign prediction on the train set (best ms) is reported. The number of test patterns that were assigned to each cluster is reported next to the cluster index. Due to space limitations, the results for one cluster containing four patterns from the test set is not reported in Table 1 for the two–step–ahead problem, while for the five–step–ahead task the results for three clusters containing one, four and five patterns respectively are not reported in Tables 2, 3.

Primarily, it is important to note that patterns from the test set were assigned to a subset of the total number of clusters detected in the training set. For the two–step–ahead prediction task, patterns from the test set were assigned to 9 out of the 15 clusters discovered in the training set. For the five–step–ahead task, patterns from the test set were assigned to 14 out of the 28 clusters. This implies that only a subset of the information contained in the training set was considered relevant for predicting the evolution of the series in the test set. Inspecting the results reported in Tables 1–3, it is evident that the degree of predictability varies substantially among the different clusters.

Cluster 5: 13 patterns

	min	mean	max	st.dev.	best ms
AOBP	0.46	0.46	0.46	0.0	0.46
SCG	0.46	0.49	0.61	0.05	0.53
iRPROP	0.46	0.46	0.46	0.0	0.46
RPROP	0.46	0.46	0.46	0.0	0.46
BPVS	0.46	0.53	0.61	0.07	0.46

Cluster 6: 39 patterns

	min	mean	max	st.dev.	best ms
AOBP	0.46	0.56	0.61	0.05	0.46
SCG	0.43	0.57	0.61	0.07	0.61
iRPROP	0.56	0.60	0.61	0.01	0.61
RPROP	0.61	0.61	0.61	0.0	0.61
BPVS	0.35	0.55	0.66	0.10	0.61

Cluster 7: 64 patterns

	min	mean	max	st.dev.	best ms
AOBP	0.37	0.41	0.43	0.01	0.42
SCG	0.45	0.45	0.45	0.0	0.45
iRPROP	0.40	0.44	0.45	0.01	0.40
RPROP	0.45	0.45	0.45	0.0	0.45
BPVS	0.40	0.43	0.46	0.02	0.40

Cluster 8: 42 patterns

	min	mean	max	st.dev.	best ms
AOBP	0.38	0.46	0.5	0.04	0.38
SCG	0.35	0.50	0.54	0.06	0.35
iRPROP	0.52	0.52	0.54	0.00	0.52
RPROP	0.5	0.52	0.54	0.01	0.54
BPVS	0.38	0.48	0.52	0.04	0.38

Cluster 9: 60 patterns

	min	mean	max	st.dev.	best ms
AOBP	0.51	0.57	0.61	0.03	0.60
SCG	0.46	0.50	0.58	0.03	0.58
iRPROP	0.43	0.45	0.48	0.01	0.45
RPROP	0.43	0.47	0.48	0.01	0.48
BPVS	0.43	0.43	0.48	0.01	0.48

Cluster 10: 23 patterns

	min	mean	max	st.dev.	best ms
AOBP	0.47	0.53	0.56	0.03	0.52
SCG	0.47	0.54	0.56	0.03	0.56
iRPROP	0.52	0.56	0.60	0.03	0.56
RPROP	0.52	0.54	0.60	0.03	0.52
BPVS	0.52	0.55	0.60	0.02	0.56

Cluster 11: 25 patterns

	min	mean	max	st.dev.	best ms
AOBP	0.56	0.61	0.72	0.06	0.56
SCG	0.52	0.52	0.6	0.02	0.52
iRPROP	0.52	0.52	0.6	0.02	0.60
RPROP	0.52	0.52	0.52	0.0	0.52
BPVS	0.44	0.56	0.72	0.10	0.44

Cluster 12: 50 patterns

	min	mean	max	st.dev.	best ms
AOBP	0.42	0.45	0.50	0.03	0.48
SCG	0.44	0.51	0.52	0.02	0.44
iRPROP	0.52	0.52	0.56	0.01	0.56
RPROP	0.52	0.52	0.52	0.0	0.52
BPVS	0.44	0.51	0.54	0.03	0.44

