Mixtures of ARMA Models for Model-Based Time Series Clustering

Yimin Xiong   Dit-Yan Yeung

Department of Computer Science
Hong Kong University of Science and Technology
Clear Water Bay, Kowloon
Hong Kong

September 11, 2002

Abstract

Clustering problems are central to many knowledge discovery and data mining tasks. However, most existing clustering methods can only work with fixed-dimensional representations of data patterns. In this paper, we study the clustering of data patterns that are represented as sequences or time series possibly of different lengths. We propose a model-based approach to this problem using mixtures of *autoregressive moving average* (ARMA) models. We derive an *expectation-maximization* (EM) algorithm for learning the mixing coefficients as well as the parameters of the component models. The algorithm can determine the number of clusters in the data automatically. Experiments were conducted on a number of simulated and real datasets. Results from the experiments show that our method compares favorably with another method recently proposed by others for similar time series clustering problems.

Keywords: ARMA model, EM algorithm, mixture model, model-based clustering, time series analysis

1 Introduction

Clustering is the unsupervised process of grouping data patterns into clusters so that patterns within a cluster bear strong similarity to one another but are very dissimilar to patterns in other clusters. Clustering problems are central to many knowledge discovery and data mining tasks. While many clustering techniques have been studied by researchers in statistics, pattern recognition, and machine learning, most of these techniques are based on the assumption that the data patterns can be represented as points in multidimensional spaces of fixed dimensionality. Unfortunately this assumption does not always hold. Temporal patterns involving sequences or time series are one important class of such problems that are found in many applications from scientific, medical, sociological, financial and other domains.

Distance-based methods (e.g., [1]) and *model-based* methods (e.g., [2]) are two major classes of clustering methods. They are analogous to other nonparametric and parametric methods, respectively, in that the former category (i.e., distance-based or nonparametric methods) assumes only some weak structure of the data, but the latter category (i.e., model-based or parametric methods) assumes some strong structure. *Partitional* distance-based methods, such as the well-known k-means algorithm, usually require the number of clusters to be known *a priori*. *Hierarchical* distance-based methods, on the other hand, do not require the number of clusters to be known. However, they have to search exhaustively for the number of clusters, either by starting from one (for *divisive* methods) or from the total number of all data patterns available (for *agglomerative* methods). Moreover, since typically pairwise distances have to be computed, hierarchical distance-based methods tend to have computational complexity that is quadratic in the number of data patterns and hence becomes prohibitive for large data sets.

Unlike distance-based methods, however, model-based methods can incorporate prior knowledge more naturally in finding the correct number of clusters. Also, for time series data, they provide a principled approach for handling the problem of modeling and clustering time series of different lengths. In this paper, we will focus on model-based time series clustering methods. In particular, *mixture models* [2] will be used.

The remainder of this paper is organized as follows. Some related work on model-based clustering of time series will be reviewed in Section 2. In Section 3, we will present our model-based time series clustering method based on mixtures of time series models. The learning algorithm will be outlined in Section 4, with details left to Appendix A. Experimental results for simulated and real datasets will be presented in Sections 5 and 6, respectively. Finally, Section 7 will conclude the paper with discussions of some possible future work.

2 Related Work

2.1 Markov Chains

Finite mixtures of Markov chains [3, 4, 5, 6, 7, 8] have been proposed for clustering time series. The *expectation-maximization* (EM) algorithm [9] is used to learn the mixing coefficients as well as the parameters of the component models. The number of clusters can be determined by comparing different choices of the number based on some scoring scheme. One possibility, used by Cadez *et al.* [7, 8], is related to minimizing the description length.

Another approach to the clustering of time series modeled by Markov chains is called Bayesian clustering by dynamics (BCD) [10, 11, 12]. Strictly speaking, this method is not a purely model-based approach. Rather, it can best be seen as a hybrid approach with both model-based and distance-based flavors. The BCD method first transforms each time series into a Markov chain, with its dynamics represented simply by a transition probability matrix. Next, it goes through an agglomerative procedure by trying to merge the two closest Markov chains at each step, using the Kullback-Leibler divergence [13] as the dissimilarity (cf. distance) measure between transition probability matrices. Based on a greedy heuristic search approach, this procedure continues until the resulting model is found to be less probable than the model before merging. Thus the number of clusters can be determined automatically.

2.2 Hidden Markov Models

While simple Markov chains are good enough for some applications, some time series can be modeled better using *hidden Markov models* (HMM) [14] due to their ability of handling uncertainty in temporal and spatial dimensions simultaneously. For example, HMMs have been very successfully used for speech recognition, handwriting recognition, and bioinformatics. The idea of HMM-based clustering was first studied by Rabiner *et al.* [15] for speech recognition applications. Multiple HMMs are created through a splitting procedure to group speech data into clusters for more accurate modeling. The HMM-based clustering method studied in [16] follows the same spirit in that HMMs are introduced one at a time to model the time series data more accurately. It also shows that this method can perform better if dynamic time warping (DTW) is used as a distance metric to form an initial partitioning of the time series for the subsequent HMM-based clustering procedure.

Finite mixtures of HMMs have also been studied by a number of researchers. Similar to mixtures of Markov chains, the EM algorithm can also be used for HMM mixtures [17, 18, 19, 20]. To trade accuracy for efficiency, the k-means algorithm (used in [21]) and the rival penalized competitive learning (RPCL) algorithm (used in [22]) have also been used in place of EM. The number of clusters can be determined using Monte-Carlo cross-validation [19] or information criteria such as the Bayesian information criterion (BIC) [20].

2.3 Autoregressive Moving Average Models

In addition to Markov chains and HMMs, *autoregressive moving average* (ARMA) and *autoregressive integrated moving average* (ARIMA) models have also been used extensively for time series analysis [23, 24]. Kwok *et al.* [25] applied mixtures of ARMA models as well as their special cases, mixtures of autoregressive (AR) models, for time series modeling and forecasting. However, clustering applications based on such mixture models were not studied by them.

More recently, a method was proposed by Kalpakis *et al.* for clustering ARIMA time series [26]. This method is similar to the BCD method for Markov chains in that it is a hybrid method with both model-based and distance-based characteristics. For each time series in the data set, a differencing operation is applied to remove the nonstationarity in the time series and a separate ARMA model is created by estimating the model parameters from the time series after nonstationarity removal. Afterwards, a partitional distance-based clustering method called the partitioning around medoids (PAM) method is applied to group the ARMA models into a prespecified number of clusters. The distance measure used is the Euclidean distance between the linear predictive coding (LPC) cepstral coefficients computed from two ARMA models.

