

# Time Series Data Analysis and Pre-process on Large Databases

Gongde Guo, Hui Wang and David Bell  
*School of Information and Software Engineering*  
*University of Ulster, UK*

## Abstract

In this paper we introduce a novel classification algorithm called MCC (Minimal Cover Classification), which works well for numerical data and categorical data. Given a new data tuple, it provides values for each class that measures the likelihood of the tuple belonging to that class. We then apply the MCC algorithm on real stock market data to predict the 'upward' or 'downward' trend of  $k$ -days stock returns. To improve the prediction accuracy we use the discrete Fourier transform and its inverse transform to filter noise whilst preserving the trend of global movement of time series in the time domain. The experimental result shows that the MCC algorithm is comparable to C4.5. Using MCC as a mining algorithm to predict the 'upward' or 'downward' trend of  $k$ -day stock returns, the average hit rate on pre-processed data is 20.55% higher than that on the original data. This means that the prediction accuracy has been remarkably improved by means of the proposed MCC algorithm on noise filtered time series.

## 1 Introduction

Time series account for a large amount of the data stored in databases. A common task with a time series database is to look for an occurrence of a particular pattern within a longer sequence. Such queries have obvious applications in many fields, such as identifying patterns associated with growth in stock prices or identifying non-obvious relationships between two time series of weather data, or detecting anomalies in an online robot monitoring system.

Usually if raw data is filled with noise this could affect the mining algorithm's accuracy. In the stock market for example, the closing price of each day is influenced daily by various factors and there is a lot of noise as a result, making it difficult to observe long-term features. Therefore, there is a clear advantage if we pre-process the original raw data and work on the pre-processed information. However, since the data sets occurring in practice tend to be very large, most of the work has focused on the design of efficient algorithms for various mining problems and, most notably, the search of similar (sub) sequences with respect to a variety of measures [1,2,3,4,10]. Given the magnitude of many time series databases, much research [5,6,7,8,9] has been devoted to speeding up the search process.

Surprisingly, little work has been done to develop efficient algorithms to analyse and mine on noise filtered time series in the time domain. Such work is of crucial importance since noise in original time series directly affects a mining algorithm's accuracy.

The remainder of the paper is organised as follows. Section 2 introduces a noise filtering method. Section 3 defines a stock prediction problem. Section 4 introduces a novel classification algorithm and describes how to use the algorithm to predict the 'upward' or 'downward' trend of stock returns. The experimental results are described and the evaluation is shown in section 5. Section 6 completes the paper with a discussion and suggestions for further work.

## 2 Time Series Noise Filtering

In the stock market application domain as an example, the closing price of each day is composed from a mixture of daily random events and long-term trends. We therefore need to pre-process the raw data in order to produce cleaner data with as little extraneous noise as possible.

Assume that the raw time series  $d_{raw}(t)$  is composed additively from a long-term signal  $d(t)$  and noise  $n(t)$ , that is  $d_{raw}(t) = d(t) + n(t)$ . The cleaning operation is expected to produce  $\hat{d}(t)$ , an estimation of the long-term signal  $d(t)$  by removing  $n(t)$ . In order to do so, we characterize the signal  $d(t)$  and the noise  $n(t)$ . The noise signal is of random nature and is influenced daily from various sources. In contrast, the long-term signal is stable, deterministic, and influenced by relatively few factors. If we apply the Fourier transform, we can identify the long-term signal  $d(t)$  as it is constructed mainly from waves with low frequency (slow changes over time), while the noise signal is constructed from waves with high frequency (fast changes over time). In this paper, we use the real discrete Fourier transform (RDFT) and its inverse operation (IRDFT) to filter noise from the raw time series.

