Spectral Clustering for Time Series

Fei Wang[1] and Changshui Zhang[1]

State Key Laboratory of Intelligent Technology and Systems,
Department of Automation, Tsinghua University, Beijing 100084, P.R.China
feiwang03@mails.tsinghua.edu.cn
zcs@mail.tsinghua.edu.cn

Abstract. This paper presents a general framework for time series clustering based on spectral decomposition of the affinity matrix. We use the Gaussian function to construct the affinity matrix and develop a gradient based method for self-tuning the variance of the Gaussian function. The feasibility of our method is guaranteed by the theoretical inference in this paper. And our approach can be used to cluster both constant and variable length time series. Further our analysis shows that the cluster number is governed by the eigenstructure of the normalized affinity matrix. Thus our algorithm is able to discover the optimal number of clusters automatically. Finally experimental results are presented to show the effectiveness of our method.

1 Introduction

The recent years have seen a surge of interest in time series clustering. The high dimensionality and irregular lengths of the time sequence data pose many challenges to the traditional clustering algorithms. For example, it is hard for the application of the k-means algorithm [1] since we cannot define the "mean" of time series with different length. Many researchers propose to use hierarchical agglomerative clustering (HAC) for time series clustering [2][3], but there are two main drawbacks of these methods. On one hand, it is difficult for us to choose a proper distance measure when we merge two clusters; on the other hand, it is hard to decide when to stop the clustering procedure, that is, to decide the final cluster number.

Recently Porikli [4] proposed to use HMM parameter space and eigenvector decomposition to cluster time series, however, they didn't give us the theoretical basis of their method, and the variance of the Gaussian function they used to construct the affinity matrix was set empirically, which is usually not desirable.

To overcome the above problems, this paper presents a more efficient spectral decomposition based framework for time series clustering. Our method has four main advantages: (1) it is based on the similarity matrix of the dataset, that is, all it needs are just the pairwise similarities of the time series, so the high dimensionality of time series will not affect the efficiency of our approach; (2) it can be used to clustering time series with arbitrary length as long as the similarity measure between them is properly defined; (3) it can determine the optimal cluster number automatically; (4) it can self-tune the variance of the Gaussian kernel. The feasibility of our method has been proved theoretically in this paper, and many experiments are presented to show its effectiveness.

The remainder of this paper is organized as follows: we analyze and present our clustering framework in Section 2 in detail. In section 3 we will give a set of experiments, followed by the conclusions and discussions in section 4.

2 Spectral Clustering for Time Series

2.1 Theoretical Background

We will introduce the theoretical background of our spectral decomposition based clustering framework in this subsection. Given a set of time series $\{\mathbf{x}_i\}_{i=1}^M$ with the same length d, we form the data matrix $\mathbf{A} = [\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_M]$. A clustering process to $\mathbf{A}$ can result in $\mathbf{B} = \mathbf{AE} = [\mathbf{A}_1, \mathbf{A}_2, \cdots, \mathbf{A}_K]$, where $\mathbf{E}$ is a permutation matrix, $\mathbf{A}_i = [\mathbf{x}_{i1}, \mathbf{x}_{i2}, \cdots, \mathbf{x}_{is_i}]$ represents the i-th cluster, $\mathbf{x}_{ij}$ is the jth data in cluster i, and s_i is the number of data in the i-th cluster.

Now let's introduce some notes. The within-cluster scatter matrix of cluster k is $\mathbf{S}_w^k = \frac{1}{s_k}\sum_{s_i \in k}(\mathbf{x}_{s_i} - \mathbf{m}_k)(\mathbf{x}_{s_i} - \mathbf{m}_k)^T$, where $\mathbf{m}_k$ is the mean vector of the k-th cluster. The total within-cluster scatter matrix is $\mathbf{S}_w = \sum_{k=1}^K s_k \mathbf{S}_w^k$. The total between-cluster scatter matrix is $\mathbf{S}_b = \sum_{k=1}^K s_k(\mathbf{m}_k - \mathbf{m})(\mathbf{m}_k - \mathbf{m})^T$ and the total data scatter matrix is $\mathbf{T} = \mathbf{S}_b + \mathbf{S}_w = \sum_{i=1}^M (\mathbf{x}_i - \mathbf{m})(\mathbf{x}_i - \mathbf{m})^T$, where $\mathbf{m}$ is the sample mean of the whole dataset.