In the next section, we will propose a new time series clustering method based on mixtures of ARMA models.

3 Model-Based Clustering with ARMA Mixtures

3.1 Standard ARMA Models

The ARIMA model introduced by Box and Jenkins [23] is a combination of three types of time series data processes, namely, autoregressive, integrated, and moving average processes. A stationary ARIMA model with autoregressive order p and moving average order q is commonly denoted as ARMA(p, q). If the ARMA(p, q) model is used on time series data integrated to order d, then the model for the original time series is denoted as ARIMA(p, d, q). Given a time series $\mathbf{x} = \{x_t\}_{t=1}^n$, the fitted ARMA$(p, q)$ model takes the form

$$x_t = \phi_0 + \sum_{j=1}^{p} \phi_j x_{t-j} + \sum_{j=1}^{q} \theta_j e_{t-j} + e_t, \qquad t = 1, 2, \ldots, n,$$

where n is the length of the time series, ϕ_0 is a constant term, $\{\phi_1, \phi_2, \ldots, \phi_p, \theta_1, \theta_2, \ldots, \theta_q\}$ is the set of AR(p) and MA(q) coefficients, and $\{e_t\}_{t=1}^n$ is a sequence of independent and identically distributed (IID) Gaussian white noise terms with variance σ^2. From [24], we can express the natural logarithm of the conditional likelihood function as

$$\ln P(\mathbf{x}|\mathbf{\Phi}) = -\frac{n}{2}\ln(2\pi) - \frac{n}{2}\ln(\sigma^2) - \frac{1}{2\sigma^2}\sum_{t=1}^{n} e_t^2,$$

where $\mathbf{\Phi} = \{\phi_0, \phi_1, \phi_2, \ldots, \phi_p, \theta_1, \theta_2, \ldots, \theta_q, \sigma^2\}$ is the set of all model parameters and e_t must be estimated recursively, i.e.,

$$e_t = x_t - \phi_0 - \sum_{j=1}^{p} \phi_j x_{t-j} - \sum_{j=1}^{q} \theta_j e_{t-j}, \qquad t = 1, 2, \ldots, n.$$

3.2 ARMA Mixtures

We now extend standard ARMA models to mixtures of ARMA models, or simply called ARMA mixtures, for time series clustering. Let us assume that the time series data are generated by M different ARMA models, which correspond to the M clusters of interest denoted as $\omega_1, \omega_2, \ldots, \omega_M$. Let $P(\mathbf{x}|\omega_k, \mathbf{\Phi}_k)$ denote the conditional likelihood function or density function of component model k, with $\mathbf{\Phi}_k$ being the set of parameters for the model. Let $P(\omega_k)$ be the prior probability that a time series comes from model k. The conditional likelihood function of the mixture model can be expressed in the form of a mixture density as

$$P(\mathbf{x}|\mathbf{\Theta}) = \sum_{k=1}^{M} P(\mathbf{x}|\omega_k, \mathbf{\Phi}_k) P(\omega_k),$$

where $\mathbf{\Theta} = \big\{\mathbf{\Phi}_1, \mathbf{\Phi}_2, \ldots, \mathbf{\Phi}_M, P(\omega_1), P(\omega_2), \ldots, P(\omega_M)\big\}$ represents the set of all model parameters for the mixture model. For a time series $\mathbf{x}$, it is assigned to cluster ω_k with posterior probability $P(\omega_k|\mathbf{x})$, where $\sum_{k=1}^{M} P(\omega_k|\mathbf{x}) = 1$.

Suppose we are given a set $\mathbf{D} = \big\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\big\}$ of N time series. Under the usual assumption that different time series are conditionally independent given the underlying model parameters, we can express the likelihood of $\mathbf{D}$ as

$$P(\mathbf{D}|\mathbf{\Theta}) = \prod_{i=1}^{N} P(\mathbf{x}_i|\mathbf{\Theta}) \qquad (1)$$

Model parameter learning amounts to finding the *maximum a posteriori* (MAP) parameter estimate given the data set $\mathbf{D}$, i.e.,

$$\widehat{\mathbf{\Theta}} = \arg\max_{\mathbf{\Theta}} \big[P(\mathbf{D}|\mathbf{\Theta}) P(\mathbf{\Theta})\big].$$

If we take a noninformative prior on $\mathbf{\Theta}$, learning degenerates to *maximum likelihood estimation* (MLE), i.e.,

$$\widehat{\mathbf{\Theta}} = \arg\max_{\mathbf{\Theta}} P(\mathbf{D}|\mathbf{\Theta}).$$

This MLE problem can be solved efficiently using the EM algorithm, which will be discussed in detail in the next section.

4 EM Learning Algorithm

4.1 Standard EM Algorithm

The EM algorithm is an iterative approach to MLE or MAP estimation problems in the presence of incomplete data. It has been widely used for many applications, including clustering and mixture density estimation problems [27].

Let us rewrite the likelihood in Equation (1) as a function of the parameter vector $\mathbf{\Theta}$ for a given data set $\mathbf{D}$:

$$L(\mathbf{\Theta}; \mathbf{D}) = P(\mathbf{D}|\mathbf{\Theta}) = \prod_{i=1}^{N} P(\mathbf{x}_i|\mathbf{\Theta}).$$

Assuming a noninformative prior on $\mathbf{\Theta}$, the goal of the EM algorithm is to find $\mathbf{\Theta}$ that maximizes the likelihood $L(\mathbf{\Theta}; \mathbf{D})$ or the log-likelihood

$$\ell(\mathbf{\Theta}; \mathbf{D}) = \sum_{i=1}^{N} \ln P(\mathbf{x}_i|\mathbf{\Theta}) = \sum_{i=1}^{N} \ln \left[\sum_{k=1}^{M} P(\mathbf{x}_i|\omega_k, \mathbf{\Phi}_k) P(\omega_k) \right].$$

Since $\mathbf{D}$ is the incomplete data, we assume the missing data to be $\mathbf{Z} = \{\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_N\}$, such that $\mathbf{D}$ and $\mathbf{Z}$ form the complete data $(\mathbf{D}, \mathbf{Z})$. Thus the complete-data log-likelihood function is $\ln P(\mathbf{D}, \mathbf{Z}|\mathbf{\Theta})$. If we knew the missing data (and hence the complete data), parameter estimation would be straightforward. Without knowing the missing data, however, the EM algorithm has to iterate between the Expectation step (E-step) and the Maximization step (M-step). In the E-step, we calculate the expected value $Q\big(\mathbf{\Theta}|\mathbf{\Theta}(t)\big)$ of the complete-data log-likelihood with respect to the unknown data $\mathbf{Z}$ given the observed data $\mathbf{D}$ and the current parameter estimate $\mathbf{\Theta}(t)$, i.e.,

$$Q\big(\mathbf{\Theta}|\mathbf{\Theta}(t)\big) = E\big[\ln P(\mathbf{D}, \mathbf{Z}|\mathbf{\Theta}) \,|\, \mathbf{D}, \mathbf{\Theta}(t)\big].$$

