Data Reduction and Noise Filtering for Predicting Times Series

Gongde Guo, Hui Wang and David Bell

School of Information and Software Engineering, University of Ulster
Newtownabbey, BT37 0QB, N.Ireland, UK
{G.Guo, H.Wang, DA.Bell} @ulst.ac.uk

Abstract. In this paper we introduce a modification of the real discrete Fourier transform and its inverse transform to filter noise and perform reduction on the data whilst preserving the trend of global moving of time series. The transformed data is still in the same time domain as the original data, and can therefore be directly used by any other mining algorithms.

We also present a classification algorithm MinCov in this paper. Given a new data tuple, it provides values for each class that measures the likelihood of the tuple belonging to that class. The experimental results show that the MinCov algorithm is comparable to C4.5, and using MinCov as a mining algorithm the average hit rate of predicting the sign of stock return is 23.92% higher than that on the original data. This means that the predicting accuracy has been remarkably improved by means of the proposed data reduction and noise filtering method.

1 Introduction

In recent years, there has been a lot of interest within the research community in the mining of time series data. Such data naturally arise in business as well as scientific decision-support applications; examples include stock prices or currency exchange rates, production capacities, sales amounts, biomedical measurements, and weather data collected over time. 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]. Given the magnitude of many time series databases, much research has been devoted to speeding up the search process. The most promising methods are techniques that perform reduction on the data, and then use spatial access methods to index the data in the transform space.

The techniques include the discrete Fourier transform (DFT) introduced in [1] and extended in [5,11,13], and the discrete Wavelet transform (DWT) introduced in [8]. The original work by Agrawal *et al*. utilises the DFT to perform dimensionality reduction. There is also a lot of other research on reducing dimensionality [9] before conducting time series data mining. According to Parseval's theorem [10] the energy in the time domain is the same as the energy in the frequency domain. Besed on Parseval's theorem which links the time and frequency domains most of methods usually transform the original time series from the time domain to the frequency domain. Also for a large number of time series of practical interest, there will be a few frequencies with high amplitude so only the first few frequencies are used to create an efficient spatial index to speed up the search process. However, sometimes some apparently close neighbours under this indexing method are actually poor

matches. Though these false alarms can be detected by examining the corresponding original time series in a post processing stage, the original data has to be reserved for further matching. These methods directly use a few frequencies with high amplitudes in the frequency domain to approximately represent the sub(sequences). This can be seen as a dimensionality reduction. These frequencies are then used to create an index to speed up the search process.

Surprisingly, little work has been done to combine noise filtering and real data reduction in the time domain and hold only the pre-processed data for further analysis. Such work is potentially of crucial importance since noise in very large time series directly affects a mining algorithm's accuracy and performance.

The method proposed in this paper is an attempt to simultaneously remove noise and perform reduction on time series using a modification of the real Fourier transform and its inverse operation while retaining its characteristic profile so that mined results are affected as little as possible. A classification algorithm based on this, called MinCov, was developed and used to predict stock moving trends. MinCov was tested on both the original time series and the pre-processed time series to compare their predicting accuracy and algorithm efficiency.

2 Pre-Processing of Time Series

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. Such pre-processing can include simple filters of a moving average or complicated mathematical transformations such as various Fourier transforms and Wavelet transforms.

2.1 Data Cleaning

In the stock market application domain, 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, a proposed modification of the real discrete Fourier transform (RDFT) and its inverse operation (IRDFT) is used to pre-process the raw data.

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

$$R_f = \sum_{t=0}^{n-1} x_t \ \cos(2\boldsymbol{p}ft/n) \ \text{ and } I_f = \sum_{t=0}^{n-1} x_t \ \sin(2\boldsymbol{p}ft/n), \ \ f=0, …, n/2 \qquad (1.1)$$

where i is the imaginary unit. The signal $\vec{x}$ can be recovered by the inverse transform:

$$x_t = (R_0 + R_{n/2} \cos(\boldsymbol{p}t))/2 + \sum_{f=1}^{n/2-1} R_f \ \cos(2\boldsymbol{p}ft/n) + \sum_{f=1}^{n/2-1} I_f \ \sin(2\boldsymbol{p}ft/n), t=0, …, n\text{-}1 \qquad (1.2)$$

