Statistical Models for Time Sequences Data Mining

Jessica K. Ting, Michael K. Ng*, Hongqiang Rong and Joshua Z. Huang
E-Business Technology Institute
*Department of Mathematics
The University of Hong Kong
Pokfulam Road, Hong Kong, China
Emails: {kwting, hrong, jhuang}@eti.hku.hk, mng@maths.hku.hk

Abstract: *In this paper, we present an adaptive modelling technique for studying past behaviors of objects and predicting the near future events. Our approach is to define a sliding window (of different window sizes) over a time sequence and build autoregression models from subsequences in different windows. The models are representations of past behaviors of the sequence objects. We can use the AR coefficients as features to index subsequences to facilitate the query of subsequences with similar behaviors. We can use a clustering algorithm to group time sequences on their similarity in the feature space. We can also use the AR models for prediction within different windows. Our experiments show that the adaptive model can give better prediction than non-adaptive models.*

Keywords: *Autoregression models, prediction, clustering*

1 INTRODUCTION

A large portion of data in data mining is time dependent. Examples are stock prices, customer credit card spending and mobile phone calls made overtime. Time dependent data is widely used in two important data mining tasks: (1) studying past behaviors of objects in question, for instance, customer spending behaviors in past ten months, and (2) predicting the near future events, for instance, the customers who are likely to delay their credit payments.

A typical form of time dependent data is time sequence (also known as "time series") which is a sequence of observations ordered by time. They arise in many financial, business and scientific applications. Other sequences also exist, for instance spatial sequences and DNA sequences which are ordered by other dimensions rather than time. Here, we only consider time sequences. We denote the observed time sequence by $\mathbf{x} = (x_1, \cdots, x_n)$.

Time sequence data is one of data mining focuses. The area is termed as temporal data mining. One research issue is to create effective indices for large number of time sequences so they can be efficiently retrieved according to their similarities. For example, given a customer's one month credit card spending sequence, find the customers whose spending patterns similar to the the given spending pattern or given a company's stock price sequence in a particular time period, find the companies whose stocks performed similar to the given company. Due to the length of real time sequences, indices are usually created in the feature space instead of the time space. Similarity search is an important topic in the current database research.

In financial and many other applications, building models from past time sequences to predict future events is another important objective in temporal data mining. This topic has well been studied in statistics and other disciplines. Standard modelling techniques are widely available. The objective is usually the models of individual time sequences. However, data mining concerns application of the standard techniques to large number of time sequences. Therefore, scalability of the model building algorithms and efficiency of handling large number of models are important.

In this paper, we present an adaptive modelling technique for time sequence indexing, clustering and prediction. Our approach is to define a sliding window (of different window sizes) over a time sequence and build autoregression models from the subsequences in different windows. The models are representations of past behaviors of the sequence objects. The coefficients of these AR models can be used as features to index subsequences to facilitate the query of subsequences with similar behaviors. We can apply a clustering algorithm to the coefficients to cluster time sequences. We can also use the AR models to predict near future values.

Similarity between two time sequences is usually defined based on the similarity of the curve shapes. For example, the two subsequences $\mathbf{x1}$ and $\mathbf{y1}$ in Figure 1 have a high similarity even though their magnitudes are different. Data mining applications usually involve a large number of time sequences. It is too difficult to visually compare similarity between time sequences. A way to solve this problem is to fit a model to each time sequence and measure the similarity of the models to decide the similarity of the time sequences.

The study on time sequence indexing has been focused on Fourier transform technique. Agrawal, Faloutsos, and Swami [1] proposed the first indexing method to use discrete Fourier transform (DFT, which is a distance-preserving transform) to map time sequences to the frequency domain and use the magnitudes of the first few frequencies to build indices in time sequence databases. The use of DFT assumes that the data sequences are periodic which is not always true in practice. Also, their method can only handle "whole matching" (i.e. all sequences and query sequence must have the same length) but not "subsequence matching". The amplitude scaling and offset translation problem was treated in a later paper by Agrawal, Lin, Sawhney, and Shim [2]. This work was particularly relevant to the fast

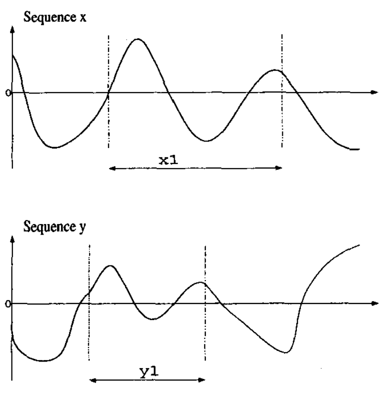

Figure 1: Two sequences **x** and **y**.

Fourier transform approach.

A similar work was reported by Rafiei and Mendelzon in [5]. They studied a set of linear transformations on the Fourier series representation of a sequence that can be used as the basis for similarity queries on time sequences data. The transformations were used so that different types of time series similarity could be searched for, depending on the needs of the user. These transformations could perhaps be used to find a better means of comparing similarity between time sequences with the fast Fourier transform approach.