The  $n$ -point *Real Discrete Fourier Transform* of a signal  $\vec{x} = [x_t], t=0, 1, \dots, n-1$  is defined to be a sequence  $\vec{X}$  of  $n/2+1$  complex numbers  $X_f, f=0, 1, \dots, n-1$ , given by

$$X_f = R_f + i I_f, \text{ where } R_f = \sum_{t=0}^{n-1} x_t \cos(2\mathbf{p}ft/n), f=0, 1, \dots, n/2$$

$$\text{and } I_f = \sum_{t=0}^{n-1} x_t \sin(2\mathbf{p}ft/n), f=1, \dots, (n/2)-1 \quad (1)$$

In eqn (1),  $i$  is the imaginary unit. The signal  $\vec{x}$  can be recovered by the inverse transform:

$$x_t = (R_0 + R_{n/2} \cos(\mathbf{p}t))/2 + \sum_{f=1}^{n/2-1} R_f \cos(2\mathbf{p}ft/n)$$

$$+ \sum_{f=1}^{n/2-1} I_f \sin(2\mathbf{p}ft/n), t=0, \dots, n-1 \quad (2)$$

To efficiently filter noise, we add a parameter  $m$  to eqn (2) to control the number of coefficients used to invert data from frequency domain to time domain. Eqn (2) is then changed to equ (3).

$$x_t = (R_0 + R_{n/2} \cos(\mathbf{p}t))/2 + \sum_{f=1}^m R_f \cos(2\mathbf{p}ft/n)$$

$$+ \sum_{f=1}^m I_f \sin(2\mathbf{p}ft/n), t=0, \dots, n-1 \quad (3)$$

To grasp the idea here, the best way is by means of an example, so we graphically illustrate the transform results of one stock called ABF randomly chosen from London Stock Exchange over the period 1995-1999 (from 6<sup>th</sup> Oct. 1995 to 8<sup>th</sup> Sept. 1999). The original time series of ABF shown in Figure 1 includes 1024 numbers, each representing the price of the stock at the end of an operational day. At first we use the real discrete Fourier transform to transform the original time series from the time domain to the frequency domain, and then invert it from the frequency domain to the time domain using only the first few of coefficients in the frequency domain - abandoning the rest of Fourier coefficients. The transformed results are shown in Figure 2 to Figure 3 using different number coefficients to invert a time series from the frequency domain to the time domain.

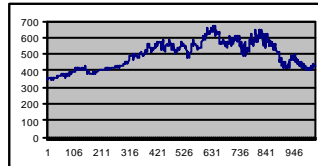


Figure 1. The original data set which contains 1024 data points.

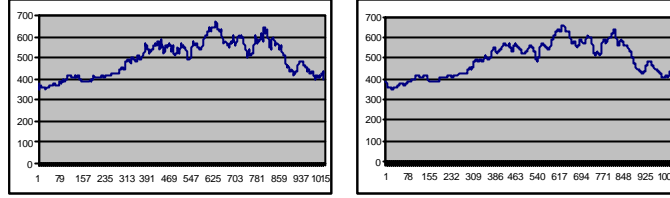


Figure 2. The noise filtering results when  $m$  is set to 256, 128 respectively.

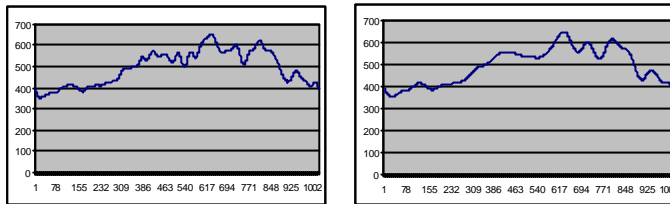


Figure 3. The noise filtering results when  $m$  is set to 64, 32 respectively.

On comparison, it is obvious that the transformed time series has some merit as follows:

- (1) it preserves the global movement trend of the original time series.
- (2) it keeps the time localization property between the original time series and the preprocessed time series in the time domain.
- (3) it eliminates the high frequencies which can be seen as a kind of noise in the original time series.

As the pre-processed time series has these features it suggests a means of abandoning the original time series, and letting any mining algorithms directly carry out operations on the transformed data to improve the mining accuracy.