Table 1. Results for the problem of 2–step ahead prediction

On the task of two–step ahead prediction (Table 1), no FNN was able to achieve a correct sign prediction exceeding 50% for the patterns that were classified to cluster 7. A similar behavior is observed for clusters $17, 18, 19$, and 20 for the five–step–ahead prediction task. On the other hand, the minimum correct sign prediction exceeds 50%

Cluster 11: 35 patterns

	min	mean	max	st.dev.	best ms
AOBP	0.51	0.53	0.54	0.01	0.54
SCG	0.34	0.48	0.54	0.07	0.45
iRPROP	0.48	0.54	0.62	0.04	0.48
RPROP	0.48	0.51	0.54	0.01	0.51
BPVS	0.25	0.44	0.62	0.11	0.45

Cluster 12: 17 patterns

	min	mean	max	st.dev.	best ms
AOBP	0.35	0.42	0.52	0.07	0.52
SCG	0.17	0.27	0.35	0.05	0.29
iRPROP	0.29	0.43	0.52	0.10	0.52
RPROP	0.17	0.33	0.52	0.13	0.52
BPVS	0.17	0.40	0.52	0.13	0.52

Cluster 13: 9 patterns

	min	mean	max	st.dev.	best ms
AOBP	0.22	0.24	0.33	0.04	0.22
SCG	0.22	0.23	0.33	0.03	0.22
iRPROP	0.22	0.28	0.44	0.07	0.22
RPROP	0.22	0.26	0.33	0.05	0.33
BPVS	0.22	0.36	0.44	0.07	0.33

Cluster 14: 75 patterns

	min	mean	max	st.dev.	best ms
AOBP	0.54	0.56	0.57	0.0	0.56
SCG	0.49	0.55	0.58	0.02	0.49
iRPROP	0.52	0.56	0.58	0.01	0.57
RPROP	0.54	0.56	0.58	0.01	0.57
BPVS	0.48	0.50	0.52	0.01	0.48

Cluster 15: 64 patterns

	min	mean	max	st.dev.	best ms
AOBP	0.59	0.60	0.60	0.0	0.59
SCG	0.57	0.60	0.60	0.0	0.60
iRPROP	0.56	0.60	0.64	0.02	0.60
RPROP	0.56	0.60	0.62	0.01	0.59
BPVS	0.57	0.60	0.64	0.01	0.60

Cluster 16: 15 patterns

	min	mean	max	st.dev.	best ms
AOBP	0.46	0.46	0.46	0.0	0.46
SCG	0.4	0.46	0.46	0.02	0.46
iRPROP	0.33	0.45	0.53	0.05	0.46
RPROP	0.26	0.41	0.46	0.06	0.40
BPVS	0.46	0.48	0.53	0.03	0.46

Cluster 17: 16 patterns

	min	mean	max	st.dev.	best ms
AOBP	0.25	0.25	0.25	0.0	0.25
SCG	0.18	0.40	0.5	0.12	0.18
iRPROP	0.25	0.28	0.43	0.06	0.25
RPROP	0.18	0.35	0.5	0.11	0.18
BPVS	0.25	0.29	0.43	0.07	0.25

Cluster 18: 9 patterns

	min	mean	max	st.dev.	best ms
AOBP	0.33	0.33	0.33	0.0	0.33
SCG	0.11	0.13	0.33	0.07	0.11
iRPROP	0.11	0.26	0.44	0.11	0.22
RPROP	0.11	0.18	0.33	0.07	0.11
BPVS	0.11	0.31	0.33	0.07	0.33

Cluster 19: 17 patterns

	min	mean	max	st.dev.	best ms
AOBP	0.35	0.39	0.41	0.02	0.41
SCG	0.17	0.23	0.29	0.02	0.23
iRPROP	0.17	0.30	0.41	0.08	0.41
RPROP	0.23	0.32	0.41	0.04	0.41
BPVS	0.17	0.25	0.29	0.04	0.29