Faloutsos, Ranganathan and Manolopoulos [3] modified the method in [1] to handle subsequence matching. They used a fixed size sliding window on time sequence to extract Fourier features. The main reason of using a fixed window is that the interpretation of DFT coefficients for different lengths of data sequences is different. Therefore, the DFT coefficients cannot be compared unless the lengths of data sequences are equal.

Ng and Huang [4] used the fast Fourier transform approach to classify time sequences of star light in an astronomical data mining project. The data preprocessing steps include removing noise, interpolating missing values, and aligning the measurements to the same time stamps. The star light time sequences were classified into two general classes: periodic and non-periodic. The periodic time sequences were analyzed in the frequency domain, whereas the non-periodic time sequences were analyzed in the time domain. The periodic time sequences were analyzed by calculating the discrete Fourier transform for each time sequence, using the fast Fourier transform (FFT) algorithm. The time sequences were then partitioned into subsets using a stepwise partitional clustering method based on the k-means algorithm. The outcome was a tree of star clusters from which some special star light curves can be identified through visualization.

Till now, most of the previous work on indexing of time sequences has focused on the Fourier transform technique. This technique favors sinusoidal type time sequences. In many application domains, time sequences are generated by stochastic processes. As such, classical time series analysis techniques are more suitable. However, there is little work reported. There are a few advantages of using time series modelling techniques. It produces short indices. The algorithms are fast to build models. More importantly, the models provide prediction capability which is crucial in many business applications.

This paper is organized as follows. Section 2 gives mathematical preliminary of time series modelling. Section 3 presents techniques and corresponding algorithms for building non-adaptive and adaptive models for time sequence indices and prediction. In Section 4, we present experimental results on time sequence clustering and prediction. We draw some concluding remarks on our current work in Section 5.

2 PRELIMINARIES OF TIME SERIES MODELLING

This section briefly reviews some of the classical time series analysis techniques for representing and modelling time sequences. We begin our discussion on classical time series analysis by giving some definitions.

Definition 1 *(Stochastic Process) A stochastic process is a family of time indexed random variables $x(\omega, t)$, where ω belongs to a sample space and t belongs to an index set.*

For a fixed t, $x(\omega, t)$ is a random variable. For a given ω, $x(\omega, t)$, as a function of t, is called a *realization* or sample function. Hence a time sequence is a *realization* from a certain stochastic process. We usually denote the random variable at time t by $x(t)$ if the time points in the index set are continuous, and by x_t if they are discrete. Moreover, the process is called a *real-valued process* if it assumes only real values. In our discussion, we consider the *discrete real-valued process* only. Also, we assume that the index set is $\mathbb{Z} = \{0, \pm 1, \pm 2, \cdots\}$ unless mentioned otherwise.

When dealing with a finite number of random variables, it is often useful to compute the covariance matrix in order to gain insight into the dependence between them. For a time sequence, we need to extend the concept of covariance matrix to deal with infinite collections of random variables. The autocovariance function provides us with the required use.

Definition 2 *(The Autocovariance Function) If $\{x_t\}$ is a process such that $Var(x_t) < \infty$ for each t, then the autocovariance function $\gamma_x(\cdot, \cdot)$ of $\{x_t\}$ is defined by*

$$\gamma_x(r, s) = \operatorname{Cov}(x_r, x_s) = \mathcal{E}[(x_r - \mathcal{E}(x_r))(x_s - \mathcal{E}(x_s))],$$

for r and s. Here $\mathcal{E}$ is the expectation operator.

An important class of stochastic processes are *stationary processes.*

Definition 3 *(Stationary Processes) The process $\{x_t, t \in \mathbb{Z}\}$ with index set $\mathbb{Z} = \{0, \pm 1, \pm 2, \cdots\}$ is said to be stationary if*

$$(i) \quad \mathcal{E}|x_t|^2 < \infty, \quad (ii) \quad \mathcal{E}[x_t] = \alpha, \quad \forall t \in \mathbb{Z},$$

$$(iii) \quad \gamma_x(r, s) = \gamma_x(r + t, s + t), \quad \forall r, s, t \in \mathbb{Z}.$$

If $\{x_t\}$ is stationary, then $\gamma_x(r,s) = \gamma_x(r-s,0)$ for all $r, s \in \mathbb{Z}$. Therefore, we can have the autocovariance function of a stationary process as the function of just one variable,

$$\gamma_x(r) \equiv \gamma_x(r,0) = \operatorname{Cov}(x_{t+r}, x_t) \quad \forall t, r \in \mathbb{Z}.$$

The function $\gamma_x(\cdot)$ is called the **autocovariance function** of $\{x_t\}$ and $\gamma_x(r)$ as its value at lag r. The **autocorrelation function** of $\{x_t\}$ is defined analogously as the function whose value at lag r is

$$\rho_x(r) \equiv \frac{\gamma_x(r)}{\gamma_x(0)} = \operatorname{Corr}(x_{t+r}, x_t), \quad \forall t, r \in \mathbb{Z}.$$

If $\{x_i\}$ is a real-valued stationary process, then from a second-order point of view it is characterized by its mean μ and its autocovariance function $\gamma_x(\cdot)$. The estimations of μ and $\gamma_x(\cdot)$ play an important role in the problem constructing an appropriate model for the data.