The goal of clustering is to achieve high within-cluster similarity and low between-cluster similarity, that is, we should minimize $trace(\mathbf{S}_w)$ and maximize $trace(\mathbf{S}_b)$. Since $\mathbf{T}$ is independent on the clustering results, then the maximization of $trace(\mathbf{S}_b)$ is equivalent to the minimization of $trace(\mathbf{S}_w)$. So our optimization object becomes

$$\min \quad trace(\mathbf{S}_w) \tag{1}$$

Since $\mathbf{m}_k = \mathbf{A}_k\mathbf{e}_k/s_k$, where $\mathbf{e}_k$ is the column vector containing s_k ones, then $\mathbf{S}_w^k = \frac{1}{s_k}\left(\mathbf{A}_k - \frac{\mathbf{A}_k\mathbf{e}_k}{s_k}\mathbf{e}_k^T\right)\left(\mathbf{A}_k - \frac{\mathbf{A}_k\mathbf{e}_k}{s_k}\mathbf{e}_k^T\right)^T = \frac{1}{s_k}\mathbf{A}_k\left(\mathbf{I}_k - \frac{\mathbf{e}_k\mathbf{e}_k^T}{s_k}\right)\mathbf{A}_k^T$, where $\mathbf{I}_k$ is the identity matrix of order s_k.

Let $\mathbf{J}_k = trace(\mathbf{S}_w^k) = trace\left(\frac{1}{s_k}\mathbf{A}_k\mathbf{A}_k^T\right) - trace\left(\frac{1}{s_k}\frac{\mathbf{e}_k^T}{\sqrt{s_k}}\mathbf{A}_k^T\mathbf{A}_k\frac{\mathbf{e}_k}{\sqrt{s_k}}\right)$. Define $\mathbf{J} = trace(\mathbf{S}_w) = \sum_{k=1}^K s_k trace(\mathbf{S}_w^k)$ and the block-diagonal matrix

$$\mathbf{Q} = \begin{pmatrix} \mathbf{e}_1/\sqrt{s_1} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{e}_1/\sqrt{s_1} & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{e}_K/\sqrt{s_K} \end{pmatrix} \tag{2}$$

then $\mathbf{J} = trace\left(\mathbf{B}\mathbf{B}^T\right) - trace\left(\mathbf{Q}^T\mathbf{B}^T\mathbf{B}\mathbf{Q}\right)$. Since $\mathbf{B} = \mathbf{AE}$ and $\mathbf{E}$ is a permutation matrix, it's straight forward to show that $trace\left(\mathbf{B}\mathbf{B}^T\right) = trace(\mathbf{A}^T\mathbf{A})$,

$trace(\mathbf{Q}^T\mathbf{B}^T\mathbf{B}\mathbf{Q}) = trace(\tilde{\mathbf{Q}}^T\mathbf{A}^T\mathbf{A}\tilde{\mathbf{Q}})$, where $\tilde{\mathbf{Q}} = \mathbf{E}\mathbf{Q}$, that is, $\tilde{\mathbf{Q}}$ is equal to $\mathbf{Q}$ with some rows exchanged. Now we relax the constraint of $\tilde{\mathbf{Q}}$ to $\tilde{\mathbf{Q}}^T\tilde{\mathbf{Q}} = \mathbf{I}$ as in [5]. Then optimizer (1) is equivalent to

$$\max_{\tilde{\mathbf{Q}}^T\tilde{\mathbf{Q}}=\mathbf{I}} H = trace\left(\tilde{\mathbf{Q}}^T\mathbf{A}^T\mathbf{A}\tilde{\mathbf{Q}}\right) \tag{3}$$

which is a constrained optimization problem. It turns out the above optimization problem has a closed-form solution according to the following theorem[5].

Theorem(Ky Fan). *Let* $\mathbf{H}$ *be a symmetric matrix with eigenvalues*

$$\lambda_1 \geqslant \lambda_2 \geqslant \cdots \geqslant \lambda_n$$

and the corresponding eigenvectors $\mathbf{U} = [\mathbf{u}_1, \mathbf{u}_2, \cdots, \mathbf{u}_n]$. *Then*

$$\lambda_1 + \lambda_2 + \cdots + \lambda_k = \max_{\mathbf{X}^T\mathbf{X}=\mathbf{I}} trace(\mathbf{X}^T\mathbf{H}\mathbf{X})$$

Moreover, the optimal $\mathbf{X}^*$ *is given by* $\mathbf{X}^* = \mathbf{U} = [\mathbf{u}_1, \mathbf{u}_2, \cdots, \mathbf{u}_k]\mathbf{R}$, *with* $\mathbf{R}$ *an arbitrary orthogonal matrix.*

From the above theorem we can easily derive the solution to (3). The optimal $\tilde{\mathbf{Q}}$ can be obtained by taking the top K eigenvectors of $\mathbf{S} = \mathbf{A}^T\mathbf{A}$, and the sum of the corresponding largest K eigenvalues of $\mathbf{S}$ gives the optimal H.

