S. Singh. and P. McAtackney. Dynamic Time-series Forecasting using Local Approximation, Proc. 10th IEEE International Conference on Tools with AI, Taiwan, IEEE Press, pp. 392-399 (10-12 November, 1998).

Dynamic Time-Series Forecasting using Local Approximation

Sameer Singh
Department of Computer Science
University of Exeter, Exeter EX4 4PT, UK
{sameer@dcs.exeter.ac.uk}

Paul McAtackney
School of Computing
University of Plymouth, Plymouth PL4 8AA, UK
{pmcatack@soc.plym.ac.uk}

ABSTRACT

Pattern recognition techniques for time-series forecasting are beginning to be realised as an important tool for predicting chaotic behaviour of dynamic systems. In this paper we develop the concept of a Pattern Modelling and Recognition System which is used for predicting future behaviour of time-series using local approximation. In this paper we compare this forecasting tool with neural networks. We also study the effect of noise filtering on the performance of the proposed system. Fourier analysis is used for noise-filtering the time-series. The results show that Fourier analysis is an important tool for improving the performance of the proposed forecasting system. The results are discussed on three benchmark series and the real US S&P financial index.

1. Introduction

Forecasting is important in several domains and a large number of studies have used classical statistical methods for predicting series behaviour. Advanced methods such as neural networks [1,7,12], genetic algorithms [4], Markov models [9] and fuzzy methods have also been frequently used [4]. Farmer and Sidorowich [6] have found that chaotic time-series prediction is several order of magnitude better using local approximation techniques than universal approximators. Local approximation refers to the concept of breaking up the domain into several small neighbourhood regions and analysing these separately. Unlike global approximators, local approximation techniques are dependent on the number of free parameters. The well-known methods of local approximation for time-series prediction include chart representations, nearest neighbour methods [6] and pattern imitation techniques [10]. In this paper, we propose a new method of local approximation that will be implemented in our Pattern Modelling and Pattern Recognition system.

Noisy time-series are common in several scientific and financial domains. Noisy time-series may or may not be completely random in nature. Previous studies on the effect of noise on classification and optimisation systems have suggested that controlled addition of noise to input data gives better results. A number of studies on neural networks have reported superior network training on noise contaminated data (Burton and Mpitos [3]; Jim et al. [8]; Murray and Edwards [11]). Fuzzy nearest neighbour methods for pattern recognition also perform better with noise, Singh [13]. In this paper we attempt to quantify the forecasting performance of the proposed local approximation method with noise-filtered time-series. The noise within a time-series signal may be identified using Fourier analysis, Brillinger [2]. In this paper, we filter four different time-series under consideration. A Fourier spectrum is generated for each time-series and noise is removed in a stepwise manner. This paper first studies the comparative performance of the Pattern Modelling and Recognition System. We then study the usefulness of noise-filtering for improving the performance of the tool.

2. Local approximation using PMRS

The main emphasis of local approximation techniques is to model the current state of the time-series for matching its key features with past states. If we choose to represent a time-series as $\mathbf{y} = \{y_1, y_2, ... y_n\}$, then the state of the time-series represents its current value y_n. One simple method of prediction may be based on identifying the closest neighbour of y_n in the past data, say y_j, and predicting y_{n+1}

on the basis of y_{j+1}. This simple approach may be extended by taking an average prediction based on a set of nearest neighbours [6]. The definition of the current state of a time-series may be extended to include more than one value, e.g. the current state of size two may be defined as $\{y_{n-1}, y_n\}$. In this paper, we will also refer to states as patterns used in the matching process. In theory, we can have a current state of any size but in practice matching current states of optimal size to past states of the same size will only yield the best results. Unfortunately, optimal state size must be determined experimentally on the basis of achieving minimal errors on standard measures.

In order to illustrate the matching process for series prediction, we need to define the mathematical representation of time-series data for our purposes. Consider the time series as a vector $\mathbf{y} = \{y_1, y_2, \ldots y_n\}$ where n is the total number of points in the series. Often, we also represent such a series as a function of time, e.g. $y_n = y_t$, $y_{n-1} = y_{t-1}$, and so on. A segment in the series may be defined as a difference vector $\boldsymbol{\delta} = (\delta_1, \delta_2, \ldots \delta_{n-1})$ where $\delta_i = y_{i+1} - y_i$, $\forall i$, $1 \le i \le n-1$. A pattern contains one or more segments and may be visualised as a string of segments $\rho = (\delta_i, \delta_{i+1}, \ldots \delta_h)$ for given values of i and h, $1 \le i,h \le n-1$, provided that $h>i$. In order to define any pattern mathematically, we choose to encode the time series y as a vector of change in direction. For this purpose, a value y_i is encoded as 0 if $y_{i+1} < y_i$, as a 1 if $y_{i+1} > y_i$ and a 2 if $y_{i+1} = y_i$. Formally, a pattern in the time-series may now be represented as $\rho = (b_i, b_{i+1}, \ldots b_h)$ where b is a binary value in most cases, either a 1 or a 0, expect in cases when the series doesn't change which is encoded as a 2.

