

# Integrating Hidden Markov Models and Spectral Analysis for Sensory Time Series Clustering

Jie Yin and Qiang Yang  
Department of Computer Science  
Hong Kong University of Science and Technology  
Clear Water Bay, Kowloon, Hong Kong, China  
{yinjie, qyang}@cs.ust.hk

## Abstract

*We present a novel approach for clustering sequences of multi-dimensional trajectory data obtained from a sensor network. The sensory time-series data present new challenges to data mining, including uneven sequence lengths, multi-dimensionality and high levels of noise. We adopt a principled approach, by first transforming all the data into an equal-length vector form while keeping as much temporal information as we can, and then applying dimensionality and noise reduction techniques such as spectral clustering to the transformed data. Experimental evaluation on synthetic and real data shows that our proposed approach outperforms standard model-based clustering algorithms for time series data.*

## 1 Introduction

Clustering is a fundamental and widely used technique in machine learning and data mining areas. Traditional clustering algorithms typically assume that data are represented by feature vectors of equal lengths and low dimensionality, and contain little noise. Using traditional approaches, such data are normally handled either by model-based methods or similarity-based methods [6]. However, as pointed out by [7], in many real-world applications, the dynamic characteristics of an environment often evolve over time, and as such, produce data that are of multi-dimensionality, contain noise and are of uneven lengths. A typical example is a sensor network, which has gained a great amount of interest in recent years. In a sensor network, a moving object may receive sequences of signals, whose values are stochastic in nature; that is, they may change significantly depending on the relative distance between the moving object and a set of sensors in the wireless environment. In such a situation, it is highly desirable to have a method that can produce accurate

clusters even when the data are non-traditional.

In this paper, we address the problem of clustering temporal sequences into different groups. Consider a data set  $C$  consisting of  $N$  temporal sequences,  $C = \{Y_1, Y_2, \dots, Y_N\}$ . Each sequence  $Y$  is a multivariate time series containing  $T$  measurements from a set of sensors such that  $Y = \{O_t | 1 \leq t \leq T\}$ . The observations  $O_t$  are potentially multi-dimensional signal vectors containing stochastic values measured from each sensor at a given time point  $t$ . Our ultimate goal is to automatically partition a set of  $N$  temporal sequences  $C = \{Y_1, Y_2, \dots, Y_N\}$  into  $K$  clusters such that the inter-cluster similarity is small and intra-cluster similarity is large. However, a key question is: when  $|O_t|$  is very large and  $O_t$  contains lots of noise, and when  $|Y_i|$  are all of different lengths, how can we obtain the clusters of high quality?

Clustering multi-dimensional sensory time series is inherently more complex than clustering traditional data that are of fixed dimensions and contain little noise. First, temporal sequences may be of different lengths, making it difficult to embed them into a metric space and use a distance measure such as the Euclidean distance to determine the similarity matrix. For sensory data obtained from a trajectory in a sensor network, the sequences obtained at different times may be of different lengths. Second, the data obtained by sensors at different time points are stochastic in nature, making it very difficult to apply straightforward similarity measures. Moreover, each sequence may consist of varying amounts of temporal information. Therefore, it is highly desirable to design an effective transformation method to extract important temporal information from the sequences while reducing the noise and dimensionality of the data.

In this paper, we propose a novel clustering algorithm to deal with the above-mentioned problems. Inspired by the work of [15], we first devise an affinity similarity matrix between the observation sequences. Our similarity matrix is obtained by modeling each sequence as a hidden Markov

model (HMM), so that the log-likelihood of each sequence given each model can be computed. These probability measures are then used to construct a uniform-length feature-vector to capture the temporal information from each sequence, regardless of its length. We contend and experimentally verify that this transformation step indeed captures the needed global temporal information, even when the data are sequences of vectors.

Our second innovation is reducing the noise and dimensionality at the same time. After the first step, each time series is modeled as an HMM. This representation allows us to construct an affinity similarity matrix for all the sequences. We then apply spectral methods [10, 11] on the similarity matrix for noise and dimensionality reduction. Specifically, a spectral clustering algorithm uses significant eigenvectors of the affinity matrix to map the original samples into a lower dimensional subspace. These vectors are then clustered by standard clustering algorithms in the Euclidean space. Through performing eigendecomposition, a spectral analysis algorithm helps reduce the uncertainty of the similarity matrix. This enhances the quality of subsequent clustering. We demonstrate the effectiveness of the two-phase approach through tests on synthetic and real data. The real data are obtained from a wireless local-area network (WLAN) environment, which consist of a number of access points (APs) that are used to track a client's movement trajectories. We show that our proposed algorithm can produce very good clustering results in this uncertain environment.

The remainder of this paper is organized as follows. Section 2 discusses previous work related to clustering algorithms. Section 3 briefly reviews HMMs and the algorithms used to train HMMs from the observation data. Section 4 describes our proposed algorithm for clustering temporal sequences. Section 5 presents the results of experiments using both synthetic and real data. Section 6 concludes the paper and discusses future work.