### 3 Defining Stock Prediction Problem

To make our ideas and mechanisms concrete, we continue to use our stock market problem for illustration. A traditional way to define the stock prediction problem is to view the stock returns as a time series  $R_k(t)$  [10]. For example,  $k$ -day returns  $R_k(t)$  are defined as

$$R_k(t) = 100 \cdot \frac{Close(t) - Close(t-k)}{Close(t-k)} \quad (4)$$

The returns  $R_k(t)$  are the primary target in most research on the predictability of stocks. Similar targets can be identified in other application domains. Some of the reasons for using it are:

- (1)  $R_k(t)$  has relatively constant range even with the input of many years of data. The raw prices  $Close(t)$  obviously vary much more and make it difficult to create a valid model for longer periods of time.
- (2)  $R_k(t)$  for different stocks may be compared on an equal basis.

- (3) It is easy to evaluate a prediction algorithm for  $R_k(t)$  by computing the prediction accuracy of the sign of  $R_k(t)$ . A long time accuracy above 50% indicates that a good prediction has taken place.

Assume that the predictions of stock prices for time  $t$  based on the previous  $k$  days are expressed by the time series  $\{\hat{C}losd(t), t=k+1, \dots, N\}$ . The actual prices are denoted by the time series  $\{Closd(t), t=1, \dots, N\}$ . The predictions of the  $k$ -day return at time  $t$  are denoted by the time series  $\{\hat{R}_k(t), t=k+1, \dots, N\}$ . The actual returns are denoted by the time series  $\{R_k(t), t=k+1, \dots, N\}$ .

To predict the  $k$  day return in the future,  $R_k(t)$  is assumed to be a function  $g$  of the  $q$  previous (lagged) values in the same time series.

$$R_k(t) = g(R_k(t-k), R_k(t-k-1), \dots, R_k(t-k-q-1)) \quad (5)$$

The task for the learning or modeling process is to find the function  $g$  that best approximates a given set of measured data.

To evaluate a prediction algorithm for  $R_k(t)$ , we modify eqn (5) to eqn (6) by computing the prediction accuracy of the sign of  $R_k(t)$  and use *hit rate* [11] as a performance metric.

$$S_k(t) = g(R_k(t-k), R_k(t-k-1), \dots, R_k(t-k-q-1)), \text{ where } S_k(t) = \begin{cases} 1: R_k(t) > 0 \\ -1: R_k(t) < 0 \end{cases} \quad (6)$$

The *hit rate* of a stock return is defined as

$$H_R = \frac{\left| \left\{ S_k(t) \hat{S}_k(t) > 0 \right\}_{k+1}^N \right|}{\left| \left\{ S_k(t) \hat{S}_k(t) \neq 0 \right\}_{k+1}^N \right|} \quad (7)$$

It indicates how often the sign of the return is correctly predicted in a prediction algorithm. It is computed as the ratio between the number of correct non-zero predictions  $\hat{S}_k(t)$  and the total number of non-zero moves in the stock time series. Refer to [11] for the reason why both zero predictions and zero returns are removed from the computation of the hit rate.

The stock return prediction problem is changed to classification problem by modifying eqn (5) to eqn (6) and it is obvious that the classification accuracy is the same as the hit rate of the sign of  $\hat{S}_k(t)$  in eqn (7).

## 4 Classification Algorithm

A classification algorithm called MCC was designed to be used for predicting in the stock market. Before describing the MCC algorithm, we introduce some notation used in our MCC algorithm first. A tuple is called a *hyper tuple* if its entries are sets for categorical data or intervals for numerical data instead of single values; a tuple is called a *simple tuple* if all its entries have a cardinality of 1. The '+' operator represents the set union operation for categorical data, and the interval merging operation for numerical data respectively. Suppose  $y$  is a simple tuple denoted  $(y_1, y_2, \dots, y_m)$ ,  $Z$  is a hyper tuple denoted  $(Z_1, Z_2, \dots, Z_m)$ .