A natural unbiased estimator of the mean μ of the stationary process $\{x_i\}$ is the sample mean

$$\bar{\mathrm{x}} = \frac{1}{n}(x_1 + x_2 + \cdots + x_n) \tag{1}$$

if we only have n data points in the sequence. The estimators for $\gamma_x(r)$ are

$$\hat{\gamma}_x(r) = \frac{1}{n}\sum_{s=1}^{n-r}(x_s - \bar{\mathrm{x}})(x_{s+r} - \bar{\mathrm{x}}), \quad 0 \le r \le n-1, \tag{2}$$

and therefore the estimators for $\rho_x(r)$ are

$$\hat{\rho}_x(r) = \frac{\hat{\gamma}_x(r)}{\hat{\gamma}_x(0)}.$$

In time series analysis, there are two useful representations to express a time series process. One useful form is to write a process x_t such that its present value depends on the immediate past values together with a random error.

Definition 4 *(The Autoregressive (AR) Processes) The process $\{x_t, t \in \mathbb{Z}\}$ is said to be an AR(p) process if $\{x_t\}$ is stationary and if for every t,*

$$x_t - a_1 x_{t-1} - \cdots - a_p x_{t-p} = z_t, \tag{3}$$

where $\{z_t\}$ is white noise with mean zero and variance σ^2. The process $\{x_t\}$ is an AR(p) process with mean μ if $\{x_t - \mu\}$ is an AR(p) process.

The autoregression (AR) models have some nice properties. One of those properties is that AR models are invariant with respect to amplitude scaling and vertical shift.

Theorem 1 *Assume $\{x_t\}$ is an AR(p) process with mean μ. Let $a_1, a_2, \cdots, a_p$ be its AR coefficients. Then $\{cx_t - \nu\}$ is also an AR(p) process with mean $(c\mu - \nu)$ and with the same set of AR coefficients.*

Another nice property is that the autocorrelation function is also invariant with respect to amplitude scaling and vertical shift up.

Theorem 2 *Assume $\{x_t\}$ is a stationary process with mean μ. Then $\{cx_t - \nu\}$ is also a stationary process with mean $(c\mu - \nu)$ and its autocovariance and autocorrelation functions are given by*

$$c^2\gamma_x(r) \quad \text{and} \quad \rho_x(r) \quad \forall r = 0, 1, 2, \cdots,$$

respectively.

3 AUTOCOVARIANCE FUNCTION INDEXING

The autocovariance function values are important information to the construction of an appropriate model for the data. In this research, we use the autocovariance function as the extracted features from time sequences. The reason is that, if two sequences have similar shapes, then the dependencies between the data (separated by same time lag) will be similar. And hence they have similar autocovariance function values. However, this idea is not sufficient to support that the autocovariance function indexing method is sufficiently good for extracting features from time sequences.

Therefore, now we need to show this indexing method guarantees no "false dismissals". i.e. if we use this method to extract the features from time sequences and the query sequences in similarity queries, it should return all the qualifying sequences without missing any. It has been proven that if the distance in the feature space satisfies

$$\mathbf{d}_{original\ space}(\mathbf{x}, \mathbf{y}) \ge \mathbf{d}_{index\ space}(\mathbf{x}', \mathbf{y}') \tag{4}$$

where **x'** and **y'** are the representations of **x** and **y** in the index space respectively, then the method can guarantee no "false dismissals" [3]. Let's see the following theorems.

Theorem 3 *Let $\mathbf{x} = (x_1, x_2, \cdots, x_n)$ and $\mathbf{y} = (y_1, y_2, \cdots, y_n)$ be two data sequences of zero mean and 2-norm being equal to 1, i.e. $\|\mathbf{x}\|_2 = \|\mathbf{y}\|_2 = \sqrt{n}$. Then*

$$\left\| \frac{\hat{\gamma}_x(0:k)}{2} - \frac{\hat{\gamma}_y(0:k)}{2} \right\|_2 \le \|\mathbf{x} - \mathbf{y}\|_2, \quad \forall k \ge 1,$$

where

$$\hat{\gamma}_x(0:k) = [\hat{\gamma}_x(0), \hat{\gamma}_x(1), \cdots, \hat{\gamma}_x(k)]^t$$

and $\hat{\gamma}_y(0:k)$ is defined similarly.

In the above theorem, the data sequences x and y must be of exactly the same length. The following theorem generalizes it by considering data sequences of similar lengths.