## 2 Related Work

Over the years, temporal-sequence clustering has attracted much attention in research and practice, because many problems that range from stock-market analysis, security monitoring, to gene analysis involve time-series clustering. In general, clustering can be classified into two broad categories [2]: *model-based approaches* [1, 5, 15] and *similarity-based approaches* [4, 13].

For similarity-based approaches, the main task is to define pairwise distances between all the sequences and then apply distance-based clustering algorithms. For example, Eisen et al. [4] adopt correlation coefficients to define the similarity among gene expression data from different time-course experiments. Agglomerative hierarchical clustering

is then applied to find clusters of genes with similar patterns of expression. Oates et al. [13] use Dynamic Time Warping (DTW) to measure the similarities between multivariate experiences of mobile robots. For complex problem domains, similarity-based approaches encounter great difficulty in how to define effective similarity measures. This definition, which is difficult to obtain, can affect the clustering quality to a large extent.

Model-based approaches rely on an analytical model for each cluster where the objective is to find the best models to fit the observation sequences. Examples of models include regression models [5], ARMA models [16] and HMMs [1, 3, 9, 15]. Among these models, HMMs have attracted increasing attention over the last decade. Smyth [15] presents a probabilistic model-based approach to clustering sequences using HMMs. This approach first devises a pairwise distance matrix between observation sequences by computing a symmetrized similarity. This similarity is obtained by training an HMM for each sequence, so that the log-likelihood of each sequence, given each model, can be computed. This information is then used to cluster the sequences into  $K$  groups using a hierarchical clustering algorithm. After that, one HMM is trained for each cluster; the resulting  $K$  models are then merged into a "composite" global HMM. This initial estimate is further refined using the EM algorithm. As a result, a global HMM for modeling all the sequences is obtained. In [9], the model-based HMM clustering problem is addressed by focusing on the model selection issue, i.e., searching for the best HMM topology and finding the most likely number of clusters. In [12], the clustering result obtained using DTW as a similarity metric is used to provide an estimate of  $K$  and to yield an initial partitioning of the data. While model-based approaches provide a general probabilistic framework for clustering temporal sequences, the quality of clustering depends critically on the initial conditions. In addition, since an HMM assumes that the observations are independent, the EM algorithm used to train the mixture model cannot achieve a good resolution when a large amount of noise appears in consecutive observations.

Our work is also closely related to *spectral clustering*, a new clustering algorithm that has emerged over the past few years and has been successfully applied to the problem of image segmentation [10, 11]. These methods use significant eigenvectors constructed from a pairwise similarity matrix between pixels and then group the pixels into images in a spectral domain. However, most spectral methods assume the number of clusters  $K$  to be known in advance, whereas the estimation of the optimal number of clusters has not been well studied. In this paper, we apply spectral clustering techniques to help remove the noise in the temporal data and to automatically determine the number of clusters for the problem domain.

### 3 Modeling Temporal Features with HMMs

Our objective is to cluster multi-dimensional time series data that originate from tracking a moving object in a sensor network. These sequential data are different in sequence lengths and vary greatly in values due to a high level of uncertainty. Therefore, how to model these data for better similarity measures is a challenging issue.

Our observation is that, since such data are *generated* by a hidden mechanism associated with an underlying moving object, it is desirable to model such data using a generative model-based method. Among others, HMMs have been demonstrated empirically to be capable of modeling such generative processes in a wide variety of real-world applications that include speech recognition [14] and gesture recognition [1]. Therefore, we adopt HMMs to model the temporal sequences for our solution.

An HMM is a non-deterministic stochastic *finite state automata*. The basic structure of an HMM consists of a connected set of states, each of which emits an observable output. A first-order continuous HMM with Gaussian observation density is formally defined by:

1. A set of  $Q$  states,  $S = \{S_1, S_2, \dots, S_Q\}$ .
2. The initial state probability distribution  $\pi = \{\pi_i\}$ , where  $\pi_i = P(q_1 = S_i)$ ,  $1 \leq i \leq Q$ .
3. The state transition probability distribution  $A = \{a_{ij}\}$ , where  $a_{ij} = P(q_{t+1} = S_j | q_t = S_i)$ ,  $1 \leq i, j \leq Q$ .
4. The observation probability density is  $b_j(O_t) = \sum_{m=1}^M c_{jm} \mathcal{N}(O_t; \mu_{jm}, \Sigma_{jm})$ ,  $1 \leq j \leq Q$ , where  $O_t$  is the observation vector being modeled,  $c_{jm}$  is the mixture coefficient for the  $m^{th}$  Gaussian mixture in the state  $j$ , and  $\mathcal{N}$  is a Gaussian density with the mean vector  $\mu_{jm}$  and the covariance matrix  $\Sigma_{jm}$  for the  $m^{th}$  mixture component in the state  $j$ .