The complete time-series is encoded as $(b_1, \ldots b_{n-1})$. For a total of k segments in a pattern, it is encoded as a string of k *b* values. For a pattern of size k, the total number of pattern shapes possible is $2^k + 1$. The technique of matching structural primitives is based on the premise that the past repeats itself. Farmer and Sidorowich [6] state that the dynamic behaviour of time-series can be efficiently predicted by using local approximation. For this purpose, a map between current states and the nearest neighbour past states can be generated for forecasting (see Figure 1).

Pattern matching in the context of time-series forecasting refers to the process of matching current state of the time series with its past states. Consider the encoded time series $(b_1, b_i, \ldots b_{n-1})$. Suppose that we are at time *n* (y_n) trying to predict y_{n+1}. A pattern of size k is first formulated from the last k values in the series, $\rho' = (b_{n-k}, \ldots b_{n-1})$. The size k of the structural primitive (pattern) used for matching has an important impact on minimising the error and correctly predicting the direction of series change. The pattern size and matching procedure itself must be optimised for obtaining the best results. The aim of a pattern matching algorithm is to find the closest match of pattern ρ' in the historical data (estimation period) and use this for predicting y_{n+1}. The magnitude and direction of prediction depend on the match found. The success in correctly predicting series depends directly on the pattern matching algorithm. The overall algorithm is shown as a flowchart in Figure 2.

Figure 2 shows the implementation of the Pattern Modelling and Recognition System for forecasting. The first step is to select a state/pattern of minimal size (k=2). A nearest neighbour of this pattern is determined from historical data on the basis of smallest offset ∇. The nearest neighbour position in the past data is termed as "marker". There are three cases for prediction: either we predict high, we predict low, or we predict that the future value is the same as the current value. The prediction $\ddot{y}_{n+1}$ is scaled on the basis of the closeness of the match found. We use three standard error measures for estimating the accuracy of the forecast [5]. These are: Mean Square Error measure which defines the sum of square difference between actual and predicted values for the forecasts made; Mean Absolute Percentage Error measure which defines the difference between actual and predicted values for the forecasts made as a ratio of the current value in percentage; and finally the direction % success measures the ratio in percentage of the number of times the actual and predicted values move in the same direction to the total number of predictions. The process is repeated for states/patterns of size greater than two and a model with smallest k and minimal error is selected.

3. Time-series data

In this paper, the analysis data has been selected from varied domains including physics, astrophysics and finance. Three of the benchmark series (A, D and E) considered here come from the Santa Fe competition [14]. The fourth series is the real S&P index for US financial market (monthly data from 1988 to 1996). The details of these univariate series are introduced below:

Series A : This is a univariate time series measured in a Physics laboratory experiment. This data set was selected because it is an example of complicated behaviour in a clean, stationary, low-dimensional non-trivial physical system for which the underlying dynamic equations are well understood. There are a total of 1024 observations (Figure 3)

Series D: This univariate time-series has been generated for the equation of motion of a dynamic particle. The series has been synthetically generated with relatively high-dimensional dynamics. There are a total of 4096 observations (Figure 4)

Series E: This univariate time-series is a set of astrophysical data (variation in light intensity of a star). The data set was selected because the information is very noisy, discontinuous and non-linear. There are a total of 2048 observations (Figure 5)

Series S&P: This series represents the S&P index over a period of eight years. This data is noisy and exponentially increasing in nature. There are a total of 2048 observations. (Figure 6)

4. Fourier analysis

Fourier Analysis is used to obtain the spectra for all time-series under consideration. The spectrum for any given series shows the energy content at discrete frequencies for that series. To produce the required spectra, a 'Discrete Fourier Transform', DFT, is used. This DFT is of the form,

$$F(j\varpi) = \sum_{n=0}^{N-1} f_n \cdot e^{-j\varpi nT}$$

$$\Delta\varpi = \frac{2\cdot\pi}{T\cdot N}$$

The frequency intervals, $\Delta\varpi$, are set such that the number of frequency samples produced is equal to the number of time-series observations, N, i.e.

$$0 \le \varpi \le \frac{2\cdot\pi}{T}$$

By inspection of each spectrum, a value of cut-off frequency, α, is determined whereby the energy content below this value, i.e. the low frequency range, represents the underlying 'trend' energy of the series. Therefore, by definition, the energy above α represents 'noise'.