To efficiently filter noise and perform reduction on data, we add a parameters m to equation 1.2 to control the reduction rate. Equation 1.2 is then changed to equation 1.3.

$$x_t = (R_0 + R_{n/(2m)} \cos(\boldsymbol{p}t))/2 + \sum_{f=1}^{n/(2m)-1} R_f \ \cos(2\boldsymbol{p}ftm/n) + \sum_{f=1}^{n/(2m)-1} I_f \ \sin(2\boldsymbol{p}ftm/n), t=0,…, n/m\text{-}1 \qquad (1.3)$$

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 6th Oct. 1995 to 8th 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 (1.1) 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 part of coefficients in the frequency domain - abandoning the rest of Fourier coefficients, having chosen an appropriate reduction rate. The transformed results are shown in Figure2 using different reduction rates to invert a time series from the frequency domain to the time domain.
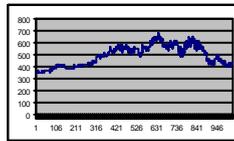

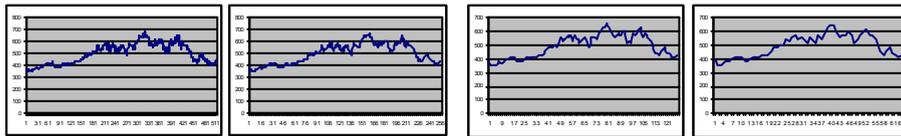**Figure 1.** The original data set which contains 1024 data points

Figure 2. The transformed results when m is set to 2, 4, 8, 16 respectively

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

(1) it preserves the trend of global moving of the original time series.

(2) it eliminates the high frequencies which can be seen as a kind of noise in the original time series.

(3) it reduces the dataset but enhances each datum - the information granularity in the transformed time series is increased.

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

2.2 Define 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 $R_k(t)$ [6]. For example, k-day returns $R_k(t)$ are defined as:

$$R_k(t)=100 \cdot \frac{Close(t) - Close(t-k)}{Close(t-k)} \tag{2.1}$$

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 $\{ \hat{Close}(t) , t=k+1, \ldots, N\}$. The actual prices are denoted by the time series $\{ Close(t) , t=1, \ldots, N\}$. The predictions of the k-day return at time t are denoted by the time series $\{ \hat{R}_k(t) , t=k+1, \ldots, N \}$. The actual returns are denoted by the time series $\{ R_k(t) , t=k+1, \ldots, N\}$.

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.

$$R_k(t)=g(R_k(t-k), R_k(t-k-1), ¼ R_k(t-k-q-1)) \tag{2.2}$$

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 equation 2.2 to equation 2.3 by computing the prediction accuracy of the sign of $R_k(t)$ and use *hit rate* [7] as a performance metric.

$$S_k(t) = g(R_k(t-k), R_k(t-k-1), ¼, R_k(t-k-q-1)), \text{ where } S_k(t) = \begin{cases} 1 : R_k(t) > 0 \\ -1 : R_k(t) < 0 \end{cases} \tag{2.3}$$

The *hit rate* of a stock return is defined as: $H_R = \dfrac{\left| \left\{ S_k(t) \hat{S}_k(t) > 0 \right\}_{k+1}^{N} \right|}{\left| \left\{ S_k(t) \hat{S}_k(t) \neq 0 \right\}_{k+1}^{N} \right|}$.   (2.4)

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.

The reason why both zero predictions and zero returns are removed from the computation of the hit rate is that: if zeros were included, we would have to decide whether the following five combinations should be regarded as "hits" or not:

$\hat{S}_k(t)$	=0	=0	=0	>0	<0
$S_k(t)$	>0	<0	=0	=0	=0

Regardless of the choice made for the classification of these situations, the result is invariably an asymmetric treatment of the positive and negative returns. Since the zero-valued one-day returns can account for more than 20% of the samples in typical stock data, they would result in "Up fractions" arbitrarily either greater or less than 50%. Such a result would conceal the random-walk nature of the time series. By removing all zeros from both predictions and outcome, "Up fractions" very close to 50% are achieved. Therefore, in the case of one-day returns (i.e. $k=1$), a hit rate H_R, significantly greater than 50%, can be regarded as an indicator of true predictions of the sign of the returns.

The stock return prediction problem is changed to classification problem by modifying equation 2.2 to equation 2.3 and it is obvious that the classificataion accuracy is the same as the hit rate of the sign of $\hat{S}_k(t)$ in equation 2.3.