Above,  $O_t$  and  $q_t$  indicate the observation and state at time  $t$ , respectively. The parameters of a continuous HMM can be represented in the following compact form

$$\lambda = \{\pi, A, \mu, \Sigma\}. \quad (1)$$

Let  $Y$  be an observation sequence, and let  $\lambda$  be the parameters of an HMM. The following are the main tasks of an HMM learning algorithm [14]:

1. Compute the probability of the observation sequence given the model, i.e.,  $P(Y|\lambda)$  (the forward-backward algorithm).
2. Find an optimal sequence of states that maximizes the probability of the observation sequence  $Y$  (the Viterbi algorithm).

3. Learn the parameters  $\lambda$  that maximize the probability of the observation sequence  $P(Y|\lambda)$  (the Baum-Welch algorithm).

Using the HMMs, each sequence can now be modeled as a set of model parameters. However, it is unreliable to measure the sequence similarities by directly using these parameters. We tackle this problem using spectral clustering in the next section.

### 4 The Proposed Clustering Algorithm

In this section, we present our approach to the problem of clustering temporal sequences, which is referred to as the HMM-Spectral algorithm in this paper. Figure 1 shows the flow diagram of our proposed HMM-Spectral algorithm. Given a set of temporal sequences  $\{Y_1, Y_2, \dots, Y_N\}$ , we first construct an affinity similarity matrix  $S$  from all the sequences. A spectral clustering algorithm is then applied to the affinity matrix to group these sequences into  $K$  clusters, where  $K$  can be automatically determined from the data. In the following, we discuss these two major steps in detail.

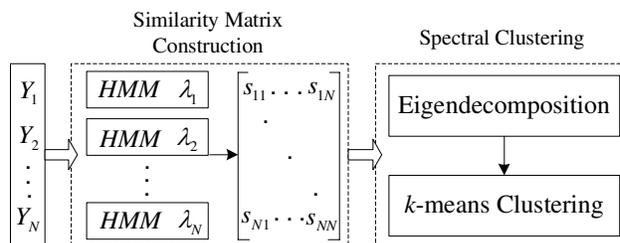


Figure 1. Flow diagram of our HMM-Spectral algorithm

#### 4.1 Constructing Similarity Matrix

To construct the similarity matrix, we fit  $N$   $Q$ -state HMMs, one for each individual sequence  $Y_i$ ,  $1 \leq i \leq N$ , by using the Baum-Welch algorithm [14]. These HMMs can be initialized in a default manner: we first set the transition matrices to be uniform; we then set the means and variances from clusters learned through a  $k$ -means algorithm. For each fitted model with parameters  $\lambda_i$ , we calculate the log-likelihood of each of the  $N$  sequences given the model parameters  $\lambda_i$ . The log-likelihood value for each pair of sequence and HMM is computed as follows:

$$L(Y_j; \lambda_i) = \log P(Y_j | \lambda_i), \quad 1 \leq i, j \leq N. \quad (2)$$

This is done by applying the standard forward-backward algorithm [14]. We can thus obtain a log-likelihood distance matrix  $L$ . This distance matrix is clearly not symmetric. Instead, we define the distance between two sequences  $Y_i$  and

$Y_j$  using the mutual fitness measure as follows:

$$d(Y_i, Y_j) = |L(Y_i; \lambda_i) + L(Y_j; \lambda_j) - L(Y_j; \lambda_i) - L(Y_i; \lambda_j)|, \quad (3)$$

which represents the cross-fitness of two sequences to the models. In this equation, the terms  $L(Y_i; \lambda_i)$  and  $L(Y_j; \lambda_j)$  indicate the likelihood of the sequences given their own fitted models. The cross terms  $L(Y_j; \lambda_i)$  and  $L(Y_i; \lambda_j)$  indicate the likelihood of a sequence generated by the fitted model of another sequence. On the one hand, if two sequences are identical, the cross terms would have maximum values. Thus Equation 3 would be equal to zero. On the other hand, if two sequences are different, their likelihood of being generated from other models would be small. Thus the distance between them would be large. In this way, we can transform the temporal sequences into a set of new feature vectors in the “log-likelihood space”.

The motivation behind the above transformation is as follows. The hypothesis is that all the sequences are generated by  $K$  models. Thus, when we fit models to an individual sequence, we might get noisy estimates of model parameters but the parameters should be clustered in some manner into  $K$  groups based on their true values. Clustering directly in the parameter space would be inappropriate because the distance between parameters is hard to define; however, the log-likelihoods provide a natural way to define pairwise distances between sequences. The distance matrix of Equation (3) is taken as input to the subsequent clustering process.