Theorem 4 *Let $\mathbf{x} = (x_1, x_2, \cdots, x_n)$ and $\mathbf{y} = (y_1, y_2, \cdots, y_m)$ be two data sequences of zero mean and 2-norm being equal to 1. Here $m \approx n$ and we assume $m \ge n$. Let $\mathbf{v}_i = (y_i, \cdots, y_{i+n-1})$ for $1 \le i \le m - n + 1$. So $length(\mathbf{v}_i) = n$. Then $\forall k \ge 1$,*

$$\left\| \frac{\hat{\gamma}_x(0:k)}{2} - \frac{\hat{\gamma}_y(0:k)}{2} \right\|_2 \le \max_{1 \le i \le m-n+1} \|\mathbf{x} - \mathbf{v}_i\|_2.$$

According to the above two theorems, the autocovariance function can be used to index time sequences (their lengths are not required to be the same, but have to be similar). In particular, we consider the following non-adaptive statistical model that uses the autocovariance function to index time sequences.

3.1 Non-adaptive Statistical Model

We start to index the time sequence $\mathbf{x} = (x_1, \cdots, x_n)$. Firstly we rescale it so that it is of zero mean and 2-norm being equal to 1. The resulting rescaled time sequence is $\mathbf{s} = (s_1, \cdots, s_n)$, say. Then we fit AR models from the first order to higher orders (order less than or equal to L which is the maximum AR order) for $\mathbf{s}$ until the decreasing rate of the modelling error is less than a specified tolerance ϵ. Correspondingly, the autocovariance function can be computed up to the order of this AR model. We repeat the above process for other time sequences. The process is formalized in the following procedure:

- (Step 1): Rescaling

  Specify ϵ and L. Rescale $\mathbf{x} = (x_1, \cdots, x_n)$ to $\mathbf{s} = (s_1, \cdots, s_n)$ by $s_i = \sqrt{\frac{n}{N}}(x_i - \bar{\mathbf{x}})$ where $N = \sum_{i=1}^{n}(x_i - \bar{\mathbf{x}})^2$ and $\bar{\mathbf{x}} = \frac{1}{n}\sum_{i=1}^{n} x_i$. ($\bar{\mathbf{s}} = 0$, $\|\mathbf{s}\|_2 = \sqrt{n}$, $\hat{\gamma}_s(0) = 1$.) Set $p = 1$.

- (Step 2): AR model fitting

  Calculate $\hat{\gamma}_s(p) = \frac{1}{n}\sum_{t=1}^{n-p} s_t s_{t+p}$ and compute the estimators of AR(p) coefficients $[\hat{a}_1, \cdots, \hat{a}_p]$.

- (Step 3): Termination

  Calculate the modelling error err_p which is defined as $err_p = \sigma^2 = \hat{\gamma}_s(0) - (\hat{a}_1, \hat{a}_2, \cdots, \hat{a}_p)[\hat{\gamma}_s(1), \cdots, \hat{\gamma}_s(p)]^t$

  If $p = 1$, set $p = p + 1$ and go to step 2.

  If $\frac{err_{p-1} - err_p}{err_{p-1}} > \epsilon$ and $p \leq L$, set $p = p + 1$ and go to step 2.

  If $\frac{err_{p-1} - err_p}{err_{p-1}} > \epsilon$ but $p > L$, terminate and return a warning message. Otherwise terminate by storing $\hat{\gamma}_s(0 : p-1) = [1, \hat{\gamma}_s(1), \cdots, \hat{\gamma}_s(p-1)]^t$ as the extracting features from $\mathbf{s}$ (or $\mathbf{x}$) together with the time sequence's identity number.

For each time sequence, we build an AR model and employ the autocovariance function to index these data points. We only need to store the autocovariance function and the identify number of that time sequence. We note that we can derive the autoregression model from the autocovariance function. Data mining operations can be applied efficiently by using these stored values.

3.2 Adaptive Statistical Models

We have already introduced the non-adaptive statistical model which is accurate (guarantees no "false dismissals") and efficient. In this subsection, we will introduce the adaptive statistical model which is modified from the non-adaptive model. But why do we need to consider the modified model for features extraction? The following example gives us the motivation.

Figures 2 and 3 show two time sequences $\mathbf{x}$ and $\mathbf{y}$. The features of $\mathbf{x}$ and $\mathbf{y}$ extracted by the non-adaptive statistical model are $\hat{\gamma}_x(0 : 4) = [1, 0.2616, 0.2499, 0.3167, -0.0960]^t$ and $\hat{\gamma}_y(0 : 4) =$

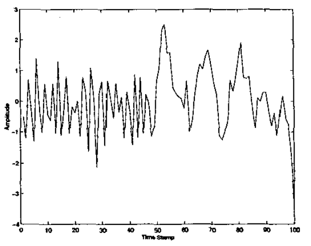

Figure 2: Time sequence $\mathbf{x}$.

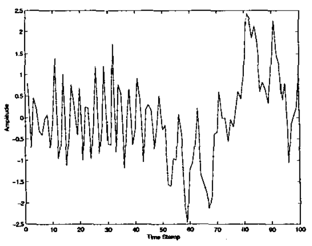

Figure 3: Time sequence $\mathbf{y}$.