The discrete frequency range between α and the digital bandwidth of the time-series is segmented into 10% intervals. Starting from the high end of the frequency range, discrete frequency contributions are removed in steps of 10% until α is reached. Each of the 10 modified spectra produced is then Inverse Fourier transformed back to its equivalent 'filtered' time-series (folds). These series are then used as the source data for the time-series prediction algorithm and the results are analysed to determine the optimal 'filter cut-off frequency', α.

This process is repeated for all four time-series under consideration and the optimal value for α is obtained in each case. A discussion on the results obtained is contained in section 5. It is important for practical purposes than any Fourier transformed series predictions, are converted into original predictions. The predictions made on the Fourier transformed series must be transformed to original predictions. This may be achieved in a straight-forward manner by deriving a mathematical equation showing the relationship between the Fourier transformed series and the original series. An equation with a low standard error of estimate is qualified by a high correlation (r) between the two series. The correlation between the original and Fourier transformed series is equal to: r = .998 for series A; r = .986 for series D; r = .992 for series E; and r = .875 for series S&P. These high correlations indicate that it will be relatively easy to transform the Fourier performance into actual performance. For this reason we performed regression analysis to determine the relationship between the Fourier transformed and original series to convert predictions on Fourier transformed series into original series predictions. The results in Table1 are for the original series after this translation (predictions for actuals).

5. Results

In all experiments, results are first produced on predicting the difference series and these predictions are translated to the original predictions (this is necessary for non-stationary data). The PMRS performance is compared with the non-linear neural networks. Neural network development is described below.

5.1 Input/output selection

Gately [7] notes that several outputs may be selected for neural networks. We could predict the recent series change, the direction of series change, change in value over the forecast period, whether the trend is up or down, the rate of change, the quality of making a trade at a given time, volatility or change in volatility. In our analysis, we develop two separate neural networks: one for predicting the change in direction (up or down movement) and the other for predicting actual value of the index. For the first network, the output is coded as a 1 if the series moves up, and a 0 if it goes down. For the second network, we first predict the ratio of change in series to the actual, and translate the results to actual predictions. Hence, the output code is a real number between [0, 1] range.

The selection of inputs depends on what is being forecast (outputs). We use a linear procedure for determining the inputs to the two networks. We perform a linear regression analysis for predicting the output using the last ten values of the series, i.e. (y_{t-9}, … y_t). The relative importance of these inputs in correctly predicting the output is measured using t-statistic. The five most important variables are selected for further analysis. Here we make an assumption that linear input selection is appropriate for a further non-linear analysis; this is in line with the work done by Refenes et al. [12]. Hence, for both networks there are five inputs. There is one output which in the first network is a 0 or 1 to predict whether the series goes up or down, and a real number for the second network to predict the actual changes. The number of hidden nodes in Figure 7 are selected using the procedure discussed below.

5.2 Network architecture/hidden nodes

We follow the guidelines set by Weiss and Kulikowski [15]. The aim is to select a model of minimal complexity which leads to minimal generalisation error. We increment the number of hidden nodes in a stepwise manner to achieve the optimal configuration.

Training procedure. The training procedure consists of dividing the data into two parts: estimation period data and test period data. The estimation period data is used in training. We use 90% of the total data in the estimation period and the remaining 10% in the test set. The training/test procedure is performed using the Stuttgart Neural Network Simulator using the standard backpropagation algorithm.

The PMRS performance has been selected for optimal pattern size (k) used for matching patterns and the Exponential smoothing method for the optimal value of constant α. The optimal value of parameter k, once determined for a series, remains fixed over all predictions made. These performances are compared on standard performance measures discussed earlier. Ideally, the MSE and MAPE values should be as low as possible and the direction success % values as high as possible.

The results are shown in Table 1. On Series A, PMRS approach is the most accurate (least mean squared error). However, the neural network method is better by 5% on correctly predicting the direction of series change. On Series D, the PMRS and NN method are in close competition. The PMRS approach has the least mean squared error and is 2% better on predicting the direction of series change, whereas the neural network method is slightly better on the MAPE measure. On Series E, the PMRS approach is again more accurate (least mean squared error), the MAPE measurements are close, but the neural network approach significantly outperforms the PMRS method this time on correctly predicting the direction of series change (better by 16%). Finally, on series S&P, neural networks are more accuracy, but the PMRS method has a slight edge on MAPE and correctly predicting series change measures. The results of noise-filtering are shown for the PMRS tool with a (F) symbol. The results are shown for the optimal value of cut-off frequency α which was obtained for the least mean square error. The amount of energy taken out of the series in percentage is: series A (2.98%); series D (16.81%); series E (15.27%) and series S&P (5.60%). On all series, the filtered data gives better prediction that original data. However, neural networks are the best method for predicting the US financial index.