We construct an affinity matrix  $S$  from the distance measure. Each element  $s_{ij}$  of the matrix reflects the similarity of the corresponding sequences  $i$  and  $j$ . The similarity matrix is defined as follows:

$$s_{ij} = \begin{cases} \exp\left(-\frac{d(i,j)}{2\sigma^2}\right) & \text{for } i \neq j, \\ 0 & \text{for } i = j, \end{cases} \quad (4)$$

where  $d(i, j)$  is the distance between the sequences  $i$  and  $j$ . Clearly, the similarity matrix  $S$  is a symmetric and affinity matrix because  $s_{ij} = s_{ji} \geq 0$  for any pair of sequences  $i$  and  $j$  where  $i \neq j$ . Here, the scaling parameter  $\sigma$  controls how fast  $s_{ij}$  falls off with the distance between  $i$  and  $j$ . While this parameter is usually pre-specified, Ng et al. [11] proposed a method of choosing  $\sigma$  automatically, which we adopt in this paper.

## 4.2 Applying Spectral Clustering to Remove Noise

After the similarity matrix is constructed, we apply spectral clustering methods to partition the sequences into  $K$  clusters; we discuss how to obtain the appropriate value for  $K$  in Section 4.3. Given an  $N \times N$  affinity matrix  $S$ , each element  $s_{ij}$  can be viewed as the similarity between the vectors  $v_i$  and  $v_j$ . For an undirected graph  $G$  with vertices  $v_i$

and edges  $s_{ij}$ , where  $1 \leq i, j \leq N$ , the matrix  $S$  is considered as an adjacency matrix for  $G$ . Let  $d_i = \sum_{j \in V} s_{ij}$  be the degree of vertex  $v_i$ , and let  $D$  be a diagonal matrix with  $d_i$  being its diagonal element. We can obtain a normalized stochastic matrix:

$$M = SD^{-1}, \quad D = \text{diag}(d_1, \dots, d_N), \quad (5)$$

where the sum of each row is one. Based on the definition of a Markov chain,  $m_{ij}$  represents the transition probability of moving from  $v_i$  to  $v_j$ . In practice, we consider a matrix

$$L = D^{-1/2}MD^{1/2} = D^{-1/2}SD^{-1/2}, \quad (6)$$

where  $L$  is symmetric and stable in eigendecomposition [11]. Then, the symmetric matrix  $L$  can be decomposed into the following form:

$$L = X\Lambda X^T, \quad (7)$$

where  $X = [x_1, x_2, \dots, x_N]$  is a matrix by stacking the eigenvectors of  $L$  in columns;  $\Lambda = \text{diag}(\lambda_1, \dots, \lambda_N)$  is a diagonal matrix with the nonnegative singular eigenvalues in descending order along the diagonal, that is,  $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_N \geq 0$ . These eigenvalues represent the importance of the corresponding eigenvectors. Since the top  $K$  eigenvectors,  $K \leq N$ , can capture a significant amount of information on the original samples, we can map the original samples into the  $K$  dimensional vectors in the spectral domain and then apply the standard clustering algorithms based on Euclidean distance.

The spectral clustering algorithm we apply is similar to the one proposed in [11]. Given the number of clusters  $K$ , the algorithm works as follows:

### Spectral Clustering (See Figure 1)

- (1) Find  $K$  principal eigenvectors  $x_1, x_2, \dots, x_K$ , corresponding to the  $K$  largest eigenvalues of  $L$ , and form a matrix  $P = [x_1, x_2, \dots, x_K] \in \mathbb{R}^{N \times K}$  by stacking the eigenvectors in columns.
- (2) Normalize the rows of the matrix  $P$  so that they have unit Euclidean norm.
- (3) Treating each row of  $P$  as a point in  $\mathbb{R}^K$ , cluster them into  $K$  clusters via the  $k$ -means clustering algorithm.
- (4) Assign the original sample  $v_i$  to cluster  $j$  if and only if the row  $i$  of the matrix  $P$  is assigned to cluster  $j$ .

## 4.3 Estimating the Number of Clusters

The main assumption of the above clustering algorithm is that the number of clusters  $K$  needs to be pre-specified. To estimate the number of clusters, an appropriate criterion

is required to measure the quality of the resultant clusters. For a specific number of clusters  $J$ , we compute a correlation matrix  $Q_J = P_J P_J'$  when the normalized matrix  $P_J$  is obtained after step (2) of the *Spectral Clustering* algorithm. Each element  $q_{ij}$  of the matrix  $Q_J$  represents the similarity between the vectors  $v_i$  and  $v_j$ . The closer to one  $q_{ij}$  is, the more similar two vectors  $v_i$  and  $v_j$  are in the spectral domain. Therefore, based on the matrix  $Q_J$ , we can compute a quality score  $\alpha_J$  using the clustering result as follows:

$$\alpha_J = \sum_{c=1}^J \frac{1}{N_c} \sum_{i,j \in Z_c} q_{ij}, \quad (8)$$

where  $Z_c$  is the set of sequences included in the cluster  $c$ , and  $N_c$  is the number of sequences in  $Z_c$ . This quality score would have a higher value if the sequences in each cluster are more similar. Therefore, the number of clusters can be automatically determined by evaluating the local maximum value of this quality score.