$[1, 0.4940, 0.3715, 0.5206, 0.2511]^t$ respectively. (Note that the order of the "best fitted" AR models of the time sequences are not necessary the same. In this example, we specified $\epsilon = 0.025$.) Since $\|\hat{\gamma}_x(0 : 4) - \hat{\gamma}_y(0 : 4)\|_2 = 0.4805$, the time sequences $\mathbf{x}$ and $\mathbf{y}$ are not considered to be similar. But indeed, both $\mathbf{x}[1 : 50] = (x_1, \cdots, x_{50})$ and $\mathbf{y}[1 : 50] = (y_1, \cdots, y_{50})$ are generated by AR(2) model with AR coefficients $a_1 = -0.8$ and $a_2 = -0.5$. And in Figures 4 and 5, we find that these two subsequences are similar.

From this example, we find that it's a good idea to extract features from *subsequences* by the adaptive statistical model since we can notice the change of model in the whole time sequence. The idea of adaptive statistical model is easy to be understood and it's similar to the idea of non-adaptive model.

Let us consider the time sequence $\mathbf{y} = (y_1, \cdots, y_m)$. We assume that the minimum window size is w, where w is specified by the user and depends on the applications. We index the subsequence $\mathbf{y}[1 : w] = (y_1, \cdots, y_w)$ by the non-adaptive statistical model. i.e. we fit the rescaled subsequence by the AR model. Next we consider the subsequence $\mathbf{y}[1 : w + d] = (y_1, \cdots, y_{w+d})$ with a new set of data points $(y_{w+1}, \cdots, y_{w+d})$. Here d is the adjusted window size which is the number of data points added in each test. We index $\mathbf{y}[1 : w + d]$ by the non-adaptive model. If the order of AR model is the same and the distance between the features are not large (up to tolerance), then we can still use the AR model and the autocovariance function for the new data subsequence, and then we continue to add data point to test the AR model. Otherwise we restart the process again and index the subsequence $(y_{w+1}, \cdots, y_{2w})$ by the non-adaptive model. The process continues until all data points in the sequence are scanned. We repeat the above process for other time sequences. The process is formalized

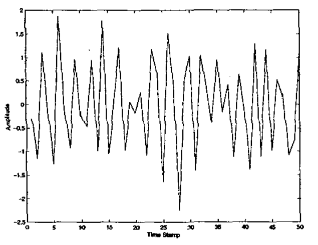

Figure 4: $\mathbf{x}[1:50]$ in Example 2.

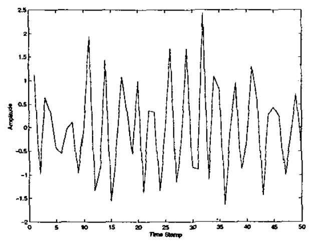

Figure 5: $\mathbf{y}[1:50]$ in Example 2.

in the following procedure:

- (Step 1): Setting Parameters

  Input the minimum window size w, the adjusted window size d, the specified tolerance ϵ_1 which is used in the AR fitting process, another specified tolerance ϵ_2 which is used in the model updating process, and the maximum AR order L.

  Set i=1, j=w. Go to Step 2.

- (Step 2): Model Fitting

  If $i \geq m$, quit. Otherwise, index $\mathbf{y}[i:j] = (y_i, \cdots, y_j)$ by the non-adaptive statistical model. (i.e. let $\mathbf{x} = \mathbf{y}[i:j]$, $\epsilon = \epsilon_1$ and execute the process stated in the non-adaptive procedure.)

  If there is a warning message (i.e. $p \geq L$ where p is the AR order), then set $i = i+1, j = min\{j+1, m\}$ and repeat Step 2 again. Else if $j = m$, store i, j, v_1 (which is the features of $\mathbf{x} = \mathbf{y}[i:j]$ obtained from the process) and the time sequence's identity number as a new row in the indexing table and quit. Otherwise, go to Step 3.

- (Step 3): Model Updating

  Set $j' = min\{j+d, m\}$. Index $\mathbf{y}[i, j']$ by the non-adaptive statistical model. (i.e. let $\mathbf{x} = \mathbf{y}[i:j']$, execute the process stated in the non-adaptive procedure and get v_2 which is the features of $\mathbf{x} = \mathbf{y}[i:j']$.)

  If the AR order does not change and $\|v_1 - v_2\|_2 \leq \epsilon_2$, then set $j = j'$ and $v_1 = v_2$. If $j = m$, store i, j, v_1 and the time sequence's identity number as a new row in the indexing table and quit. Otherwise (i.e. $j \neq m$), repeat Step 3 again.

Table 1: Synthetic Data Set

Batch No	M	$(A_1, \cdots, A_M)$	$(\theta_1, \cdots, \theta_M)$
1	2	(5, 1)	(0.2, 0.25)
2	3	(5, 1, 1)	(0.04, 0.1, 0.15)
3	4	(5, 1, 1, 1)	(0.5, 0.55, 0.7, 0.9)

Table 2: Classification accuracy, with varying noise level and frequency perturbation

AR/DFT Classification Accuracy				
Freq.	Noise n_A			
n_f	0.00	0.025	0.050	0.100
0.00	100/100	100/100	100/100	100/100
0.01	67/80	73/90	87/70	70/73
0.02	93/70	77/67	83/70	77/80
0.03	63/70	83/87	77/67	80/70
0.04	67/63	70/67	70/93	87/83
0.05	70/63	77/70	80/63	63/77

However, if the AR order changes or $\|v_1 - v_2\|_2 > \epsilon_2$, then store i, j, v_1 and the time sequence's identity number as a new row in the indexing table. Set $i = j+1, j = min\{i+w-1, m\}$ and go to Step 2.