We also plot the last 50 forecasts against the actuals for both Neural Networks and PMRS. For the US financial index, the data comes from the summer of 1996. These are shown in Figs. 8-11. The plots show how the accuracy of the two methods when making single forecasts.

6. Conclusion

In this paper we have used a novel method of local approximation in time-series forecasting. The pattern recognition method is based on the philosophy that it is easier to approximate the chaotic behaviour of time-series at the local than global level. The results have been produced using this approach on untreated original series and Fourier transformed noise filtered series. The results are highly encouraging. In most cases, noise-filtered series is easier to predict. The main conclusion of this study is that PMRS is a useful tool for forecasting time-series behaviour, and filtering noise can be a very useful pre-processing stage in making accurate forecasts. The PMRS performance is in close competition and often superior than neural networks tested on our data and further research should now be devoted to enhancing pattern recognition based tools for forecasting.

References

[1] Azoff, M. E. *Neural network time series forecasting of financial markets*, John Wiley and Sons, 1994.

[2] Brillinger, D. R. *Time series: Data analysis and theory*, McGraw-Hill, 1981.

[3] Burton, R. M. and Mpitsos, G. J., "Event-dependent control of noise enhances learning in neural networks," *Neural Networks*, vol. **5**, pp. 627-637, 1992.

[4] Chorafas, D. N. *Chaos theory in the financial markets: Applying fractals, fuzzy logic, genetic algorithms, Swarn simulation & the Monte Carlo method to manage markets*, Probus Publishing Company, 1994.

[5] Delurgio, S. *Forecasting: Principles and applications*, McGraw-Hill, 1998.

[6] Farmer, J. D. and Sidorowich, J. J. Predicting chaotic dynamics, in *Dynamic patterns in complex systems*, J. A. S.

Kelso, A. J. Mandell and M. F. Shlesinger (Eds.), pp. 265-292, Singapore: World Scientific, 1988.

[7] Gately, E. *Neural Networks for Financial Forecasting*, John Wiley, 1996.

[8] Jim, K., Horne, B. and Giles, C. L. "Effects of noise on convergence and generalisation in recurrent networks," *Neural information processing systems* 7, G. Tesauro, D. Touretzky and T. Leen (eds.), MIT Press, p. 649, 1995.

[9] MacDonald, I. L. and Zucchini, W. *Hidden Markov and other models for discrete-valued time-series*, London: Chapman and Hall, 1997.

[10] Motnikar, B. S., Pisanski, T. and Cepar, D. Time-series forecasting by pattern imitation, *OR Spektrum*, 18(1), pp. 43-49, 1996.

[11] Murray A. F. and Edwards, P. J. "Synaptic weight noise during multilayer perceptron training: Fault tolerance and training improvements," *IEEE Transactions on Neural Networks*, vol. 4, no. 4, pp. 722-725, 1993.

[12] Refenes, A. N., Burgess, A. N. and Bentz, Y. "Neural networks in financial engineering: A study in methodology," *IEEE Transactions on Neural Networks*, vol. 8, no. 6, 1997.

[13] Singh, S. "Effect of noise on generalisation in Massively Parallel Fuzzy Systems," (in press, *Pattern Recognition*)

[14] Weigend, A. S. and Gersehnfield, N. A. 1994 *Time series prediction: Forecasting the future and understanding the past*. eds. Reading, MA:Addison-Wesley.

[15] Weiss, S.M. and Kulikowski, C.A. *Computer systems that learn*, Morgan Kaufmann, San Mateo, CA, 1991.