In summary, our proposed algorithm works as follows: given a maximum number of clusters  $K_{max}$ , for  $J = 1, 2, \dots, K_{max}$ , iterate the steps of *Spectral Clustering* from (1) to (4). Find the optimal number of clusters  $K^*$  such that the corresponding quality score  $\alpha_{K^*}$  is maximized. With this iterative algorithm, we can automatically group all the sequences into  $K^*$  clusters.

## 5 Experimental Evaluation

In order to evaluate the performance of our proposed algorithm, experiments were carried out on both synthetic data and real data from a wireless LAN environment. For comparison, four different clustering approaches were used as the baselines. The first one is referred to as K-Means (Loglik). This algorithm differs from our HMM-spectral algorithm in that, after the similarity matrix  $S$  is built, a standard  $k$ -means algorithm rather than a spectral algorithm is applied for clustering. This baseline is used to test the ability of our algorithm in reducing noise and dimensionality. The other three approaches are used to test the sensitivity of our algorithm against different initialization procedures used to train a mixture of HMMs for clustering when the EM algorithm is applied. The first one uses a random initialization, which is called MHMMs (Random). The second one trains a mixture model using a clustering-based initialization in “log-likelihood space” as given in [15]. In this approach, after computing the log-likelihood distance matrix  $L$ , the authors computed a different similarity matrix

$$S'(Y_i, Y_j) = \frac{1}{2}(L(Y_j; \lambda_i) + L(Y_i; \lambda_j)). \quad (9)$$

Then a  $k$ -means algorithm is used to cluster the sequences into  $K$  groups which are subsequently used to initialize the

mixture of HMMs. We call this method EHMMs (Loglik). The third algorithm is called EHMMs (DTW), which initializes the mixture of HMMs based on the clustering results using a DTW-based similarity measure [13].

In summary, we compare the performance of five clustering algorithms: (1) K-Means (Loglik), (2) EHMMs (Random), (3) EHMMs (Loglik), (4) EHMMs (DTW), and (5) HMM-Spectral (our proposed algorithm). In the following, we first introduce the criterion used to evaluate the clustering results in Section 5.1. Based on this criterion, we compare the performance of the five algorithms on synthetic data in Section 5.2. We then demonstrate the effectiveness of our HMM-Spectral clustering algorithm on real data collected from a wireless LAN environment in Section 5.3.

### 5.1 Evaluation Criterion

The criterion used for testing the validity of the clustering algorithms is the F-measure, which combines the concept of recall and precision measures in information retrieval area [8]. Specifically, for each actual cluster  $i$ , we first compute the recall and precision measures of all the detected clusters  $j$ . The definitions of recall and precision are given as  $Recall(i, j) = n_{ij}/n_i$  and  $Precision(i, j) = n_{ij}/n_j$ , where  $n_i$  is the number of sequences belonging to cluster  $i$ ,  $n_j$  is the number of sequences belonging to cluster  $j$ , and  $n_{ij}$  is the number of sequences in cluster  $i$  that are correctly identified in cluster  $j$ . Based on the recall and the precision measures, the F-measure of two clusters  $i$  and  $j$  is defined as

$$F(i, j) = \frac{2 \times Recall(i, j) \times Precision(i, j)}{Recall(i, j) + Precision(i, j)}. \quad (10)$$

The overall F-measure of the final clustering result is computed as a weighted sum over all the values of  $F(i, j)$ , which is defined as follows:

$$F = \sum_i \frac{n_i}{n} \max_j \{F(i, j)\}, \quad (11)$$

where  $n$  is the total number of sequences. For a particular cluster  $i$ , the max operation is taken over all the detected clusters  $j$ . In the following, we use F-measure to evaluate the performance of clustering.

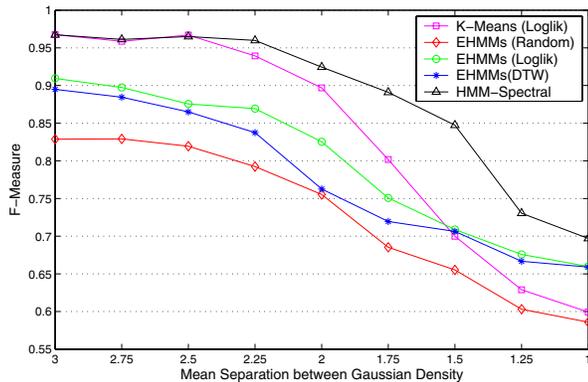
### 5.2 Experiments on Synthetic Data

In this experiment, 80 sequences of an average length of 180 ranging from 150 to 200 are generated from a 2-component HMM mixture (40 sequences from each component). Each observation consists of a one-dimensional value (in the next section, we extend to multi-dimensional vector sequences). Both HMMs are modeled with two states, which use a one-dimensional Gaussian observation density

for generating observations in each state. Similar to [1], we model the amount of noise by varying the amount of overlap between the generative models and by varying the mean separation between the Gaussian densities of the two states. The variations are similar for both HMMs. Specifically, the transition matrices for two HMMs are as follows:

$$A_1 = \begin{pmatrix} 0.7 & 0.3 \\ 0.4 & 0.6 \end{pmatrix}, \quad A_2 = \begin{pmatrix} 0.3 & 0.7 \\ 0.6 & 0.4 \end{pmatrix}.$$