Table 2. Results for the problem of 5–step ahead prediction

	Cluster 20: 10 patterns				
	min	mean	max	st.dev.	best ms
AOBP	0.20	0.26	0.30	0.05	0.20
SCG	0.20	0.24	0.40	0.06	0.30
iRPROP	0.20	0.31	0.40	0.05	0.30
RPROP	0.20	0.26	0.30	0.05	0.30
BPVS	0.30	0.30	0.30	0.0	0.30
	Cluster 21: 40 patterns				
	min	mean	max	st.dev.	best ms
AOBP	0.55	0.55	0.55	0.0	0.55
SCG	0.55	0.58	0.60	0.01	0.55
iRPROP	0.60	0.60	0.62	0.00	0.60
RPROP	0.57	0.60	0.62	0.01	0.60
BPVS	0.32	0.52	0.62	0.08	0.47

Table 3. Results for the problem of 5–step ahead prediction – continued

for most training algorithms in clusters 10 and 11 of Table 1 and clusters 14, 15, and 21 of Tables 2 and 3. Furthermore, it is important to note that in most cases the FNNs that achieved the best performance on the task of two– and five–step–ahead prediction on the training set were rarely the ones that exhibited the highest performance on the test set. Selecting among the trained FNNs for each cluster the one with the highest performance with respect to minimum, mean, maximum and highest multi–step–prediction accuracy on the training set, respectively, we computed the mean forecasting performance achieved on the entire test set. These results are illustrated in Table 4 for the two– and five–step–ahead tasks. As expected the accuracy of the forecasts deteriorates as the forecasting horizon is expanded.

	min	mean	max	best ms
2–step–ahead	0.51	0.53	0.575	0.55
5–step–ahead	0.48	0.51	0.56	0.51

Table 4. Overall forecasting accuracy achieved by selecting the best performing FNN with respect to min, mean, max, and best ms, respectively

Since the embedding dimension used to construct the input patterns for the FNNs acting as local predictors was five, to perform six–step–ahead prediction through the aforementioned approach, implies that all the elements of input vector are previous outputs of the model. In other words, the problem becomes one of iterated (closed–loop) prediction. We have tested the performance of the system on this task, but the model fails to keep track of the evolution of the series. In effect beyond a certain number of iterated predictions the output of the model converges, to a constant value, implying that the system has been trapped in a fixed point. Enhancing the model so as to be able to overcome this limitation is a very interesting problem which we intend to address in future work.

4 Conclusions

This paper presents a time series forecasting methodology which draws from the disciplines of chaotic time series analysis, clustering, and artificial neural networks. The methodology consists of four stages. Primarily the minimum dimension necessary for phase space reconstruction through time–delayed embedding is calculated using the method of false nearest neighbors. To identify neighborhoods in the state space, time delayed vectors are subjected to clustering through the UKW algorithm. This algorithm has the capability to endogenously determine the number of clusters present in a dataset. Subsequently, a different feedforward neural network is trained on each cluster. Having completed the training of the networks, the performance of the model on the task of multiple–step–ahead prediction is evaluated on the test set. Beyond this point the system uses both predicted and true values of the series in order to formulate the patterns that will be used to forecast the evolution of the series. This methodology was applied to generate two– and five–step–ahead predictions for the time–series of the daily exchange rate of the Japanese Yen against the US Dollar for a period of time which covers approximately the final year of available data. The obtained results were promising.

In future work we intend to address the issue of iterated prediction. To this end we aim to incorporate the test proposed by Diks et al. [2] so as to obtain a measure of the extent to which the developed prediction system has the ability to accurately capture the attractor of the measured data, during the training process [1]. We also intend to consider recurrent neural networks.