Figure 1. Matching current states with the past states using PMRS algorithm

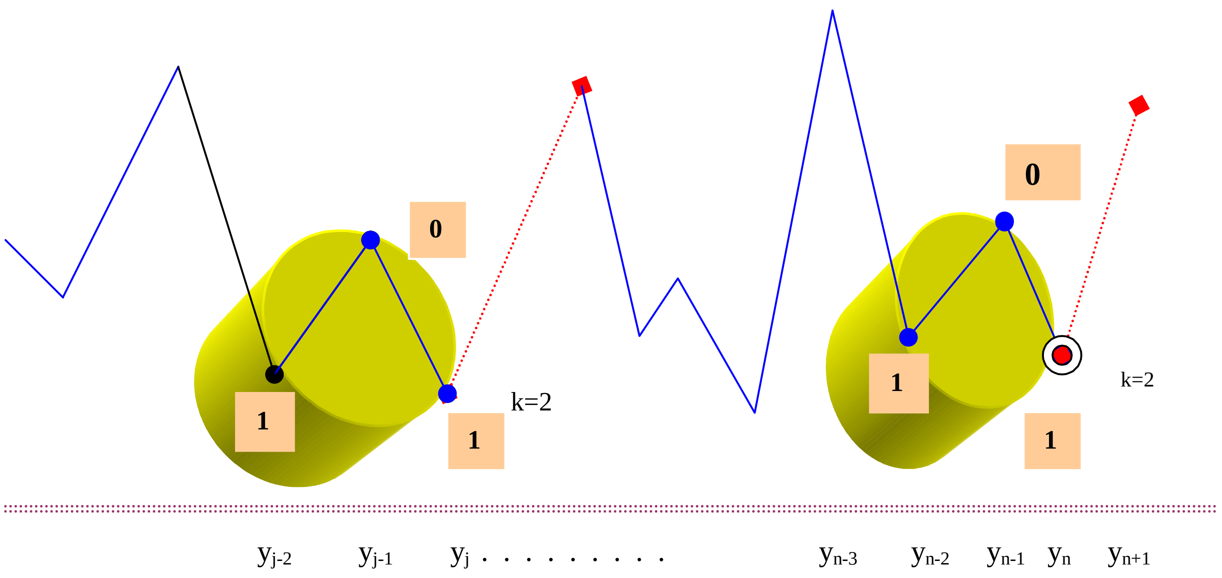


Table 1. The prediction performance of the PMRS, ES and NN methods of forecasting on the original untreated series for a test size of 10% of total data

Series	Method	k	MSE	MAPE	direction success %	% size predicted
A	PMRS	4	202.4	11.8	95	*10*
A	PMRS(F)	4	200.0	12.4	99	*10*
A	NN	-	605.3	10.8	100	*10*
D	PMRS	3	.003	8.6	84	*10*
D	PMRS(F)	3	.0005	2.9	95	*10*
D	NN	-	.036	7.9	82	*10*
E	PMRS	2	.015	8.6	66	*10*
E	PMRS(F)	2	.003	8.9	62	*10*
E	NN	-	.078	8.0	82	*10*
S&P	PMRS	3	82.3	1.1	76	*10*
S&P	PMRS(F)	3	55.5	5.5	72	*10*
S&P	NN	-	9.8	1.5	75	*10*