4 EXPERIMENTAL RESULTS

In this section, some experimental results are shown to demonstrate the effectiveness of statistical models for data mining operations.

4.1 Statistical Models versus Fourier Transforms

In the first test, we compare the performance of statistical models and Fourier Transforms in clustering operations. The methodology consists of four steps. The first step is to generate several sets of time sequences of known classes. The second step is to calculate the autocovariance function values and Fourier coefficients for each time sequence. The third step is to use the autocovariance function value and the magnitude of Fourier coefficients as feature vectors for classification with a clustering algorithm. The final step is to compare the clustering results with the originally known classes, and calculate the classification accuracy. To understand how noise and frequency perturbation affect the classification, several different datasets with different noise levels and frequency perturbations were generated.

In real applications, many time sequences look like cosine curves (or sinusoidal curves). Therefore the synthetic data were modeled as the sum of a number of cosine curves with a noise function. Their formulation is

$$x_t = \sum_{i=1}^{M} A_i \cos(\tilde{\theta}_i t) + n_t \qquad (5)$$

where M is the number of cosine curves, each $\tilde{\theta}_i$ is the adjusted frequency component, A_i is the associated amplitude of each frequency component and n_t is a noise function. Given the frequency perturbation level n_f and the frequency

Table 3: Setting for adaptive and non-adaptive models.

Non-adaptive Model
Maximum AR order $L = 10$
Specified Tolerance for AR fitting process $\epsilon = 0.025$
Adaptive Model
Minimum window size $w = 20$
Adjusted window size $d = 1$
Maximum AR order $L = 10$
Specified Tolerance for AR fitting process $\epsilon_1 = 0.025$
Specified Tolerance for model updating process $\epsilon_2 = 0.1$

component θ_i, the adjusted frequency component is formulated by

$$\tilde{\theta}_i = \theta_i + n_f Z_i^u \pi \qquad (6)$$

where Z_i^u is a uniformly distributed random variable on the interval [-1,1]. And the noise function n_t is found by

$$n_t = n_A Z_t \qquad (7)$$

where Z_t is a normally distributed random variable with mean $\mu = 0$ and standard deviation $\sigma = 1$. In this research, we generated three batches of ten time sequences. Each of them has length 200. And the time sequences in the same batch have the same A_i and θ_i. Table 1 shows the details of the batches of time sequences.

Here we set the maximum AR order L to be 10 and the specified tolerance ϵ for AR fitting process to be 0.025. In our experiment, we found that this maximum AR order is sufficiently large for all those synthetic data. And this setting of ϵ is supported by the experiment in [6, 7]. Table 2 shows the result of classification accuracy. It is clear to show that our method is in general more accurate than the Fourier transform. Moreover, for the Fourier transform, the magnitude of first seven Fourier coefficients were used as the feature vector. For statistical models, the average order of the best fitted AR model is about 3.60 only. Since the number of parameters required by our approach is less than that required by the Fourier transforms, our proposed method is more efficient.

4.2 Adaptive versus Non-adaptive Models

In this subsection, we compare adaptive models with non-adaptive models by using three Hong Kong stocks: Guangdong Investment Ltd. (called it *gdinvest*), Great Eagle Holdings Ltd. (called it *geh*) and Wheelock Co. Ltd. (called it *wheelock*). In Figures 6, 7 and 8, we show their stock values respectively. All of these time sequences are of length 750, which is approximately equal to 3 years trading days.

We applied the non-adaptive and adaptive models on these time sequences. Table 3 shows the setting of these models in our experiment. Note that for the adaptive models, we set the minimum window size w to be 20 which is about one month trading days. Therefore each adaptive model is built based on at least one month stock data. In this research, we used both the non-adaptive model and the last adaptive model to predict five data points $y_{751}, \cdots, y_{755}$. Since in our experiment the real values of these data points

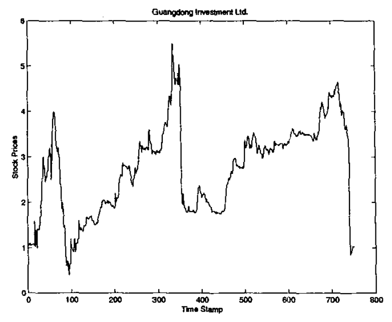

Figure 6: Stock prices of Guangdong Investment Ltd.

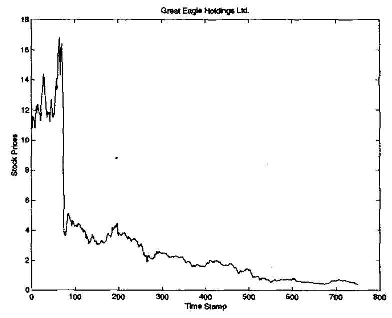

Figure 7: Stock prices of Great Eagle Holdings Ltd.