In the M-step, we try to maximize $Q\big(\mathbf{\Theta}|\mathbf{\Theta}(t)\big)$ with respect to $\mathbf{\Theta}$ to find the new parameter estimate $\mathbf{\Theta}(t{+}1)$. If the M-step is carried out to increase $Q\big(\mathbf{\Theta}|\mathbf{\Theta}(t)\big)$ only but there is no guarantee that $Q\big(\mathbf{\Theta}|\mathbf{\Theta}(t)\big)$ is maximized, this generalized version of the EM algorithm is often called a generalized EM (GEM) algorithm.

4.2 EM Algorithm for ARMA Mixtures

In the context of using ARMA mixtures for clustering, the missing data correspond to the unknown cluster or group membership of each time series $\mathbf{x}_i$. The log-likelihood $\ell(\mathbf{\Theta}; \mathbf{D})$ can thus be expressed as

$$\ell(\mathbf{\Theta}; \mathbf{D}) = \sum_{i=1}^{N} \ln P(\mathbf{x}_i|\omega_{\mathbf{z}_i}, \mathbf{\Phi}_{\mathbf{z}_i}) + \sum_{i=1}^{N} \ln P(\omega_{\mathbf{z}_i}).$$

Given the observed data $\mathbf{D}$ and the current parameter estimate $\mathbf{\Theta}(t)$, the expectation of the complete-data log-likelihood becomes

$$Q\big(\mathbf{\Theta}|\mathbf{\Theta}(t)\big) = \sum_{i=1}^{N}\sum_{k=1}^{M} P\big(\omega_k|\mathbf{x}_i, \mathbf{\Theta}(t)\big) \ln P(\mathbf{x}_i|\omega_k, \mathbf{\Phi}_k) + \sum_{i=1}^{N}\sum_{k=1}^{M} P\big(\omega_k|\mathbf{x}_i, \mathbf{\Theta}(t)\big) \ln P(\omega_k), \quad (2)$$

where the posterior probabilities $P(\omega_k|\mathbf{x}_i, \mathbf{\Theta})$ can be computed using the Bayes rule as

$$P(\omega_k|\mathbf{x}_i, \mathbf{\Theta}) = \frac{P(\mathbf{x}_i|\omega_k, \mathbf{\Phi}_k)P(\omega_k)}{\sum_{u=1}^{M} P(\mathbf{x}_i|\omega_u, \mathbf{\Phi}_u)P(\omega_u)}, \qquad i = 1, 2, \ldots, N \text{ and } k = 1, 2, \ldots, M. \quad (3)$$

The EM algorithm iteratively maximizes the function $Q\big(\mathbf{\Theta}|\mathbf{\Theta}(t)\big)$ until convergence. For each iteration, we compute the posterior probabilities $P(\omega_k|\mathbf{x}_i, \mathbf{\Theta}(t))$ and $Q\big(\mathbf{\Theta}|\mathbf{\Theta}(t)\big)$ using the current parameter estimate $\mathbf{\Theta}(t)$ in the E-step, and update the parameter estimate by maximizing $Q\big(\mathbf{\Theta}|\mathbf{\Theta}(t)\big)$ with respect to $\mathbf{\Theta}$ to obtain $\mathbf{\Theta}(t{+}1)$ in the M-step. Table 1 summarizes our EM algorithm for ARMA mixtures. One possible convergence condition is that the difference in log-likelihood between two time steps, i.e., $\ell(\mathbf{\Theta}(t{+}1); \mathbf{D}) - \ell(\mathbf{\Theta}(t); \mathbf{D})$, is less than some prespecified threshold. Another possibility, which was used in our implementation, is to detect if the change from $\mathbf{\Theta}(t)$ to $\mathbf{\Theta}(t{+}1)$ is significantly large. Detailed derivation of the corresponding M-step equations can be found in Appendix A.

Table 1: EM algorithm for ARMA mixtures

initialize $\mathbf{\Theta}$, $t \leftarrow 0$
  do $t \leftarrow t + 1$
    E-step: Compute posterior probabilities $P(\omega_k|\mathbf{x}_i, \mathbf{\Theta}(t))$ and $Q\big(\mathbf{\Theta}|\mathbf{\Theta}(t)\big)$ using the current parameter estimate and Equation (3)
    M-step: $\mathbf{\Theta}(t{+}1) \leftarrow \arg\max_{\mathbf{\Theta}} Q\big(\mathbf{\Theta}|\mathbf{\Theta}(t)\big)$ using Equations (5), (6) and (10)
  until some convergence condition is satisfied
return $\mathbf{\Theta}(t{+}1)$

5 Experimental Results for Simulated Datasets

As in [26], experiments were conducted on both simulated and real datasets. Instead of handling ARIMA time series directly, a preprocessing step of differencing was first applied to convert each nonstationary ARIMA time series into the corresponding stationary ARMA time series. Moreover, as discussed in [24], ARMA models can be converted into equivalent AR models.[1]

[1] Theoretically speaking, finite-order ARMA models are equivalent to infinite-order AR models. In practice, however, there usually exist finite-order AR models that are good enough for approximation.

Thus, for simplicity, we in fact used mixtures of AR models in all our experiments, although the EM algorithm presented above can be used for general ARMA mixtures.

We implemented our method in Matlab. In addition, we also implemented the hybrid method proposed by Kalpakis *et al.* [26]. The AR coefficients of each time series in a dataset were estimated using the forward-backward method [28] provided by the System Identification Toolbox in Matlab. The LPC cepstral coefficients were then computed based on the estimated AR coefficients. The Euclidean distance between the first eight LPC cepstral coefficients of two time series was used as the distance measure for the PAM partitional distance-based clustering method to partition the time series into clusters.