Figure 2. Flowchart for the PMRS forecasting algorithm

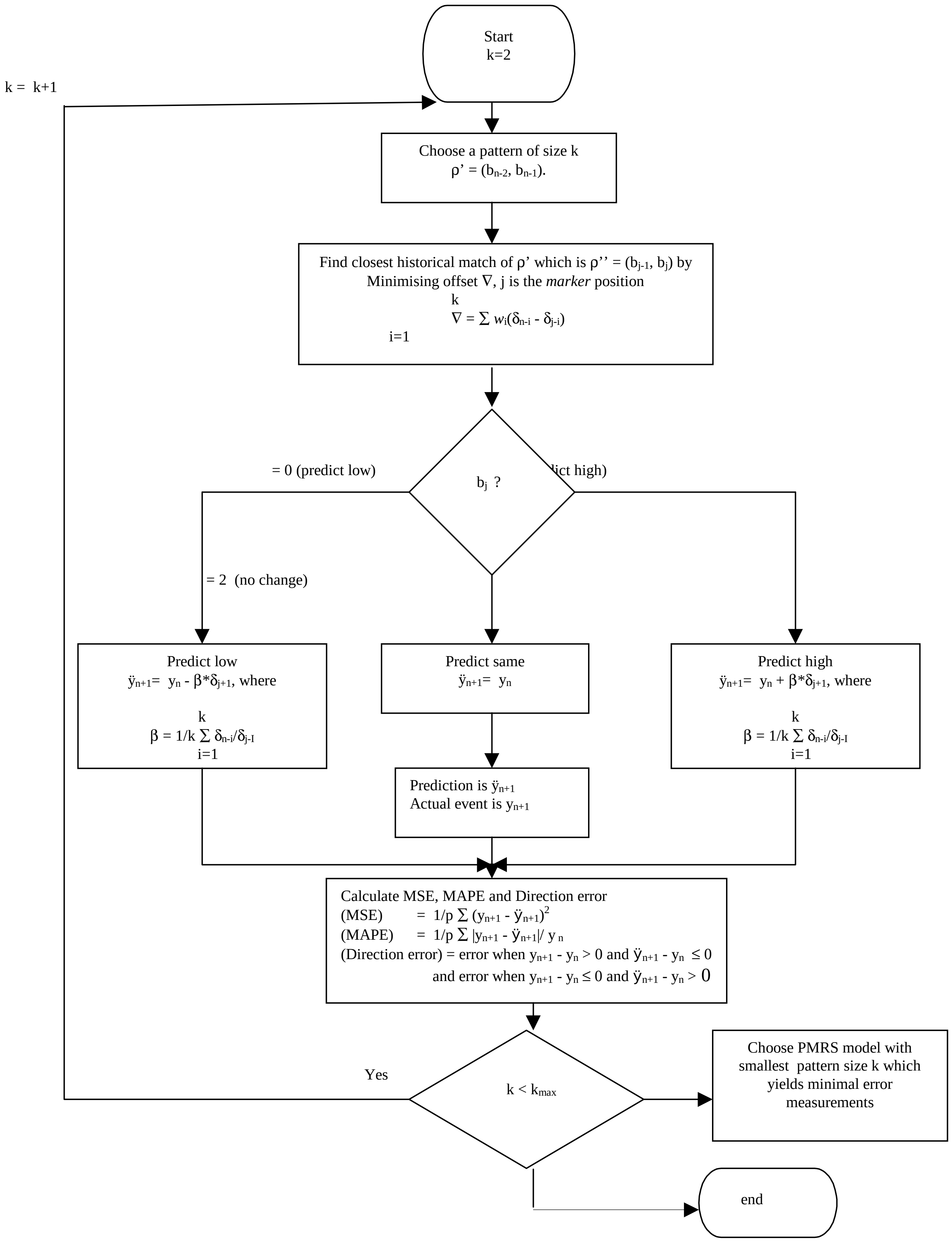

Start k=2
k = k+1
Choose a pattern of size k $\rho' = (b_{n-2}, b_{n-1})$.
Find closest historical match of ρ' which is $\rho'' = (b_{j-1}, b_j)$ by Minimising offset ∇, j is the *marker* position $\nabla = \sum_{i=1}^{k} w_i(\delta_{n-i} - \delta_{j-i})$
= 0 (predict low)
ict high)
b_j ?
= 2 (no change)
Predict low $\ddot{y}_{n+1} = y_n - \beta^*\delta_{j+1}$, where $\beta = 1/k \sum_{i=1}^{k} \delta_{n-i}/\delta_{j-I}$
Predict same $\ddot{y}_{n+1} = y_n$
Predict high $\ddot{y}_{n+1} = y_n + \beta^*\delta_{j+1}$, where $\beta = 1/k \sum_{i=1}^{k} \delta_{n-i}/\delta_{j-I}$
Prediction is $\ddot{y}_{n+1}$ Actual event is y_{n+1}
Calculate MSE, MAPE and Direction error
(MSE) $= 1/p \sum (y_{n+1} - \ddot{y}_{n+1})^2$
(MAPE) $= 1/p \sum |y_{n+1} - \ddot{y}_{n+1}| / y_n$
(Direction error) = error when $y_{n+1} - y_n > 0$ and $\ddot{y}_{n+1} - y_n \leq 0$
and error when $y_{n+1} - y_n \leq 0$ and $\ddot{y}_{n+1} - y_n > 0$
Yes
$k < k_{max}$
Choose PMRS model with smallest pattern size k which yields minimal error measurements
end