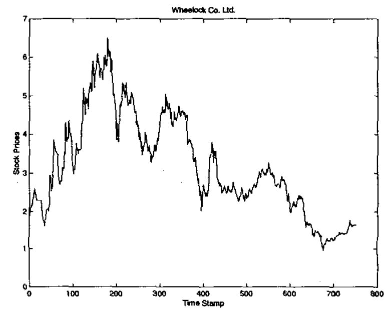

Figure 8: Stock prices of Wheelock Co. Ltd.

are known, we can easily calculate the prediction error using the following formula:

$$\frac{1}{n} \sum_{i=1}^{n} \left| \frac{\hat{y}_i - y_i}{y_i} \right|$$

where n is the number of predicted data points, y_i is the real value and $\hat{y}_i$ is the predicted value at the date i.

Table 4 shows the results of non-adaptive and adaptive models for these three stocks. We find that the stock values change quite a lot as there are a number of models constructed by the adaptive method. Here, let's use the stock data *gdinvest* as an example. By adaptive method, the subsequences $(y_{75}, \cdots, y_{354})$ and $(y_{355}, \cdots, y_{502})$ are fitted by two models. For the subsequence $(y_{75}, \cdots, y_{354})$, the order of model is 1 and the AR coefficient is 0.99. And for

Table 4: Results for adaptive and non-adaptive models.

	gdinvest	geh	wheelock
Non-adaptive			
Order	1	2	1
AR Coefficients	0.9840	(1.1659, -0.1765)	0.9924
Autocovariance Function	0.9840	(0.9910, 0.9790)	0.9924
Prediction Error	10.3%	30.4%	11.1%
Adaptive			
No. of models	14	10	18
Order of last model	4	1	1
AR Coefficients	(0.7200,0.2020, -0.2007 -0.2259)	0.9895	0.9918
Autocovariance Function	(0.7453,0.4805, 0.1274,-0.1867)	0.9895	0.9918
Prediction Error	5.5%	15.5%	9.6%

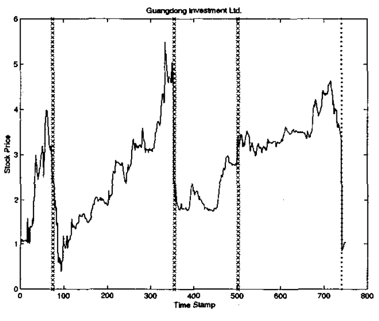

Figure 9: Stock prices of Guangdong Investment Ltd.

the subsequence $(y_{355}, \cdots, y_{502})$, the order of model is 1 and the AR coefficient is 0.96. It seems that their models are almost the same. So why the adaptive method partitions $(y_{75}, \cdots, y_{502})$ into two subsequences? In Figure 9, the subsequences between the crossed lines are the subsequences $(y_{75}, \cdots, y_{354})$ and $(y_{355}, \cdots, y_{502})$. Note that at time stamp 354, it's a great change of value. Therefore, the autocorrelation between these data (include the new data) is no longer the same (or similar) as before. We can conclude that the adaptive model can figure out the great change of value of time sequence. By using adaptive method, the last subsequence is $(y_{741}, \cdots, y_{750})$, which is the subsequence on the right hand side of the dotted line in Figure 9. It is easy to note that there is a great change at time stamp 741. Moreover, we can find that using the last adaptive model to predict values is more accurate than using the non-adaptive model.

From Table 4, we find that the last adaptive models of the stocks *gdinvest* and *geh* are different from their non-adaptive models. The last adaptive model for *wheelock* is quite similar to its non-adaptive model. However, the autoregression coefficients and the autocovariance function values are different. The results show that the prediction error of the adaptive model is less than that of the non-adaptive model. Table 5 shows the details of these errors. $\hat{y}_i^n$ and $\hat{y}_i^a$ denote the predicted values by non-adaptive and adaptive models respectively. It's easy to find that for both non-adaptive and adaptive models, the absolute relative errors of the first few predicted points are much less than that of the last few ones. The reason is that, the first few data points are calculated by the model using the real values, but the last few data points are calculated using the previous predicted values. For example, for *geh*, the order of last adaptive model is 1. So we used the real value y_{750} and the model to predict $\hat{y}_{751}^a$. Then we used the previous predicted value $\hat{y}_{751}^a$ to predict $\hat{y}_{752}^a$. Hence the errors are accumulated. Figures 10, 11 and 12 show the predicted values by non-adaptive model (i.e. $\hat{y}_{751}^n, \cdots, \hat{y}_{755}^n$), the predicted values by adaptive model (i.e. $\hat{y}_{751}^a, \cdots, \hat{y}_{755}^a$) and the real values (i.e. $y_{751}, \cdots, y_{755}$) of these three stocks. The symbols "circle", "square" and "cross" represent the real value, the predicted value by non-adaptive method and the predicted value by adaptive method respectively. We find that the distance between the "cross" and "circle" is always less than that between the "square" and "circle". Therefore we can conclude that it's still practicable to apply the adaptive model on the data to predict a few data points.