The matrix $\mathbf{S}$ can be expanded as

$$\mathbf{S} = \mathbf{A}^T\mathbf{A} = [\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_M]^T[\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_M] = \begin{pmatrix} \mathbf{x}_1^T\mathbf{x}_1 & \mathbf{x}_1^T\mathbf{x}_2 & \cdots & \mathbf{x}_1^T\mathbf{x}_M \\ \mathbf{x}_2^T\mathbf{x}_1 & \mathbf{x}_2^T\mathbf{x}_2 & \cdots & \mathbf{x}_2^T\mathbf{x}_M \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{x}_M^T\mathbf{x}_1 & \mathbf{x}_M^T\mathbf{x}_2 & \cdots & \mathbf{x}_M^T\mathbf{x}_M \end{pmatrix}$$

Thus the (i, j)-th entry of $\mathbf{S}$ is the inner product of $\mathbf{x}_i$ and $\mathbf{x}_j$, which can be used to measure the similarity between them. Then $\mathbf{S}$ can be treated as the similarity matrix of the dataset $\mathbf{A}$. Moreover, we can generalize this idea and let the entries of $\mathbf{S}$ be some other similarity measure, as long as it satisfies the symmetry and positive semidefinite properties. Since there has been so many methods for measuring the similarities between time series with different length (for a comprehensive study, see [7]), we can drop the assumption at the beginning of this section that all the time series have the same length d and let the entries in $\mathbf{S}$ be some similarity measure that can measure the similarity of time series with arbitrary length. Then our method can be used to cluster time series with any length.

2.2 Estimating the Number of Clusters Automatically

In order to estimate the optimal number of the clusters, we first normalize the rows of $\mathbf{S}$, that is, define $\mathbf{U} = diag(u_{11}, u_{22}, \cdots, u_{MM})$, where $u_{ii} = \sum_{j=1}^{M}\mathbf{S}_{ij}$, then our normalization makes $\mathbf{S}' = \mathbf{U}^{-1}\mathbf{S}$.

In the ideal case, $\mathbf{S}(i,j) = 0$ if $\mathbf{x}_i$ and $\mathbf{x}_j$ belong to different clusters. We assume that the data objects are ordered by clusters, that is $\mathbf{A} = [\mathbf{A}_1, \cdots, \mathbf{A}_K]$, where $\mathbf{A}_i$ represents the data in cluster i. Thus the similarity matrix $\mathbf{S}$ and normalized similarity matrix $\mathbf{S}'$ will become block-diagonal. It can be easily inferred that each diagonal block in $\mathbf{S}'$ has the largest eigenvalue 1 [8]. Therefore we can use the number of repeated eigenvalue 1 to estimate the number of clusters in the dataset. Moreover, Ng et al [9] told us that this conclusion can also be extended to the general cases through matrix perturbation theory.

In practice, since the similarity matrix may not be block-diagonal, we can choose the number of eigenvalues which are most close to 1. Therefore we can predefine a small threshold $\delta \in [0,1]$, and determine the number of clusters by count the eigenvalues λ_i which satisfy $|\lambda_i - 1| < \delta$.

2.3 Constructing the Affinity Matrix

Now the only problem remained for us is to construct a "good" similarity matrix which is almost block-diagonal. We use the Gaussian function $S_{ij} = exp\left(-\frac{d_{ij}^2}{2\sigma^2}\right)$ to construct it like in [4], where d_{ij} is some similarity measure between $\mathbf{x}_i$ and $\mathbf{x}_j$. To distinguish the transformed matrix $\mathbf{S}$ from the previously constructed similarity matrix $\mathbf{D}$, we will call $\mathbf{S}$ affinity matrix throughout the paper. The diagonal elements of the affinity matrix are set to zero as in [9]. A gradient ascent method is used to determine the parameter σ^2.