We kept the variances of the Gaussian densities for two states as  $\sigma_1^2 = \sigma_2^2 = 1$ , and the mean of the first state as  $\mu_1 = 0$ . Then we varied the mean of the second state  $\mu_2$  in a range from 1 to 3. This corresponds to a change in the mean separation  $\frac{\Delta\mu}{\sigma_2}$  between the two Gaussian densities. This clustering task is non-trivial both because the data have exactly the same marginal statistics if the temporal information is removed from the sequences, and because the Markov dynamics governed by  $A_1$  and  $A_2$  are relatively similar for each HMM.



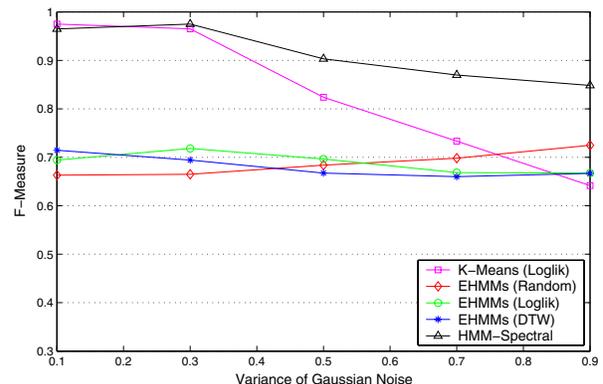
**Figure 2. Comparison of clustering algorithms vs. Mean separation between the observation Gaussian densities**

For each value of  $\frac{\Delta\mu}{\sigma_2}$ , the five clustering algorithms were applied to the generated sequences with 20 trials. Figure 2 shows the clustering results with respect to different values of the mean separation between two Gaussian densities. Each value plotted in this figure corresponds to the overall F-measure averaged over 20 trials. We can see from the figure that, as the mean separation between the two Gaussian densities in the two states decreases, that is, when the noise level increases, the overall performance of all the clustering algorithms decreases. This occurs because, when the mean separation becomes smaller, the two Gaussian densities become more indistinguishable from each other and thus more noise is involved in the observations. Consequently, the clustering task becomes more difficult.

Let us look into Figure 2 in detail. First, when the mean separation is small, our HMM-Spectral algorithm significantly outperforms the K-Means (Loglik) algorithm. This

indicates that, by performing eigendecomposition, HMM-Spectral can reduce the uncertainty of the similarity matrix, which enhances the quality of the similarity-based clustering. Second, our HMM-Spectral algorithm consistently outperforms the other three EM-based clustering algorithms. EHMMs (Random) gives the poorest clustering result because the convergence of the EM algorithm is affected by the random initialization to a large extent. By applying better initialization methods, EHMMs (Loglik) and EHMMs (DTW) improve the performance of EHMMs (Random), whereas their performance is roughly comparable. In summary, our HMM-Spectral algorithm performs the best among the five clustering algorithms.

We performed another set of experiments to evaluate the sensitivity of the five clustering algorithms to the observation noise. In this experiment, we kept the means of two Gaussian densities as  $\mu_1 = 0$  and  $\mu_2 = 1.5$ , and kept the variances as  $\sigma_1^2 = 1$  and  $\sigma_2^2 = 1.5$ . We set the transition matrices  $A_1$  and  $A_2$  for each HMM to be uniform. Since the observations in temporal sequences are not usually independent, we added Gaussian noise to the observations, whereby we set the mean  $\mu' = 0$  and  $\sigma'^2$  to vary from 0.1 to 0.9. For each setting, we also generated 80 sequences of average length 200 from the 2-component HMM mixture.



**Figure 3. Comparison of clustering algorithms vs. Variance of Gaussian noise**

Figure 3 shows the clustering results with respect to different values of the variance  $\sigma'^2$  of the Gaussian noise. As we can see from the figure, the three EHMM algorithms performed poorly when the Gaussian noise is added into the consecutive observations. Since HMMs assume that the observations are independent of each other over time, the three model-based algorithms cannot accurately estimate the parameters of the mixture model. In contrast, K-Means (Loglik) and HMM-Spectral performed better in the case of Gaussian noise. However, when more noise is involved in the observations, our HMM-Spectral algorithm outperforms K-Means (Loglik) because it can effectively reduce the noise through spectral clustering.