References

[1] R. Bakker, J. C. Schouten, C. L. Giles, F. Takens, and C. M. van den Bleek, *Learning of chaotic attractors by neural networks*, Neural Computation **12** (2000), no. 10, 2355–2383.

[2] C. Diks, W. R. van Zwet, F. Takens, and J. DeGoede, *Detecting differences between delay vector distributions*, Physical Review E **53** (1996), no. 3, 2169–2176.

[3] J. D. Farmer and J. J. Sidorowich, *Predicting chaotic time series*, Physical Review Letters **59** (1987), no. 8, 845–848.

[4] L. C. Giles, , S. Lawrence, and A. H. Tsoi, *Noisy time series prediction using a recurrent neural network and grammatical inference*, Machine Learning **44** (2001), no. 1/2, 161–183.

[5] K. Hornik, *Multilayer feedforward networks are universal approximators*, Neural Networks **2** (1989), 359–366.

[6] C. Igel and M. Hüsken, *Improving the Rprop learning algorithm*, Proceedings of the Second International ICSC Symposium on Neural Computation (NC 2000) (H. Bothe and R. Rojas, eds.), ICSC Academic Press, 2000, pp. 115–121.

[7] M. B. Kennel, R. Brown, and H. D. Abarbanel, *Determining embedding dimension for phase–space reconstruction using a geometrical construction*, Physical Review A **45** (1992), no. 6, 3403–3411.

[8] G.D. Magoulas, V.P. Plagianakos, and M.N. Vrahatis, *Adaptive stepsize algorithms for on-line training of neural networks*, Nonlinear Analysis, T.M.A. **47** (2001), no. 5, 3425–3430.

[9] G.D. Magoulas, M.N. Vrahatis, and G.S. Androulakis, *Effective backpropagation training with variable stepsize*, Neural Networks **10** (1997), no. 1, 69–82.

[10] T. P. Meyer and N. H. Packard, *Local forecasting of high dimensional chaotic dynamics*, Nonlinear Modelling and Forecasting (M. Casdagli and S. Eubank, eds.), Addison-Wesley, 1992, pp. 249–264.

[11] M. Moller, *A scaled conjugate gradient algorithm for fast supervised learning*, Neural Networks **6** (1993), 525–533.

[12] N. G. Pavlidis, D. K. Tasoulis, and M. N. Vrahatis, *Financial forecasting through unsupervised clustering and evolutionary trained neural networks*, Proceedings of the Congress on Evolutionary Computation (CEC 2003), 2003, pp. 2314–2321.

[13] A. Pinkus, *Approximation theory of the MLP model in neural networks*, Acta Numerica (1999), 143–195.

[14] J. C. Principe, A. Rathie, and J. M. Kuo, *Prediction of chaotic time series with neural networks and the issue of dynamic modeling*, Int. J. of Bifurcation and Chaos **2** (1992), no. 4, 989–996.

[15] J. C. Principe, L. Wang, and M. A. Motter, *Local dynamic modeling with self–organizing maps and applications to nonlinear system identification and control*, Proceedings of the IEEE, no. 6, 1998, pp. 2240–2257.

[16] M. Riedmiller and H. Braun, *A direct adaptive method for faster backpropagation learning: The rprop algorithm*, Proceedings of the IEEE International Conference on Neural Networks, San Francisco, CA, 1993, pp. 586–591.

[17] M.N. Vrahatis, B. Boutsinas, P. Alevizos, and G. Pavlides, *The new k-windows algorithm for improving the k-means clustering algorithm*, Journal of Complexity **18** (2002), 375–391.

[18] S. Walczak, *An empirical analysis of data requirements for financial forecasting with neural networks*, Journal of Management Information Systems **17** (2001), no. 4, 203–222.

[19] A. S. Weigend, M. Mangeas, and A. N. Srivastava, *Nonlinear gated experts for time series: Discovering regimes and avoiding overfitting*, International Journal of Neural Systems **6** (1995), 373–399.

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