More precisely, assume the solution of the optimizer (3) is $\tilde{\mathbf{Q}}^* = [\mathbf{q}_1, \cdots, \mathbf{q}_K]$, and $\mathbf{q}_i = (\mathbf{q}_i^1, \cdots, \mathbf{q}_i^M)^T \in \mathbb{R}^M$, where $\mathbf{q}_i^j$ represents the j-th element of the column vector $\mathbf{q}_i$. Then $H = trace\left(\tilde{\mathbf{Q}}^{*T}\mathbf{S}\tilde{\mathbf{Q}}^*\right) = \sum_{i=1}^{K} \mathbf{q}_i^T \mathbf{S}\mathbf{q}_i$, hence

$$H = \sum_{i=1}^{K}\sum_{j=1}^{M}\sum_{k=1}^{M} \mathbf{q}_i^j \mathbf{q}_i^k S_{jk} = \sum_{i=1}^{K}\sum_{j=1}^{M}\sum_{k=1}^{M} \mathbf{q}_i^j \mathbf{q}_i^k exp\left(-\frac{d_{jk}^2}{2\sigma^2}\right)$$

where S_{jk} is the (j,k)-th entry of the affinity matrix $\mathbf{S}$. If we treat H as a function of σ, then the gradient of H is

$$G = \frac{\partial H}{\partial \sigma} = \sum_{i=1}^{K}\sum_{j=1}^{M}\sum_{k=1}^{M} \mathbf{q}_i^j \mathbf{q}_j^k \frac{\partial S_{jk}}{\partial \sigma} = \sum_{i=1}^{K}\sum_{j=1}^{M}\sum_{k=1}^{M} \mathbf{q}_i^j \mathbf{q}_i^k \frac{d_{jk}^2}{\sigma^3} exp\left(-\frac{d_{jk}^2}{2\sigma^2}\right) \qquad (4)$$

Inspired by the work in [10], we propose a gradient based method to tune the variance of the Gaussian function. More precisely, we can first give an initial guess of σ, then use G to adjust it iteratively until $\|G\| < \varepsilon$. The detailed algorithm is shown in Table 1.

3 Experiments

In this section, we will give two experiments where we used our spectral decomposition based clustering framework to cluster time series. First we used a

Table 1. Clustering Time Series via Spectral Decomposition

```
Input: Dataset $\mathbf{X}$, Precision $\varepsilon$, Max iteration T, Initialize $\sigma$ to $\sigma_0$, Learning rate $\alpha$
Output: Clustering results.
1. Choose some similarity metric to construct the similarity matrix $\mathbf{D}$.
2. Initialize $\sigma$ to $\sigma_0$, construct the affinity matrix $\mathbf{S}$;
3. Calculate $\mathbf{S}'$ by normalizing $\mathbf{S}$, do spectral decomposition on it and find the
number of eigenvalues which are closest to 1, which corresponds to the optimal
number of clusters $K$
4. For i=1:T
    (a).Solve (3) to achieve $\tilde{\mathbf{Q}}^*$
    (b).Compute the gradient according to (4) $G^i$
    (c).If $\|G\| < \varepsilon$, break; else let $\sigma = \sigma + \alpha G^i$
5. Treat each column of the final $\tilde{\mathbf{Q}}^*$ as a new point in $\mathbb{R}^K$ and cluster them into
$K$ clusters via kmeans algorithm
```

synthetic dataset generated in the same way as in [2]. This is a two-class clustering problem. In our experiments, 40 time series are generated from each of the 2 HMMs. The length of these time series vary from 200 to 300. We use Both HMMs have two hidden states and use the same priors and observation parameters. The priors are uniform and the observation distribution is a univariate Gaussian with $\mu = 3$ and variance $\sigma^2 = 1$ for hidden state 1, and with mean $\mu = 0$ and variance $\sigma^2 = 1$ for hidden state 2. The transition matrices of them are $A_1 = \begin{pmatrix} 0.6 & 0.4 \\ 0.4 & 0.6 \end{pmatrix}$ and $A_2 = \begin{pmatrix} 0.4 & 0.6 \\ 0.6 & 0.4 \end{pmatrix}$. Fig.1 shows us two samples generated from these two HMMs, the left figure is a time series generated by the first HMM, and the right is generated by the second HMM.