To evaluate and compare the clustering results obtained by Kalpakis *et al.*'s method (abbreviated in the tables below as CEP for cepstral coefficients) and our method (abbreviated as MAR for mixtures of AR models), we used the cluster similarity measure in [29], which can be defined as

$$Sim(G, A) = \frac{1}{M} \sum_{i=1}^{M} \max_{1 \leq j \leq M} Sim(G_i, A_j),$$

where $G = G_1, G_2, \ldots, G_M$ is the clustering for the ground truth, $A = A_1, A_2, \ldots, A_M$ is that obtained by a clustering method under evaluation, and

$$Sim(G_i, A_j) = \frac{2\,|G_i \cap A_j|}{|G_i| + |A_j|}.$$

Apparently, the similarity measure has values ranging from 0 to 1, with 1 corresponding to the case when G and A are identical. Note that this similarity measure is asymmetric.

5.1 Known Number of Clusters

We first study the simpler scenario with simulated time series data generated by a known number of AR models. We consider two cases separately. The first case involves AR models with the same noise variance, and the second case involves AR models with different noise variances.

5.1.1 Time Series with Same Noise Variance

In this experiment, we used two AR(1) models with their AR coefficients uniformly distributed in the ranges (0.30 ± 0.01) and (0.60 ± 0.01), respectively. The noise variance was 0.01 for both models. Each model generated 15 time series to form the dataset. As expected, both our MAR method and the CEP method worked very well because the two groups of time series are easily separable. The cluster similarity measure was always equal to 1.

We further conducted more experiments on time series generated by two closer AR(1) models. As before, the AR coefficient of one model was uniformly distributed in the range (0.30 ± 0.01), but that for the other model was set to four different ranges in four different experiments, varying from (0.55 ± 0.01) to (0.40 ± 0.01). In each experiment, each model generated 15 time series to form the dataset. Both methods were run 10 times on each dataset. The minimum, average, and maximum values of the cluster similarity measure over 10 trials were recorded. Table 2 summarizes the results obtained by the two methods. Our method is slightly better than CEP when the two AR(1) models are farther from each other, but CEP becomes slightly better when the range of AR coefficient of one model decreases to (0.40 ± 0.01), which is very close to that of the other model.

Table 2: Clustering results for time series generated by two AR(1) models with the same noise variance but different AR coefficient distribution ranges

Range of AR coefficient	Cluster similarity (min/avg/max)	
	MAR	CEP
(0.55 ± 0.01)	(0.93/0.99/1.00)	(0.93/0.98/1.00)
(0.50 ± 0.01)	(0.83/0.93/0.97)	(0.80/0.93/0.97)
(0.45 ± 0.01)	(0.80/0.88/0.93)	(0.71/0.86/0.93)
(0.40 ± 0.01)	(0.63/0.77/0.90)	(0.63/0.79/0.93)

5.1.2 Time Series with Different Noise Variances

We repeated the experiments above under the same setup, except that the two AR(1) models had the same AR coefficient distribution range of (0.30 ± 0.01) but different noise variances of 0.01 and 0.02, respectively. Table 3 shows the results obtained by the two methods. Our method gives perfect clustering of the two groups of time series, but CEP, which makes no use of the noise variances, gives very poor results on this dataset.

Table 3: Clustering results for time series generated by two AR(1) models with the same AR coefficient distribution range but different noise variances

Cluster similarity (min/avg/max)	
MAR	CEP
(1.00/1.00/1.00)	(0.51/0.59/0.67)

5.2 Unknown Number of Clusters

We now consider the more general scenario in which the number of underlying clusters is unknown. We improve our EM algorithm so that the number of clusters can be determined automatically.

The improved EM algorithm shown in Table 4 simply calls the basic EM algorithm multiple times with an increasing number of component models until at least one redundant component model is found. When the number of component models specified is equal to or less than the actual number of clusters, the basic EM algorithm will converge. However, if the number of component models specified is larger than the actual number of clusters, the EM algorithm will not converge within a reasonably large number of iterations. Moreover, some component models will learn to become very similar to each other. Based on these characteristics, we can decide whether too many component models are specified. Hence the correct number of clusters can be determined and returned.

Table 4: Improved EM algorithm for ARMA mixtures

```
initialize m ← 1
    do m ← m + 1
        Call the basic EM algorithm for ARMA mixtures with m component models
    until at least one redundant component model is found
return result with m − 1 models
```

5.2.1 Time Series with Same Noise Variance

In this experiment, we used three AR(1) models to generate the dataset. However, the actual number of clusters was not made known to both methods. The three AR(1) models had the same noise variance of 0.01 but different AR coefficient distribution ranges, namely, (0.20 ± 0.01), (0.50 ± 0.01), and (0.80 ± 0.01). As before, each model generated 15 time series to form the dataset and each method was run 10 times. Table 5 shows the results of the two methods. Our method can always find the correct number of clusters automatically. The CEP method can only give comparable results if the number of clusters specified for the algorithm is correct. However, it cannot find the correct number automatically.

5.2.2 Time Series with Different Noise Variances

We repeated the above experiment on the three models with their noise variances changed to 0.01, 0.02, and 0.01, respectively. Table 6 summarizes the experimental results. It can be seen

Table 5: Clustering results for time series generated by three AR(1) models with the same noise variance but different AR coefficient distribution ranges

MAR		CEP	
Number of clusters found	Cluster similarity (min/avg/max)	Number of clusters specified	Cluster similarity (min/avg/max)
Always 3	(0.98/1.00/1.00)	2	(0.72/0.76/0.78)
		3	(0.98/1.00/1.00)
		4	(0.67/0.69/0.70)
		5	(0.48/0.50/0.54)

that the different noise variances of the closer pairs of models (the first two models as one pair and the last two models as another pair) can help to separate the corresponding groups of time series more effectively. Since our method makes use of noise variance information, it can give better results than the previous experiment. However, there is no significant difference for the CEP method.

Table 6: Clustering results for time series generated by three AR(1) models with different AR coefficient distribution ranges and noise variances

MAR		CEP	
Number of clusters found	Cluster similarity (min/avg/max)	Number of clusters specified	Cluster similarity (min/avg/max)
Always 3	(1.00/1.00/1.00)	2	(0.72/0.76/0.78)
		3	(0.98/1.00/1.00)
		4	(0.67/0.69/0.71)
		5	(0.48/0.51/0.54)

5.3 Discussions

This set of experiments based on simulated datasets allows us to explore the strengths and weaknesses of the two methods under different controlled settings. While our method, like other EM-based methods, generally degrades in clustering performance when the underlying clusters are very close to each other, it is better than Kalpakis *et al.*'s distance-based method under more general situations. Specifically, our method is significantly better when the models have different noise variances. It is also more flexible in determining the underlying number of clusters automatically.