Assume that  $Z_m$  is a decision attribute, 'y is covered by Z' means  $y_i \in Z_i$  ( $Z_i$  is a set) for categorical attribute and  $Z_{i1} \leq y_i \leq Z_{i2}$  ( $Z_i$  is an interval denoted  $[Z_{i1}, Z_{i2}]$ ) for numerical attribute, where  $i=1, 2, \dots, m-1$ . MCC is described as follows:

Let  $D$  be a dataset,  $D = \{d_1, d_2, \dots, d_n\}$ .  $d_i$  is a simple tuple,  $d_i = (a_{i1}, a_{i2}, \dots, a_{im})$ , in which,  $a_{im}$  is a decision attribute. Suppose that  $S$  is a finite set of class labels denoted as  $S = (s_1, s_2, \dots, s_p)$ , we have  $f(d_i) = a_{im} \in S$ . Given a new data point  $d$  (a simple tuple), classification is done in the following way:

- (1) Calculate  $CI(d, x)$  and  $C2(d, x)$  for all  $x$  in  $D$ , where  $CI(d, x) = \{y \mid y \in D, f(y) \neq f(x) \text{ and } y \text{ is covered by } d+x\}$ .  $C2(d, x) = \{y \mid y \in D, f(y) = f(x) \text{ and } y \text{ is covered by } d+x\}$ .
- (2) Pick up all the  $x_i$  having  $CI(d, x_i) = 0$ , if there is just one  $x_i \in D$  ( $i=1$ ) such that  $d+x_i$  (a hyper tuple) does not overlap any  $y$  in  $D$ , where  $y \neq x_i$  and  $f(y) \neq f(x_i)$ , then classify  $d$  as  $f(x_i)$ .
- (3) If there are more than one  $x_i$  denoted  $x_1, x_2, \dots, x_i, i \geq 2$  such that  $CI(d, x_i) = 0$ . In this situation, majority voting of class label among the  $x_i$  is used to decide the class label for  $d$ . If the number of  $x_i$  with  $CI(d, x_i) = 0$  belonging to different classes is the same, we calculate their  $Sum(d, s_j)$  - the sum of all  $C2(d, x_i)$  that  $x_i \in s_j$  with  $CI(d, x_i) = 0$  for all  $s_j \in S$  and then classify  $d$  by  $s_k = \max\{Sum(d, s_j) \text{ for any } s_j \text{ in } S\}$ .

An experiment using the 5-fold cross validation method has been carried out to evaluate the MCC prediction accuracy, and to compare the experimental results with C4.5 as our benchmark. The latter is implemented in the Clementine' software package. Seven public datasets were chosen from the UCI machine learning repository. Some information about these datasets is listed in Table 1.

Table 1. General information about the data sets.

Data set	NA	NN	NO	NB	NE	CD
Aust.	14	4	6	4	690	383:307
Diab.	8	0	8	0	768	268:500
Hear.	13	3	7	3	270	120:150
Iris	4	0	4	0	150	50:50:50
Germ.	20	11	7	2	1000	700:300
TTT	9	9	0	0	958	332:626
Vote	18	0	0	18	232	108:124

In Table 1, the meaning of the title in each column is follows: NA-Number of attributes, NN-Number of Nominal attributes, NO-Number of Ordinal attributes, NB-Number of Binary attributes, NE-Number of Example, and CD-Class Distribution.

The comparison of C4.5 and MCC in testing accuracy (TA) using 5-fold cross validation method is shown in Table 2.

Table 2. A comparison of C4.5 with MCC

Data set	TA: C4.5	TA: MCC
Aust.	85.2	86.0
Diab.	72.9	72.0
Hear.	77.1	76.0
Iris	94.0	95.0
Germ.	72.5	76.6
TTT	86.2	86.3
Vote	96.1	97.0
Average	83.43	84.13

In Table 2, the meaning of TA is Testing Accuracy. Obviously, The MCC is an extremely simple and general classification algorithm. Experimental results show that MCC is comparable to C4.5.