3   Classification Algorithm

A classification algorithm called MinCov was designed to be used for predicting in the stock market. In the MinCov algorithm, 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 $(y_1, y_2,..., y_m)$, Z is a hyper tuple denoted $(Z_1, Z_2,..., Z_m)$. Assume that Z_m is decision attribute, 'y is covered by Z' means $y_i \in Z_i$ (Z_i is a set) for categorical attribute and $Z_{i1} \pounds y_i \leq Z_{i2}$ (Z_i is an interval denoted $[Z_{i1}, Z_{i2}]$) for numerical attribute, where $i=1, 2, ..., m-1$. Refer to [12] for more details. MinCov is then described as follows:

Let $\boldsymbol{D}$ be a dataset, $\boldsymbol{D} = \{ d_1, d_2, ..., d_n \}$. d_i is a simple tuple, $d_i = (a_{i1}, a_{i2}, ..., a_{im})$ labelled by a_{im}, viz. $f(d_i) = a_{im}$. d is a new data point (a simple tuple). Classification is done in the following way:

(1) If there is $x \in \boldsymbol{D}$ such that $d+x$ (a hyper tuple) does not overlap any y in $\boldsymbol{D}$, where $y \neq x$ and $f(y) \neq f(x)$, then classify d as $f(x)$.

(2) Suppose there are more than one tuple, denoted $x_1, x_2, …, x_i$, $i \geq 2$ such that $\forall i$, $d + x_i$ does not overlap any y in D, where $y \neq x_i$ and $f(y) \neq f(x_i)$. In this situation, we calculate the volumn V_i of $d + x_i$. If $d + x_i = (v_1, v_2, …, v_{m-1})$ then V_i is defined as $V_i = |v_1| \times |v_2| \times … \times |v_{m-1}|$, in which, $|v_j|$, $j=1, 2, …, m-1$ is defined as the *number of elements in the set* for categorical data or the *interval length* for numerical data. We classify d as $f(x_k)$, where k satisfies $V_k = \min\{V_1, V_2, …, V_i\}$.

(3) Otherwise classify d by $f(x)$ such that x in D and $d+x$ has the least coverage of y in D where $f(y) \neq f(x)$. This coverage is denoted by $C(d,x)$.

The detailed calculation of $C(d,x)$ is described as follows:

- Given D and d as above, calculate $C(d,x)$ for all x in D, where $C(d,x) = \{y \mid y \in D, f(y) \neq f(x)$ and y is covered by $d+x\}$
- Classify d by $f(x)$, where $|C(d,x)| = \min\{|C(d,y)|$ for *any* y in $D\}$.
- If there are $x_1, x_2, …, x_i$, $i \geq 2$ that $\forall i$, $|C(d, x_i)| = \min\{|C(d,y)|$ for any y in $D\}$. then classify d as $f(x_k)$, where k satisfies $V_k = \min\{V_1, V_2, …, V_i\}$.

A experiment using the 5-fold cross validation method has been carried out to evaluate the MinCov prediction accuracy, and to compare the experimental results with C4.5 as our benchmark. The latter is implemented in the Clementine' software package.

Five public datasets were chosen from the UCI machine learning repository. Some information about these datasets is listed in Table 1 as well as the comparison of C4.5, MinCov in testing accuracy using 5-fold cross validation method.

Table 1. A comparison of C4.5 and MinCov

Dataset	NA	NN	NO	NB	NE	CD	TA:C4.5	TA:MinCov
Aust	14	4	6	4	690	383:307	85.2	86.0
Diab	8	0	8	0	768	268:500	72.9	72.0
Hear	13	3	7	3	270	120:150	77.1	76.0
Iris	4	0	4	0	150	50:50:50	94.0	95.0
Vote	18	0	0	18	232	108:124	96.1	97.0
Average							85.06	85.20

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, CD-Class Distribution, and TA-Testing Accuracy.

The MinCov is an extremely simple and general classification algorithm that works well for both numerical data and categorical data. Given a new data tuple, it provides values for each class which measures the likelihood of the tuple being in that class. Experimental results show that MinCov is comparable to C4.5.

4 Experiment Results

The ultimate goal of data reduction and noise filtering is to improve the prediction accuracy and efficiency 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 2.