6 Experimental Results for Real Datasets

For comparison, we conducted further experiments with the same four real datasets used by Kalpakis *et al.* [26]. The same preprocessing steps used by them were also applied to the datasets to remove the nonstationarity in the data. Moreover, due to differences in level and scale, a normalization step was applied to generate normalized data so that the time series values fall in the range $[0, 1]$. All the experiments were conducted on both normalized and unnormalized data.

6.1 Personal Income Dataset

The personal income dataset consists of 25 time series representing the per capita personal income from 1929–1999 in 25 states of the USA.[2] The 25 states are divided into two groups based on their personal income growth rate. The east coast states, CA and IL form the first group with a high growth rate, while the mid-west states form the second group with a low growth rate.

Kalpakis *et al.* decided that this dataset could be modeled by ARIMA$(1, 1, 0)$ models. Thus AR(1) coefficients were extracted to compute the LPC cepstral coefficients. For our method, we used a mixture of two AR(1) models. The results for both normalized and unnormalized time series using both methods are shown in Table 7 below. Since the CEP method does not make use of noise variance information, it gives the same results on both normalized and unnormalized data. However, our method can perform better on unnormalized data.

Table 7: Clustering results for personal income dataset

Dataset	Cluster similarity	
	MAR	CEP
Normalized	0.78	0.84
Unnormalized	0.90	0.84

6.2 ECG Dataset

The ECG dataset was obtained from the ECG database at PhysioNet.[3] It consists of two groups of time series. The first group contains 22 time series representing the 2 second ECG recordings

[2] `http://www.bea.gov/bea/regional/spi/`
[3] `http://www.physionet.org/physiobank/database/`

of people having malignant ventricular arrhythmia, and the second group contains 13 time series of the 2 second ECG recordings of healthy people.

For the CEP method, the time series was assumed to be from ARIMA$(2, 3, 0)$ models. So we used a mixture of two AR(2) models for our method. Table 8 shows the results. While our method is worse than CEP for normalized data, they give the same result for unnormalized data.

Table 8: Clustering results for ECG dataset

Dataset	Cluster similarity	
	MAR	CEP
Normalized	0.80	0.94
Unnormalized	0.94	0.94

6.3 Temperature Dataset

The temperature dataset is a collection of time series of the daily temperature in the year 2000 from various places in Florida, Tennessee and Cuba.[4] The dataset we used includes temperature recordings from 9 places in Tennessee, 4 places in northern Florida, 8 places in southern Florida, and 8 places in Cuba. Based on geographical proximity and similarity in temperature, time series from Tennessee and northern Florida form the first group and those from southern Florida and Cuba form the second group. The CEP method assumed ARIMA$(2, 1, 0)$ models and our method used a mixture of two AR(2) models. The clustering results for both methods are shown in Table 9. Since the variances of the two groups of time series are quite different, our method can give very good result on unnormalized data.

Table 9: Clustering results for temperature dataset

Dataset	Cluster similarity	
	MAR	CEP
Normalized	0.58	0.65
Unnormalized	1.00	0.65

[4] `http://lwf.ncdc.noaa.gov/oa/climate/climatedata.html`

6.4 Population Dataset

The population dataset is a collection of 20 time series representing the population estimates from 1900–1999 in 20 states of the USA.[5] The 11 time series in the first group have exponentially increasing trend and the remaining time series in the second group have a stabilizing trend. The CEP method assumed ARIMA$(1, 1, 0)$ models and our method used a mixture of two AR(1) models. Table 10 summarizes the results. Although our method gives the same result as the CEP method for unnormalized data, both are not considered satisfactory in clustering the time series into two groups. The major reason for the unsatisfactory clustering performance is that both methods actually work on stationary time series after the nonstationarity is removed. However, for this dataset, it is the exact nonstationary trend of the time series that is the most discriminating feature.

Table 10: Clustering results for population dataset

Dataset	Cluster similarity	
	MAR	CEP
Normalized	0.62	0.64
Unnormalized	0.64	0.64

6.5 Discussions

Compared with the CEP method, our method can give the same (for two datasets) or better (for another two datasets) results when unnormalized data are used. Our method always works better on unnormalized data because the variance information can be utilized in separating the clusters. However, both our method and the CEP method, due to their nature of modeling stationary ARMA processes only, do not learn the differences in trend of the time series and hence cannot give very satisfactory results for the population dataset. It should be noted, however, that the trends of the two groups of population time series are actually visually distinguishable. Extension of our method to address this issue will be discussed in the next section.

7 Conclusion and Future Work

In this paper, we have proposed a model-based method for clustering univariate ARIMA time series. This mixture-model method, based on mixtures of ARMA models, uses an EM algorithm to learn the mixing coefficients as well as the parameters of the component models. In addition,

[5] `http://eire.census.gov/popest/archives/state/st_stts.php`

the number of clusters in the data can be determined automatically. Experimental results on both simulated and real datasets show that this method is generally effective in clustering time series, and that it compares favorably with the hybrid method proposed recently by Kalpakis *et al.* for similar time series clustering problems.

Our method can be improved in a number of aspects. One aspect is related to parameter initialization for the EM algorithm, which may affect the convergence speed of the algorithm and the quality of the solution found. Currently our method sets the initial prior probabilities of the clusters to be equal, and randomly picks M different time series to initialize the M component models. A possible improvement is to initialize the parameters of the mixture model based on the clustering results of some faster but less accurate method. This is analogous to the use of the k-means algorithm for finding the initial parameter values for an EM algorithm.

Computational speedup can be achieved by pruning away some models if their posterior probabilities become very close to 0, indicating that their significance is negligible. The issue to address here is to decide the appropriate time to apply this pruning operation without making premature decisions. The current version of our learning algorithm applies the EM algorithm multiple times on the entire dataset with different numbers of component models. An interesting direction to explore is to use a hierarchical scheme to search for the correct number of clusters in the data more efficiently. Other model selection methods will also be studied.

One problem with our method, like other EM-based methods, is that its clustering performance can degrade significantly when the underlying clusters are very close to each other. Another problem, as illustrated by the experiment with the population dataset, is that it is not designed for modeling the differences in trend of the time series. A possible extension is to model ARIMA time series without removing the nonstationarity in trend. In our future research, both issues will be addressed.

References

[1] A.K. Jain and R.C. Dubes. *Algorithms for Clustering Data.* Prentice Hall, Englewood Cliffs, NJ, USA, 1988.

[2] G.J. McLachlan and K.E. Basford. *Mixture Models: Inference and Applications to Clustering.* Marcel Dekker, New York, NY, USA, 1988.

[3] C.S. Poulsen. Mixed Markov and latent Markov modelling applied to brand choice behaviour. *International Journal of Research in Marketing*, 7(1):5–19, 1990.