## 5 Experiment Results

The ultimate goal of noise filtering is to improve the prediction accuracy of the proposed algorithm. So we designed an experiment to evaluate it to see how well it performed in prediction with real world time series on the original data and on the transformed data.

Ten stocks closing prices were randomly chosen and collected from London Stock Exchange in 1024 trading days. The general information about the chosen stocks time series is shown in Table 3.

Table 3. General information about the chosen stocks time series

Stock name	Beginning time	Ending time	Trading days
ABF	6 <sup>th</sup> Oct. 1995	8 <sup>th</sup> Sept. 1999	1024
BAY	6 <sup>th</sup> Oct. 1995	8 <sup>th</sup> Sept. 1999	1024
CCM	6 <sup>th</sup> Oct. 1995	8 <sup>th</sup> Sept. 1999	1024
SDR	6 <sup>th</sup> Oct. 1995	8 <sup>th</sup> Sept. 1999	1024
VOD	6 <sup>th</sup> Oct. 1995	8 <sup>th</sup> Sept. 1999	1024
LOG	6 <sup>th</sup> Oct. 1995	8 <sup>th</sup> Sept. 1999	1024
KGF	6 <sup>th</sup> Oct. 1995	8 <sup>th</sup> Sept. 1999	1024
WWP	6 <sup>th</sup> Oct. 1995	8 <sup>th</sup> Sept. 1999	1024
NGG	8 <sup>th</sup> Dec. 1995	10 <sup>th</sup> Nov. 1999	1024
RTK	17 <sup>th</sup> May. 1996	19 <sup>th</sup> Apr. 2000	1024

All the original time series were pre-processed by using eqn (1) and eqn (3) with  $m=128$  to filter the noise. A pre-processed time series with the same size of original data was obtained for each time series. Then, for each original time series and its transformed time series we replaced daily stock price by its daily stock return in terms of eqn (4).

To predict the sign of the stock returns  $k$ -day in the future, a sliding window technique has been used to translate the original time series and the transformed time series from one-dimensional space into multidimensional space in terms of eqn (5). The window size was set to 6 in the experiment. So each data point in the multidimensional space had 6 attributes. The 6<sup>th</sup> attribute was characterised using a class label and was filled in with the sign of its value in terms of eqn (6).

When all this was done, we used the first four fifths of data points for training and the last one fifth of data points for predicting in the multidimensional space. This is a better alternative than cross validation, since data at time  $t$  and at time  $t+k$ ,  $k>0$  is often correlated. (Consider for example the returns  $R_k(t)$  and  $R_k(t+1)$ ). In such a case, predicting a function value  $R_k(t_j+1)$  using a model trained with data  $t>t_j$  is cheating and should obviously be avoided.

In experiments, we use MCC as a mining algorithm to predict the ‘up’ or ‘down’ trend of  $k$ -day stock returns on both the original time series and the pre-processed time series. The experimental results were shown in Table 4 and Table5.

In Table 5, the average hit rate of predicting the sign of the stock returns  $k$ -day ( $k=1,2,5,10$ ) in the future on the original time series is only 49.49%, very close to random prediction. But predicting on the pre-processed time series, the average hit rate is up to 59.66%. This means that the prediction accuracy has been remarkably improved by means of the proposed noise filtering method in this paper.