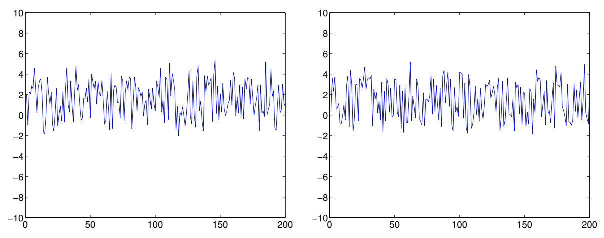

Fig. 1. Samples generated from different HMMs

From Fig.1 we cannot easily infer which sample is generated from which HMM. We measure the pairwise similarity of the time series by the BP metric [11] which is defined as follows.

Definition 1 (BP metric). *Suppose we train two HMMs λ_i and λ_j for time series $\mathbf{x}_i$ and $\mathbf{x}_j$ respectively. Let $L_{ij} = P(\mathbf{x}_j|\lambda_i)$ and $L_{ii} = P(\mathbf{x}_i|\lambda_i)$. Then the*

BP metric between $\mathbf{x}_i$ *and* $\mathbf{x}_j$ *is defined as*

$$L_{BP}^{ij} = \frac{1}{2}\left[\frac{L_{ij} - L_{ii}}{L_{ii}} + \frac{L_{ji} - L_{jj}}{L_{jj}}\right] \tag{5}$$

The reason why we used the BP metric here is because that it not only considers the likelihood of $\mathbf{x}_i$ under λ_j as usual[2], but also take into account the modeling goodness of $\mathbf{x}_i$ and $\mathbf{x}_j$ themselves. Thus it can be viewed as a relative normalized difference between the sequence and the training likelihoods[11].

After the similarity matrix $\mathbf{D}$ having been constructed by the BP metric, we will come to step 2 in Table 1. The initial variance σ_0 of the Gaussian function is set to 0.1. Fig. 2 shows the normalized affinity matrix and the corresponding top ten eigenvalues, from which we can see that our method is able to discover the correct cluster number 2 automatically.

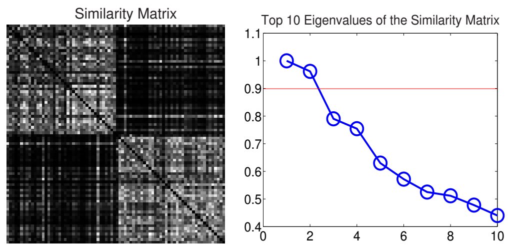

Fig. 2. Affinity matrix and the corresponding top 10 eigenvalues

We use clustering accuracy to evaluate the final clustering results as in [12]. More precisely, if we treat the cluster problem as a classification problem, then the clustering accuracy can be defined as follows.

Definition 2 (Clustering Accuracy). *Let* $\{t_i\}$ *denote the true classes and* $\{c_j\}$ *denote the clusters found by a cluster algorithm. We then label all the data in cluster* c_j *as* t_i *if they share the most data objects. Note that the number of clusters need not be the same as the number of classes. Then we can calculate the clustering accuracy* η *as* :

$$\eta = \frac{\sum_{\mathbf{x}} I(c_i(\mathbf{x}) = t_j(\mathbf{x}))}{M} \tag{6}$$

where $I(\cdot)$ *is the indicator function,* $c_i(\mathbf{x})$ *is the label of the cluster which* $\mathbf{x}$ *belongs to,* $t_j(\mathbf{x})$ *is the true class of* $\mathbf{x}$*, and* M *is the size of the dataset.*

Fig. 3 provides the results of our algorithm after 50 iterations (We don't use the termination condition in Table 1 step 4 (c)). In all these figures the horizontal axis corresponds to the iteration number. The vertical axis of these figures represents the clustering accuracy η in Eq.(6), gradient G in Eq.(4), and the object function value H in Eq.(3).

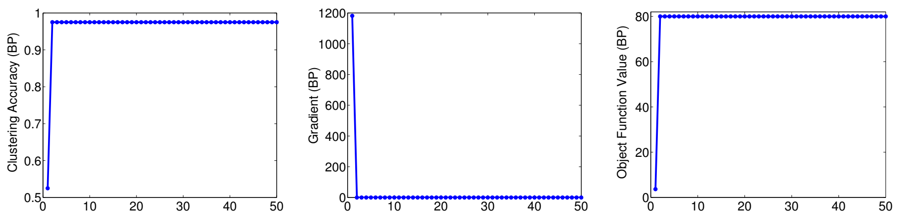

Fig. 3. Experimental results for Smyth's dataset

From Fig.3 we can see that as the iteration procedure goes deeply, the gradient G will become smaller while the the clustering accuracy and the object function value H are increasing. And our algorithm will converge after only two steps in this experiment.