### 5.3 Experiments on Real Sensory Trajectory Data

In this experiment, we clustered the signal sequences obtained from a moving object's trajectory data in a wireless LAN environment as shown in Figure 4. This environment is the office area of the Computer Science Department at the Hong Kong University of Science and Technology. This area is equipped with an IEEE 802.11b wireless network in the 2.4 GHz frequency bandwidth. A user's activities are carried out in the three main areas (Office, Room1 and Room2), three entrances and seven hallways. The two rooms provide facilities for printing services and holding seminars. In the figure, four access points (APs) are marked with double solid circles, each of which is identified by its unique Media Access Control (MAC) address.

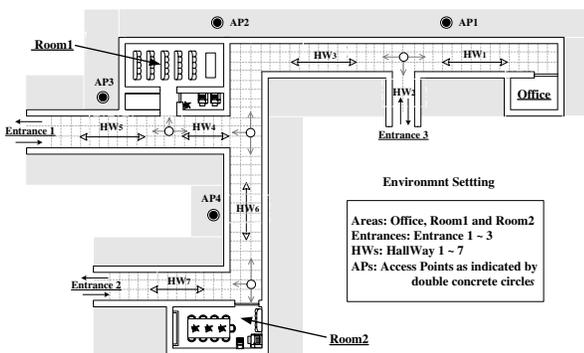


Figure 4. The layout of office area

While a user with a mobile device performs different activities in this environment, the mobile device can periodically record signal-strength measurements from the APs. For illustration, an observation  $o = \langle (AP_1 : -81)(AP_2 : -77)(AP_3 : -64)(AP_4 : -41) \rangle$  is a signal vector where each element consists of the MAC address of an AP and the corresponding signal-strength value. Accordingly, the observed sequence on users' behavior is represented as a sequence of signal-strength measurements recording their movements in the environment. These sequences contain lots of noise due to the multi-path fading effect in signal propagation, and are of different lengths.

Using the device driver and API that we have developed, we collected 180 sequences of a professor's 6 different activities in this office area. These activities include "Entrance1-to-Office", "Office-to-Room2", "Office-to-Entrance2", "Office-to-Entrance3", and "Entrance2-to-Room1". The number of sequences for each activity is 30. In total, 25 APs can be detected: four are distributed within this area; the others are distributed in adjacent areas on the same or different floors. In our experiment, we chose six APs because their signals occurred frequently and their average signal-strength values are the strongest. Therefore, each sequence is a 6-dimensional time series. The average

length of sequences is 122, ranging from 80 to 140 (within sequence lengths between 80-100, 101-120, 121-140, the mean lengths are 92.34, 116.08, 128.79, respectively, and the variances are 12.3, 4.7, 26.3, respectively). For evaluation purpose, we manually labeled each sequence by its corresponding intended activity. These labels serve as ground truth for evaluating the clustering results in terms of the F-measure, as well as for evaluating the automatically determined number of clusters against the number of different user activities. The task of performing clustering on this data set is difficult because signal-strength measurements are extremely noisy and uncertain in an indoor environment.

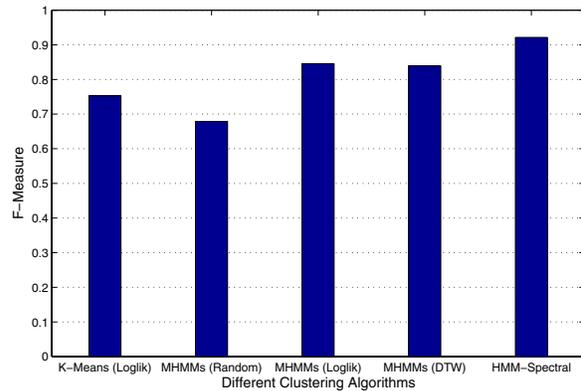


Figure 5. Comparison of clustering algorithms on real sensory trajectory data

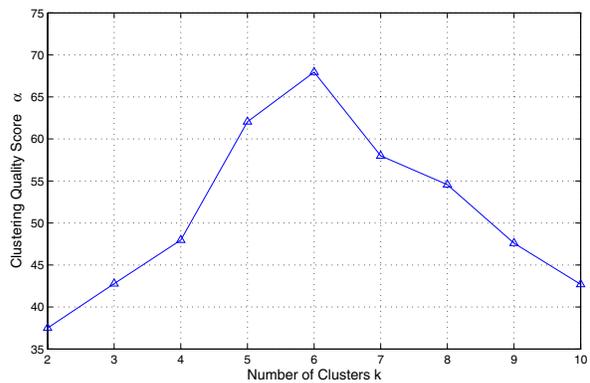


Figure 6. Clustering quality score  $\alpha$  vs. Number of clusters

We applied the five clustering algorithms to this data set. The clustering results are summarized in Figure 5. Each value of F-measure plotted in the figure is also the average of 20 trials. We can see from the figure that our HMM-Spectral algorithm performs the best on this real data set. Compared with the K-Means (Loglik) algorithm, the performance of HMM-Spectral is much better. This shows that HMM-Spectral is effective in reducing the uncertainty involved in the similarity matrix used for further clustering.