Table 5: Details of the predicted values by non-adaptive model and adaptive model.

y_i	$\hat{y}_i^n$	$\hat{y}_i^a$	$\left\vert\frac{\hat{y}_i^n - y_i}{y_i}\right\vert$	$\left\vert\frac{\hat{y}_i^a - y_i}{y_i}\right\vert$
gdinvest				
1.0200	1.0572	0.9939	0.0365	0.0255
1.0000	1.0839	0.9660	0.0839	0.0340
1.0000	1.1102	0.9405	0.1102	0.0595
1.0000	1.1361	0.9261	0.1361	0.0739
1.0100	1.1616	0.9243	0.1501	0.0848
geh				
0.3800	0.4197	0.4059	0.1045	0.0682
0.3300	0.4511	0.4167	0.3670	0.2628
0.3700	0.4834	0.4274	0.3065	0.1551
0.3750	0.5155	0.4380	0.3747	0.1679
0.4000	0.5473	0.4484	0.3681	0.1211
wheelock				
1.6000	1.6518	1.6443	0.0324	0.0277
1.4700	1.6635	1.6486	0.1316	0.1215
1.5400	1.6751	1.6528	0.0877	0.0732
1.4500	1.6866	1.6570	0.1632	0.1428
1.4900	1.6980	1.6612	0.1396	0.1149

5 CONCLUDING REMARKS

In this paper, we have presented statistical modelling techniques for time sequence indexing, clustering and prediction. Unlike statistical study on fine tuning models to obtain more accurate prediction, our interests are to apply these modelling techniques to large number of time se-

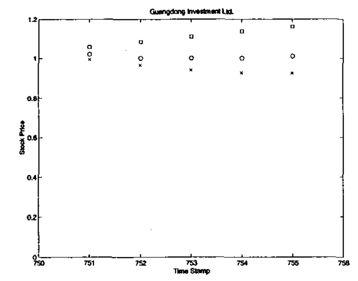

Figure 10: Real and predicted stock prices of Guangdong Investment Ltd.

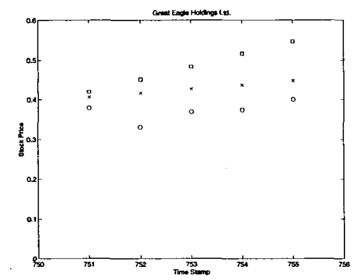

Figure 11: Real and predicted stock prices of Great Eagle Holdings Ltd.

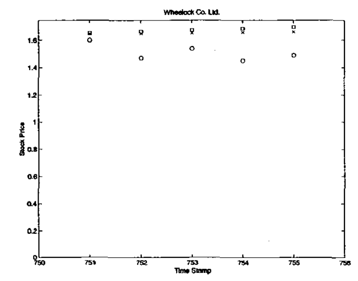

Figure 12: Real and predicted stock prices of Wheelock Co. Ltd.

quences. We have demonstrated that the modelling techniques can play double roles for indexing and prediction. The major advantages are (i) the computational efficiency of calculating the autocovariance functions and AR models, which are capable to handle very large data volume, (ii) short indices, and (iii) prediction capability.

Our future work is to study a classification scheme of AR models built from large number of time sequences using our adaptive modelling technique. With the classification scheme, we will be able to transform a time sequences to a list of AR models in different classes. Then, we can apply an association rule algorithm to discover association rules of AR models. If we consider each model represents a behavior of an object in a particular time period, we will be able to investigate how behaviors change and how different behaviors are related overtime.

ACKNOWLEDGMENT

M. Ng's research is supported in part by RGC Grant Nos. 7132/00P and 7130/02P, and HKU CRCG Grant Nos. 10203501, 10203907 and 10203408.

REFERENCES

[1] R. Agrawal, C. Faloutsos, and A. Swami, *Efficient Similarity Search in Sequence Databases*, Foundations of Data Organizations and Algorithms (FODO) Conference, 1993.

[2] R. Agrawal, K.-I. Lin, H. S. Sawhney, and K. Shim, *Fast Similarity Search in the Presence of Noise, Scaling, and Translation in Time-series Databases*, Proceedings of the 21st VLDB Conference, 1995.

[3] C. Faloutsos, M. Ranganathan and Y. Manolopoulos, *Fast subsequence matching in time-series databases*, Proceedings of ACM SIGMOD Conference, 1994.

[4] M. K. Ng, and J. Z. Huang, *Data-mining massive time series astronomical data: challenges, problems and solutions*, Information and Software Technology 41 (1999) 545-556.

[5] D. Rafiei, and A. Mendelzon, *Similarity-based Queries for Time Series Data*, Proceedings of ACM SIGMOD Conference, ACM, 1997.

[6] J. K. Ting, *Time Sequences Data Mining*, MPhil thesis, The University of Hong Kong, 2001.

[7] K. H. Skinner, *Temporal Data Mining Techniques for Classification of Time Series Data*, BSc thesis, Australian National University, 1997.