Table 4. Predictions of  $k$ -day stock returns using MCC algorithm

Stock name	1-day (O) %	1-day (T) %	2-days (O) %	2-days (T) %
ABF	51.06	65.12	53.81	65.61
BAY	49.48	62.68	47.24	62.68
CCM	49.74	61.22	52.02	60.73
KGF	47.40	61.22	50.25	62.20
LOG	57.74	62.68	52.91	60.73
NGG	47.03	62.68	50.52	62.68
RTK	51.56	66.59	44.44	59.12
SDR	48.90	64.15	53.85	59.27
VOD	60.10	65.12	58.08	63.73
WWP	50.85	60.73	49.74	58.39
Average	51.39	63.22	51.29	61.51

Stock name	5-days (O) %	5-days (T) %	10-days (O) %	10-days (T) %
ABF	38.69	63.73	42.29	60.88
BAY	52.97	62.18	41.09	58.53
CCM	51.72	57.39	51.24	54.46
KGF	58.91	58.90	47.03	54.56
LOG	56.78	55.45	44.28	52.14
NGG	45.96	57.39	46.03	54.62
RTK	44.55	57.76	47.76	56.92
SDR	44.50	55.98	45.27	54.67
VOD	55.45	57.35	47.76	55.41
WWP	52.26	56.57	38.19	53.59
Average	50.18	58.27	45.09	55.64

In Table 4, the symbol (O) represents original data, (T) represents transformed data.

Table 5. A comparison of the average hit rates

$k$ -day	Average Hit Rate on Original data (%)	Average Hit Rate on Transformed data (%)
1	51.39	63.22
2	51.29	61.51
5	50.18	58.27
10	45.09	55.64
Average	49.49	59.66

## 6 Conclusion

In this paper we use the real discrete Fourier transform to filter noise with the aim of improving the prediction ability in time series – in particular financial time series. We also developed a novel classification algorithm MCC which was used to predict the ‘up’ or ‘down’ trend of  $k$ -day stock returns. This experimental results show that the proposed MCC algorithm despite its simplicity is a very competitive data-mining algorithm. The experimental results further show applying MCC algorithm on real stock market to predict  $k$ -day stock return the average hit rate predicting on the pre-processed time series is statically 20.55% higher than that on the original time series.

Further research is required into how to efficiently reduce the original time series to improve the performance of the MCC algorithm for efficiently predicting long-term stock movement trend.

## References

- [1] Agrawal, R., Faloutsos, C., and Swami, A. *Efficient Similarity Search in Sequence Databases*. In *Proc. of the Fourth International Conference on Foundations of data Organization and Algorithms*, 1993.

- [2] Agrawal, R., Lin, K., Sawhney, H. S., and Shim, K. "Fast Similarity Search in the Presence of Noise, Scaling, and Translation in Time Series Database", In *Proc. of VLDB'95*.
- [3] Das, G., Gunopulos, D., and Mannila, H. *Finding Similar Time Series*, In *Proc. KDD97*, p. 88-100.
- [4] Das, G., Kin, K., Mannia, H., Renganathan., and G., Smyth, P. *Rule Discovery from Time Series*, In *Proc. KDD98*, p.16-22.
- [5] Faloutsos, C., Ranganathan, M., & Manolopoulos, Y. (1994). *Fast Subsequence Matching in Time-Series Databases*. In *Proc. ACM SIGMOD Conf., Minneapolis*.
- [6] Refiei, D. (1999) *On Similarity-Based Queries for Time Series Data*. In *Proc. of the 15<sup>th</sup> IEEE International Conference on Data Engineering*. Sydney, Australia.
- [7] Yi, B.K., Jagadish, H., & Faloutsos, C. (1998). *Efficient Retrieval of Similar Time Sequences Under Time Warping*. *IEEE International Conference on Data Engineering*, p. 201-208.
- [8] Kin-pong, C., and Ada, W. F. *Efficient Time Series Matching by Wavelets*.
- [9] Keogh, E., Chakrabarti, K., Pazzani, M., and Mehrotra, S. *Dimensionality Reduction for Fast Similarity Search in Large Time Series Databases*.
- [10] Hellstrom, T., and Holmstrom, K. *The Relevance of Trends for Predictions of Stock Returns*. *International Journal of Intelligent Systems in Accounting, Finance & Management*.
- [11] Hellstrom, T. *Data Snooping in the Stock Market*. *Theory of Stochastic Processes Vol.5 (21), no.1-2, 1999*, p.33-50.