HMM-Spectral also outperforms the other three EHMMs algorithms because the EM-based algorithms can be easily trapped in a local minimum when lots of noise exists.

Experiments were also carried out to test the ability of our HMM-Spectral algorithm to determine the optimal number of clusters. Figure 6 shows different values of the quality score  $\alpha$  by varying the number of clusters. As we can see from the figure, the maximum value of  $\alpha$  is achieved when the number of clusters is equal to six, which is the same as our ground truth from the human labeled data set.

## 6 Conclusions and Future Work

In this paper, we investigated the problem of clustering variable length, noisy and multi-dimensional time series data. These data are abound in tracking and monitoring applications in a sensor network. We argued that traditional clustering algorithms have difficulty in dealing with these data due to the simplicity of their assumptions. We introduced the HMM-based mixture models for transforming the time series data into equal length vectors, which are in turn used to produce an affinity similarity matrix. We then explored how to apply spectral clustering on this matrix to further remove noise and obtain the final clusters. Our experimental results on both synthetic and real data demonstrated that our proposed HMM-Spectral algorithm is both robust and accurate for noisy data clustering.

In the future, we plan to apply other types of generative models to replace the HMM model. For example, we might apply a linear dynamic model for this purpose. In addition, we plan to test the utility of other clustering algorithms to replace the  $k$ -means algorithm in the HMM-Spectral algorithm on the projected eigenspace to find the best combination of algorithms for different types of time series data.

## Acknowledgment

We would like to thank Hong Kong RGC for supporting this work under grant CA 03/04.EG01 (HKBU 2/03C).

## References

- [1] J. Alon, S. Sclaroff, G. K. G., and V. Pavlovic. Discovering clusters in motion time-series data. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 375–381, Madison, Wisconsin, USA, June 2003.
- [2] L. V. Cadez, S. Gaffney, and P. Smyth. A general probabilistic framework for clustering individuals and objects. In *Proceedings of sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 140–149, Boston, MA, USA, August 2000.
- [3] G. Czielnik, M. Bennewitz, and W. Burgard. Where is ... ? learning and utilizing motion patterns of persons with mobile robots. In *Proceedings of International Joint Conference on Artificial Intelligence*, pages 909–914, Acapulco, Mexico, August 2003.
- [4] M. B. Eisen, P. T. Spellman, P. O. Brown, and D. Bostein. Cluster analysis and display of genome-wide expression patterns. *Proceedings of the National Academy of Sciences of the United States of America*, 95(25):14863–14868, December 1998.
- [5] S. Gaffney and P. Smyth. Trajectory clustering with mixtures of regression models. In *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 63–72, San Diego, CA, USA, August 1999.
- [6] A. K. Jain and R. C. Dubes. *Algorithms for Clustering Data*. Prentice Hall, Englewood Cliffs, NJ, USA, 1988.
- [7] E. Keogh and S. Kasetty. On the need for time series data mining benchmarks: A survey and empirical demonstration. In *Proceedings of the eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 102–111, Edmonton, Alberta, Canada, July 2002.
- [8] B. Larsen and C. Aone. Fast and effective text mining using linear-time document clustering. In *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 16–22, San Diego, CA, USA, August 1999.
- [9] C. Li and G. Biswas. A Bayesian approach to temporal data clustering using hidden Markov models. In *Proceedings of the International Conference on Machine Learning*, pages 543–550, Stanford, CA, USA, June 2000.
- [10] M. Meilă and J. Shi. Learning segmentation by random walks. In T. K. Leen, T. G. Dietterich, and V. Tresp, editors, *Advances in Neural Information Processing Systems 13*, pages 873–879. MIT Press, Denver, CO, USA, 2000.
- [11] A. Y. Ng, M. I. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In T. Dietterich, S. Becker, and Z. Ghahramani, editors, *In Advances in Neural Information Processing Systems 14*, pages 849–856. MIT Press, Vancouver, British Columbia, Canada, December 2001.
- [12] T. Oates, L. Firoiu, and R. P. Cohen. Using dynamic time warping to bootstrap hmm-based clustering of time series. *Sequence Learning: Paradigms, Algorithms and Applications*, pages 35–52, 2001.
- [13] T. Oates, M. D. Schmill, and P. R. Cohen. A method for clustering the experiences of a mobile robot that accords with human judgements. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence*, pages 846–851, Austin, Texas, USA, August 2000.
- [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] P. Smyth. Clustering sequences with hidden Markov models. In M. C. Mozer, M. I. Jordan, and T. Petsche, editors, *Advances in Neural Information Processing System 9*, pages 648–654. The MIT Press, 1997.
- [16] Y. Xiong and D. Yeung. Mixtures of ARMA models for model-based time series clustering. In *Proceedings of the 2002 IEEE International Conference on Data Mining*, pages 717–720, Maebashi City, Japan, December 2002.