Figure 3. Plot of Series A

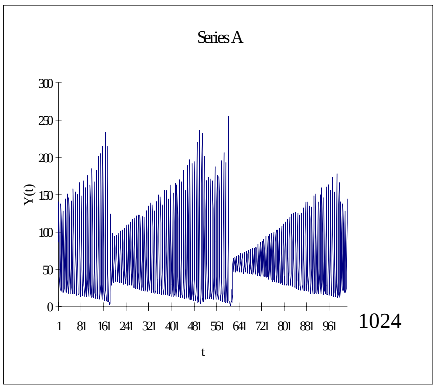


Figure 4. Plot of series D

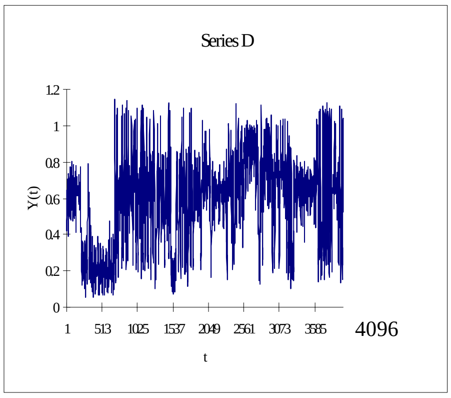


Figure 5. Plot of series E

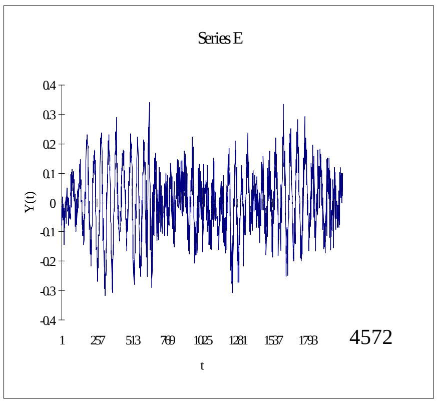


Figure 6. Plot of Series S&P

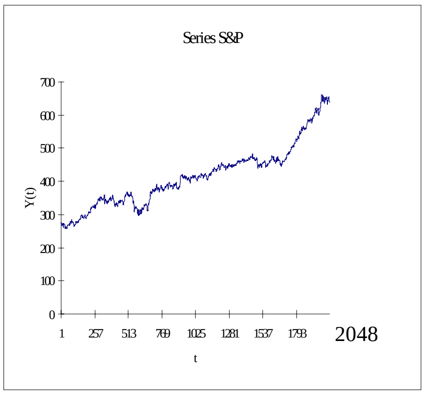


Figure 8. Last 50 predictions on Series A using the Neural Network and PMRS approach

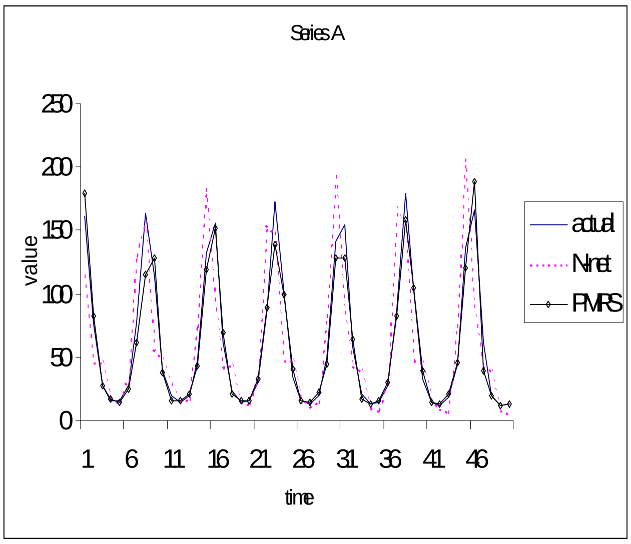


Figure 9. Last 50 predictions on Series D using the Neural Network and PMRS approach

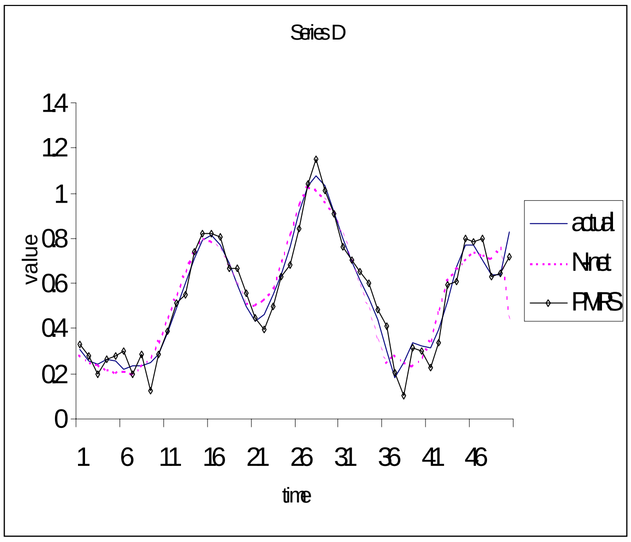


Figure 10. Last 50 predictions on Series E using the Neural Network and PMRS approach

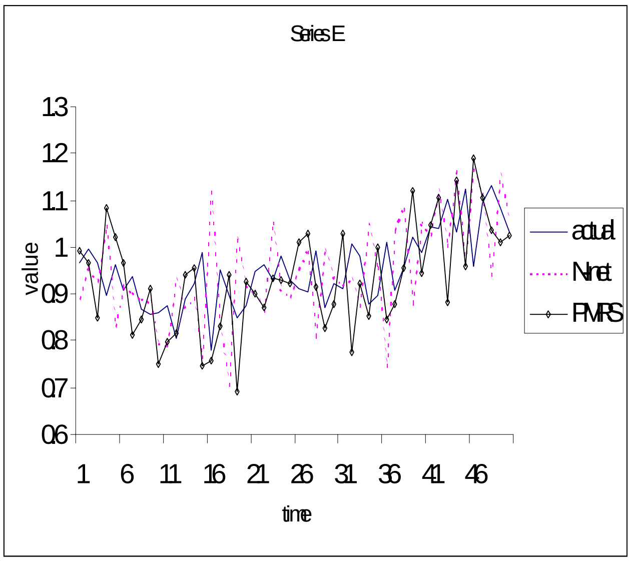


Figure 11. Last 50 predictions on Series S&P using the Neural Network and PMRS approach