[4] G. Ridgeway. Finite discrete Markov process clustering. Technical Report MSR-TR-97-24, Microsoft Research, Redmond, WA, USA, September 1997.

[5] P. Smyth. Probabilistic model-based clustering of multivariate and sequential data. In *Proceedings of the Seventh International Workshop on Artificial Intelligence and Statistics*, pages 299–304, Fort Lauderdale, FL, USA, 4–6 January 1999.

[6] I.V. Cadez, S. Gaffney, and P. Smyth. A general probabilistic framework for clustering individuals and objects. In *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 140–149, Boston, MA, USA, 20–23 August 2000.

[7] I. Cadez, D. Heckerman, C. Meek, P. Smyth, and S. White. Model-based clustering and visualization of navigation patterns on a web site. Technical Report MSR-TR-00-18, Microsoft Research, Redmond, WA, USA, March 2000. Revised September 2001.

[8] I. Cadez, D. Heckerman, C. Meek, P. Smyth, and S. White. Visualization of navigation patterns on a web site using model-based clustering. In *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 280–284, Boston, MA, USA, 20–23 August 2000.

[9] A.P. Dempster, N.M. Laird, and D.B. Rubin. Maximum likelihood from incomplete data via the EM algorithm (with discussion). *Journal of the Royal Statistical Society, Series B*, 39:1–38, 1977.

[10] P. Sebastiani, M. Ramoni, P. Cohen, J. Warwick, and J. Davis. Discovering dynamics using Bayesian clustering. In *Proceedings of the Third International Symposium on Intelligent Data Analysis*, pages 199–209, Amsterdam, Netherlands, 9–11 August 1999.

[11] M. Ramoni, P. Sebastiani, and P. Cohen. Multivariate clustering by dynamics. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence*, pages 633–638, Austin, TX, USA, 30 July 30 - 3 August 2000.

[12] M. Ramoni, P. Sebastiani, and P. Cohen. Bayesian clustering by dynamics. *Machine Learning*, 47(1):91–121, 2002.

[13] S. Kullback and R.A. Leibler. On information and sufficiency. *Annals of Mathematical Statistics*, 22:79–86, 1951.

[14] L.R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[15] L.R. Rabiner, C.H. Lee, B.H. Juang, and J.G. Wilpon. HMM clustering for connected word recognition. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 1, pages 405–408, Glasgow, UK, 23–26 May 1989.

[16] T. Oates, L. Firoiu, and P.R. Cohen. Using dynamic time warping to bootstrap HMM-based clustering of time series. In R. Sun and C.L. Giles, editors, *Sequence Learning: Paradigms, Algorithms, and Applications*, pages 35–52. Springer-Verlag, Berlin, Germany, 2001.

[17] A. Krogh, M. Brown, I.S. Mian, K. Sjölander, and D. Haussler. Hidden Markov models in computational biology: applications to protein modeling. *Journal of Molecular Biology*, 235(5):1501–1531, 1994.

[18] L.M.D. Owsley, L.E. Atlas, and G.D. Bernard. Self-organizing feature maps and hidden Markov models for machine-tool monitoring. *IEEE Transactions on Signal Processing*, 45(11):2787–2798, 1997.

[19] P. Smyth. Clustering sequences with hidden Markov models. In *Advances in Neural Information Processing Systems 9*, pages 648–654. MIT Press, 1997.

[20] C. Li and G. Biswas. A Bayesian approach to temporal data clustering using hidden Markov models. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 543–550, Stanford, CA, USA, 29 June - 2 July 2000.

[21] M.P. Perrone and S.D. Connell. K-means clustering for hidden Markov models. In *Proceedings of the Seventh International Workshop on Frontiers in Handwriting Recognition*, pages 229–238, Amsterdam, Netherlands, 11–13 September 2000.

[22] M.H. Law and J.T. Kwok. Rival penalized competitive learning for model-based sequence clustering. In *Proceedings of the Fifteenth International Conference on Pattern Recognition*, volume 2, pages 195–198, Barcelona, Spain, 3–7 September 2000.

[23] G.E.P. Box and G.M. Jenkins. *Time Series Analysis: Forecasting and Control*. Holden Day, San Francisco, CA, USA, 1970. Revised 1976.

[24] G.E.P. Box, G.M. Jenkins, and G.C. Reinsel. *Time Series Analysis: Forecasting and Control*. Prentice Hall, Englewood Cliffs, NJ, USA, 3rd edition, 1994.

[25] H.Y. Kwok, C.M. Chen, and L. Xu. Comparison between mixture of ARMA and mixture of AR model with application to time series forecasting. In *Proceedings of the Fifth International Conference on Neural Information Processing*, pages 1049–1052, Kitakyushu, Japan, 21–23 October 1998.

[26] K. Kalpakis, D. Gada, and V. Puttagunta. Distance measures for effective clustering of ARIMA time-series. In *Proceedings of the IEEE International Conference on Data Mining*, pages 273–280, San Jose, CA, USA, 29 November - 2 December 2001.

[27] R.A. Redner and H.F. Walker. Mixture densities, maximum likelihood and the EM algorithm. *SIAM Review*, 26(2):195–239, 1984.

[28] L. Ljung. *System Identification Toolbox User's Guide*. MathWorks, 5th edition, 2000.

[29] M. Gavrilov, D. Anguelov, P. Indyk, and R. Motwani. Mining the stock market: Which measure is best? In *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 487–496, Boston, MA, USA, 20–23 August 2000.

A Derivation of M-Step Equations of EM Algorithm for ARMA Mixtures

Recall that in the M-step of the EM algorithm, we find a new parameter estimate by maximizing $Q\big(\mathbf{\Theta}|\mathbf{\Theta}(t)\big)$.