Table 2. General information about the chosen stocks time series

No.	Stock name	Beginning time	Ending time	Trading days
1~8	ABF,BAY,CCM,SDR, VOD,LOG,KGF,WWP	6th Oct. 1995	8th Sept. 1999	1024
9	NGG	8th Dec. 1995	10th Nov.1999	1024
10	RTK	17th May. 1996	19th Apr. 2000	1024

All the original time series were pre-processed by using equation 1.1 and equation 1.3 with reduction rate m=8 to filter the noise and perform data reduction. A pre-processed time series with 128 data points transformed from the original time series of 1024 data points 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 equation 2.1.

To predict the sign of the return one 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 equation 2.2. The window size was set to 6 in the experiment. So each data point in the multidimensional space had 6 attributes. The 6th attribute was characterised using a class label and was filled in with the sign of its value in terms of equation 2.3.

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 $R_k(t_1+1)$ using a model trained with data $t>t_1$ is cheating and should obviously be avoided.

Table 3. Predictions of the sign of the stock returns using the MinCov algorithm

No	Stock name	Hit rate (Original Data)	Hit rate (Transformed Data)
1	ABF	51.06	65.38
2	BAY	49.48	61.54
3	CCM	49.74	60.00
4	SDR	48.90	69.23
5	VOD	60.10	64.00
6	LOG	57.74	73.08
7	KGF	47.40	53.85
8	WWP	50.84	68.00
9	NGG	47.03	57.69
10	RTK	51.56	64.00
Average hit rate		51.39	63.68

In table 3, the average hit rate of predicting the sign of the stock returns one day in the future on the original time series is only 51.39%, very close to random prediction. But predicting on the transformed time series, the average hit rate is up to 63.68%. The average hit rate on the pre-processed time series is 23.92% higher than that on the original time series. This means that the prediction accuracy has been remarkably improved by means of the proposed data reduction and noise filtering method in this paper. Moreover, data reduction also improves the performance of MinCov algorithm.

5 Conclusion

We have shown how a sophisticated data pre-processing method can be modified and used for noise filtering and data reduction with the aim of improving the prediction ability in time series – in particular financial time series. The modification of the real discrete Fourier transform and its inverse operation is easily made operational. It "opens up" the time series data to modelling and prediction, and yet the concepts and algorithm are exceedingly simple. The experimental results have shown the efficiency of this approach.

The experimental results also show that the MinCov algorithm is comparable to C4.5 in testing accuracy on five public datasets. Using the MinCov algorithm to predict the moving trend of 10 randomly chosen stocks from London Stock Exchange on the transformed time series, the hit rate is on average 23.92% higher than that on the original time series. The results from initial experiments are quite remarkable with the accuracy up to 63.68%. This shows that the proposed MinCov algorithm despite its simplicity is a very competitive data-mining algorithm.

Further research is required into how to exactly establish the correlation between the original time series and the transformed time series in the time domain for efficiently predicting long-term stock moving 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. Hellstrom, T., and Holmstrom, K. *The Relevance of Trends for Predictions of Stock Returns. Internation Jounnal of Intelligent Systems in Accounting, Finance &Management*
7. Hellstrom, T. *Data Snooping in the Stock Market. Theory of Stochastic Processes Vol.5 (21), no.1-2, 1999, p.33-50..*
8. Kin-pong, C., and Ada, W. F. *Efficient Time Series Matching by Wavelets.*
9. Keogh, E., Chakrabarti, K., Pazzani, M., and Mehrotea, S. *Dimensionality Reduction for Fast Similarity Search in Large Time Series Databases.*
10. Oppenheim, A.V. and Schafer, R.W. *Digital Signal Processing, Prentice-Hall, Englewood Cliffs, N.J.,* 1975.
11. Refiei, D. (1999) *On Similarity-Based Queries for Time Series Data.* In *Proc. of the 15th IEEE International Conference on Data Engineering.* Sydney, Australia.
12. Wang, H., Duntsch, I., and Bell, D. *Data Reduction Based on Hyper Relations.* In *Proc. of KDD98, New York,* p.349-353, 1998.
13. Yi, B,K., Jagadish, H., & Faloutsos, C. (1998). *Efficient Retreval of Similar Time Sequences Under Time Warping. IEEE International Conference on Data Engineering,* p. 201-208.