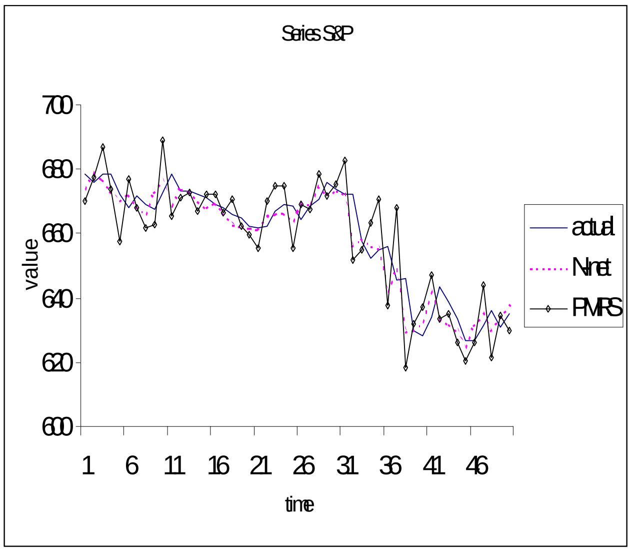
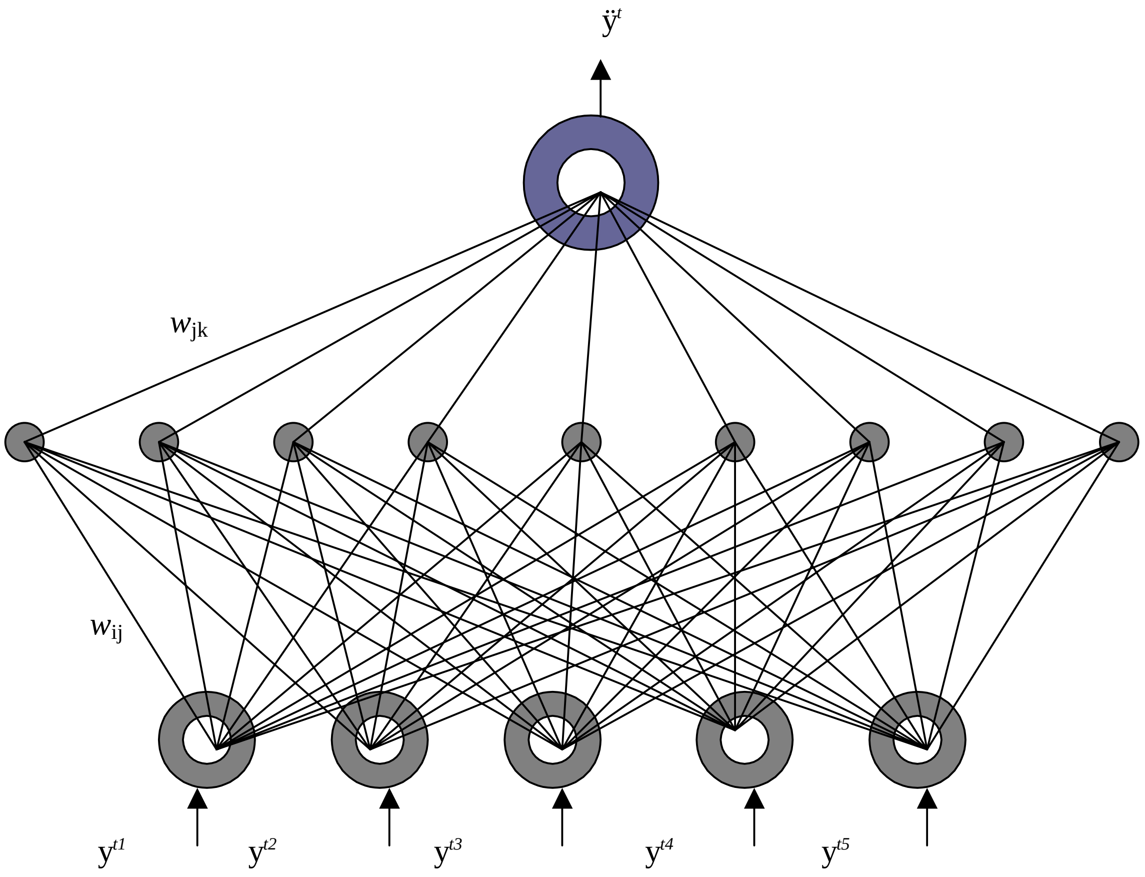


Figure 7. Neural Network architecture for time-series forecasting


ÿ^t
w_jk
w_ij
y^t1
y^t2
y^t3
y^t4
y^t5