We first maximize $Q\big(\mathbf{\Theta}|\mathbf{\Theta}(t)\big)$ with respect to each $P(\omega_k)$, subject to the constraint that $\sum_{k=1}^{M} P(\omega_k) = 1$. This can be solved using the Lagrangian multiplier method:

$$\frac{\partial}{\partial P(\omega_k)}\bigg(Q\big(\mathbf{\Theta}|\mathbf{\Theta}(t)\big) - \lambda\Big(\sum_{k=1}^{M} P(\omega_k) - 1\Big)\bigg) = \frac{\partial Q\big(\mathbf{\Theta}|\mathbf{\Theta}(t)\big)}{\partial P(\omega_k)} - \lambda = 0. \tag{4}$$

We first rewrite $Q\big(\mathbf{\Theta}|\mathbf{\Theta}(t)\big)$ as

$$Q\big(\mathbf{\Theta}|\mathbf{\Theta}(t)\big) = R(\mathbf{\Theta}) + S(\mathbf{\Theta}),$$

where

$$R(\mathbf{\Theta}) = \sum_{i=1}^{N}\sum_{k=1}^{M} P\big(\omega_k|\mathbf{x}_i, \mathbf{\Theta}(t)\big) \ln P(\mathbf{x}_i|\omega_k, \mathbf{\Phi}_k),$$

$$S(\mathbf{\Theta}) = \sum_{i=1}^{N}\sum_{k=1}^{M} P\big(\omega_k|\mathbf{x}_i, \mathbf{\Theta}(t)\big) \ln P(\omega_k).$$

Since

$$\frac{\partial S(\mathbf{\Theta})}{\partial P(\omega_k)} = \frac{1}{P(\omega_k)} \sum_{i=1}^{N} P(\omega_k|\mathbf{x}_i, \mathbf{\Theta}(t)),$$

we can apply the above constraint to obtain the best estimate as

$$\hat{P}(\omega_k) = \frac{1}{N}\sum_{i=1}^{N} P\big(\omega_k|\mathbf{x}_i, \mathbf{\Theta}(t)\big). \tag{5}$$

To get the best parameter estimate of σ_k^2, we need to solve this equation:

$$\frac{\partial R(\mathbf{\Theta})}{\partial \sigma_k^2} = \sum_{i=1}^{N} P\big(\omega_k|\mathbf{x}_i, \mathbf{\Theta}(t)\big)\big[-\frac{n}{2\sigma_k^2} + \frac{1}{2\sigma_k^4}\sum_{t=1}^{n} e_{i,t}^2\big] = 0,$$

that is,

$$\sum_{i=1}^{N} P\big(\omega_k|\mathbf{x}_i, \mathbf{\Theta}(t)\big)\big[-\frac{n\sigma_k^2}{2} + \frac{1}{2}\sum_{t=1}^{n} e_{i,t}^2\big] = 0,$$

or

$$\sum_{i=1}^{N}\Big[-P\big(\omega_k|\mathbf{x}_i, \mathbf{\Theta}(t)\big)\frac{n\sigma_k^2}{2} + \frac{1}{2}P\big(\omega_k|\mathbf{x}_i, \mathbf{\Theta}(t)\big)\sum_{t=1}^{n} e_{i,t}^2\Big] = 0.$$

We get the best parameter estimate of σ_k^2 as

$$\hat{\sigma}_k^2 = \frac{\sum_{i=1}^{N} \Big[P\big(\omega_k|\mathbf{x}_i, \mathbf{\Theta}(t)\big) \sum_{t=1}^{n} e_{i,t}^2\Big]}{\sum_{i=1}^{N} \Big[nP\big(\omega_k|\mathbf{x}_i, \mathbf{\Theta}(t)\big)\Big]}. \tag{6}$$

To get the best estimates of $\boldsymbol{\phi}_k$ and $\boldsymbol{\theta}_k$, we need to solve the following equations:

$$\frac{\partial R(\mathbf{\Theta})}{\partial \phi_{k,j}} = \sum_{i=1}^{N} \Big[P\big(\omega_k|\mathbf{x}_i, \mathbf{\Theta}(t)\big)\Big(-\frac{1}{2\sigma_k^2}\Big) \sum_{t=1}^{n} \big(2\, e_{i,t}\, \frac{\partial e_{i,t}}{\partial \phi_{k,j}}\big)\Big] = 0, \qquad j = 0, 1, 2, \ldots, p.$$

Since

$$\frac{\partial e_{i,t}}{\partial \phi_{k,0}} = -1,$$

we have

$$\sum_{i=1}^{N} \Big[P\big(\omega_k|\mathbf{x}_i, \mathbf{\Theta}(t)\big) \sum_{t=1}^{n} e_{i,t}\Big] = 0,$$

or

$$\sum_{i=1}^{N} \Big[P\big(\omega_k|\mathbf{x}_i, \mathbf{\Theta}(t)\big) \sum_{t=1}^{n} \big(x_{i,t} - \phi_{k,0} - \sum_{l=1}^{p} \phi_{k,l}\, x_{i,t-l} - \sum_{l=1}^{q} \theta_{k,l}\, e_{i,t-l}\big)\Big] = 0,$$

or

$$\sum_{i=1}^{N} \Big[P\big(\omega_k|\mathbf{x}_i, \mathbf{\Theta}(t)\big)\big(\sum_{t=1}^{n} x_{i,t} - n\, \phi_{k,0} - \sum_{l=1}^{p} \phi_{k,l} \sum_{t=1}^{n} x_{i,t-l} - \sum_{l=1}^{q} \theta_{k,l} \sum_{t=1}^{n} e_{i,t-l}\big)\Big] = 0,$$

that is,

$$\begin{aligned} & n\, \phi_{k,0} \sum_{i=1}^{N} \Big[P\big(\omega_k|\mathbf{x}_i, \mathbf{\Theta}(t)\big)\Big] + \\ & \sum_{l=1}^{p} \phi_{k,l} \sum_{i=1}^{N} \Big[P\big(\omega_k|\mathbf{x}_i, \mathbf{\Theta}(t)\big) \sum_{t=1}^{n} x_{i,t-l}\Big] + \\ & \sum_{l=1}^{q} \theta_{k,l} \sum_{i=1}^{N} \Big[P\big(\omega_k|\mathbf{x}_i, \mathbf{\Theta}(t)\big) \sum_{t=1}^{n} e_{i,t-l}\Big] \\ = & \sum_{i=1}^{N} \Big[P\big(\omega_k|\mathbf{x}_i, \mathbf{\Theta}(t)\big) \sum_{t=1}^{n} x_{i,t}\Big]. \end{aligned} \tag{7}$$

Since

$$\frac{\partial e_{i,t}}{\partial \phi_{k,j}} = -x_{i,t-j}, \qquad j = 1, 2, \ldots, p,$$

we have

$$\sum_{i=1}^{N}\Big[P\big(\omega_k|\mathbf{x}_i,\boldsymbol{\Theta}(t)\big)\sum_{t=1}^{n}\big(e_{i,t}\,x_{i,t-j}\big)\Big]=0,$$

or

$$\sum_{i=1}^{N}\Big[P\big(\omega_k|\mathbf{x}_i,\boldsymbol{\Theta}(t)\big)\sum_{t=1}^{n}\big(x_{i,t}\,x_{i,t-j}-\phi_{k,0}\,x_{i,t-j}-\sum_{l=1}^{p}\phi_{k,l}\,x_{i,t-l}\,x_{i,t-j}-\sum_{l=1}^{q}\theta_{k,l}\,e_{i,t-l}\,x_{i,t-j}\big)\Big]=0,$$

or

$$\sum_{i=1}^{N}\Big[P\big(\omega_k|\mathbf{x}_i,\boldsymbol{\Theta}(t)\big)\big(\sum_{t=1}^{n}x_{i,t}\,x_{i,t-j}-\phi_{k,0}\sum_{t=1}^{n}x_{i,t-j}-\sum_{l=1}^{p}\phi_{k,l}\sum_{t=1}^{n}x_{i,t-l}\,x_{i,t-j}-\sum_{l=1}^{q}\theta_{k,l}\sum_{t=1}^{n}e_{i,t-l}\,x_{i,t-j}\big)\Big]=0,$$

that is,

$$\begin{aligned}&\phi_{k,0}\sum_{i=1}^{N}\Big[P\big(\omega_k|\mathbf{x}_i,\boldsymbol{\Theta}(t)\big)\sum_{t=1}^{n}x_{i,t-j}\Big]+\\&\sum_{l=1}^{p}\phi_{k,l}\sum_{i=1}^{N}\Big[P\big(\omega_k|\mathbf{x}_i,\boldsymbol{\Theta}(t)\big)\sum_{t=1}^{n}x_{i,t-l}\,x_{i,t-j}\Big]+\\&\sum_{l=1}^{q}\theta_{k,l}\sum_{i=1}^{N}\Big[P\big(\omega_k|\mathbf{x}_i,\boldsymbol{\Theta}(t)\big)\sum_{t=1}^{n}e_{i,t-l}\,x_{i,t-j}\Big]\\=&\sum_{i=1}^{N}\Big[P\big(\omega_k|\mathbf{x}_i,\boldsymbol{\Theta}(t)\big)\sum_{t=1}^{n}x_{i,t}\,x_{i,t-j}\Big],\qquad j=1,2,\ldots,p.\end{aligned}\tag{8}$$

For the same reason, we have

$$\frac{\partial e_{i,t}}{\partial\theta_{k,j}}=-e_{i,t-j},$$

$$\begin{aligned}&\phi_{k,0}\sum_{i=1}^{N}\Big[P\big(\omega_k|\mathbf{x}_i,\boldsymbol{\Theta}(t)\big)\sum_{t=1}^{n}e_{i,t-j}\Big]+\\&\sum_{l=1}^{p}\phi_{k,l}\sum_{i=1}^{N}\Big[P\big(\omega_k|\mathbf{x}_i,\boldsymbol{\Theta}(t)\big)\sum_{t=1}^{n}x_{i,t-l}\,e_{i,t-j}\Big]+\\&\sum_{l=1}^{q}\theta_{k,l}\sum_{i=1}^{N}\Big[P\big(\omega_k|\mathbf{x}_i,\boldsymbol{\Theta}(t)\big)\sum_{t=1}^{n}e_{i,t-l}\,e_{i,t-j}\Big]\\=&\sum_{i=1}^{N}\Big[P\big(\omega_k|\mathbf{x}_i,\boldsymbol{\Theta}(t)\big)\sum_{t=1}^{n}x_{i,t}\,e_{i,t-j}\Big],\qquad j=1,2,\ldots,q.\end{aligned}\tag{9}$$

Combining the $(1+p+q)$ equations from Equations (7), (8) and (9), we get

$$\widehat{W}_k\;\hat{\boldsymbol{\delta}}_k=\widehat{U}_k,$$

or

$$\widehat{\boldsymbol{\delta}}_k = \big(\widehat{W}_k\big)^{-1}\widehat{U}_k, \tag{10}$$

where

$$\widehat{\boldsymbol{\delta}}_k = (\phi_{k,0}\ \phi_{k,1}\ \phi_{k,2}\ \ldots\ \phi_{k,p}\ \theta_{k,1}\ \theta_{k,2}\ \ldots\ \theta_{k,q})^T,$$

$$\widehat{W}_k = \sum_{i=1}^{N}\left[P\big(\omega_k|\mathbf{x}_i,\boldsymbol{\Theta}(t)\big)\begin{pmatrix} A & B \\ C & D \end{pmatrix}\right],$$

$$\widehat{U}_k = \sum_{i=1}^{N}\left[P\big(\omega_k|\mathbf{x}_i,\boldsymbol{\Theta}(t)\big)\big(a_0\ a_1\ a_2\ \ldots\ a_p\ c_1\ c_2\ \ldots\ c_q\big)^T\right],$$

and

$$A = (a_{uv})_{(0\sim p)\times(0\sim p)},\quad B = (b_{uv})_{(0\sim p)\times(1\sim q)},\quad C = (c_{uv})_{(1\sim q)\times(0\sim p)},\quad D = (d_{uv})_{(1\sim q)\times(1\sim q)};$$

$$a_{00} = n,$$

$$a_{0v} = \sum_{t=1}^{n} x_{i,t-v},\quad v = 1,2,\ldots,p,$$

$$b_{0v} = \sum_{t=1}^{n} e_{i,t-v},\quad v = 1,2,\ldots,q,$$

$$a_{u0} = \sum_{t=1}^{n} x_{i,t-u},\quad u = 1,2,\ldots,p,$$

$$c_{u0} = \sum_{t=1}^{n} e_{i,t-u},\quad u = 1,2,\ldots,q,$$

$$a_{uv} = \sum_{t=1}^{n} x_{i,t-u}\,x_{i,t-v},\quad u = 1,2,\ldots,p,\quad v = 1,2,\ldots,p,$$

$$b_{uv} = \sum_{t=1}^{n} e_{i,t-u}\,x_{i,t-v},\quad u = 1,2,\ldots,p,\quad v = 1,2,\ldots,q,$$

$$c_{uv} = \sum_{t=1}^{n} x_{i,t-u}\,e_{i,t-v},\quad u = 1,2,\ldots,q,\quad v = 1,2,\ldots,p,$$

$$d_{uv} = \sum_{t=1}^{n} e_{i,t-u}\,e_{i,t-v},\quad u = 1,2,\ldots,q,\quad v = 1,2,\ldots,q,$$

$$a_0 = \sum_{t=1}^{n} x_{i,t},$$

$$a_v = \sum_{t=1}^{n} x_{i,t}\,x_{i,t-v},\quad v = 1,2,\ldots,p,$$

$$c_v = \sum_{t=1}^{n} x_{i,t}\,e_{i,t-v},\quad v = 1,2,\ldots,q.$$