We have compared the clustering accuracies resulted from our approach with the results achieved from hierarchical agglomerative clustering (HAC) methods, since most of the developed time series clustering approaches have adopted an HAC framework[2][3]. In an agglomerative fashion, the HAC method starts with M different clusters, each containing exactly one time sequence. Then the algorithm will merge the clusters continuously based on some similarity measure until the stopping condition is met. There are three kinds of HAC approaches according to the different similarity measure they use to merge clusters[1]. They are *Complete-linkage* $HAC(CHAC)$, *Single-linkage* $HAC(SHAC)$ and *Average-linkage* $HAC(AHAC)$, which adopt furthest-neighbor distance, nearest-neighbor distance and average-neighbor distance to measure the similarity between two clusters respectively. In our experiments, the final cluster number of all these HAC methods is set to 2 manually. The final clustering accuracies are shown in Table 2. From which we can see that our algorithm can perform better than HAC methods in this case study.

Table 2. Clustering accuracies on the synthetic dataset

	CHAC	SHAC	AHAC	Our method
BP	0.9500	0.5125	0.9625	0.9725

In the second experiment we use a real EEG dataset which is extracted from the 2nd Wadsworth BCI dataset in BCI2003 competition [13]. According to [13], the data objects can be generated from 3 classes: the EEG signals evoked by flashes containing targets,the EEG signals evoked by flashes adjacent to targets, and other EEG signals. All the data objects have an equal length 144. All the data objects have an equal length 144. Fig. 4 shows an example for each class.

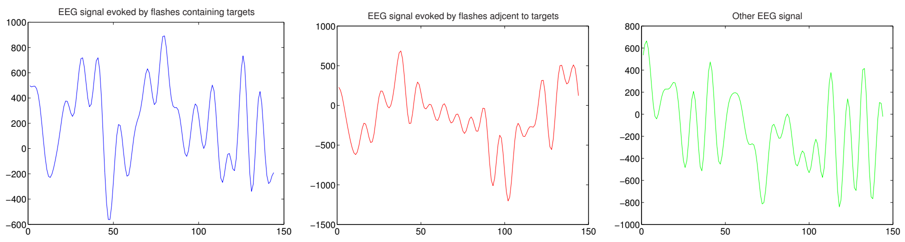

Fig. 4. EEG signals from the 2nd Wadsworth BCI Dataset

We randomly choose 50 EEG signals from each class. As all the time series have the same length, therefore we can use the Euclidean distance to measure the pairwise distances of the time series. The Euclidean distance between two time series can be defined as follows[14].

Definition 3 (Euclidean distance). *Assume time series* $\mathbf{x}_i$ *and* $\mathbf{x}_j$ *have the same length* l, *then the Euclidean distance between them is simply*

$$D_{ij}^{Euc} = \sqrt{\sum_{k=1}^{l}(\mathbf{x}_i^k - \mathbf{x}_j^k)^2} \tag{7}$$

where $\mathbf{x}_i^k$ *refers to the* $k-th$ *element of* $\mathbf{x}_i$.

In our experiments, we adopt both the Euclidean distance (7) and the BP metric (5)to construct the similarity matrix **D**. And apply the Gaussian function to transform them to the affinity matrices. The initial variance of the Gaussian function is set to 600. The final experimental results of our method after 50 iterations are presented in Fig. 5, where the first row shows the trends of clustering accuracy, gradient, and object function value achieved based on the Euclidean distance (ED) versus iteration, and the second row shows the trends of these indexes achieved based on the BP metric (BP) versus iteration.

Fig.5 shows us that the trend of these indexes are very similar with that in Fig.3. The clustering accuracy and objective function value are increasing and more and more stable with the decreasing of gradient.

We also compared the clustering accuracies achieved from *HAC* methods and our approach, the final cluster number of the *HAC* methods is also set to 3 manually. Table 3 gives us the final results. Each column in Table 3 shows the results of a method. The second and third rows are the final clustering accuracies when we use the Euclidean distance and the BP metric respectively to measure the similarity of pairwise time series in these methods.

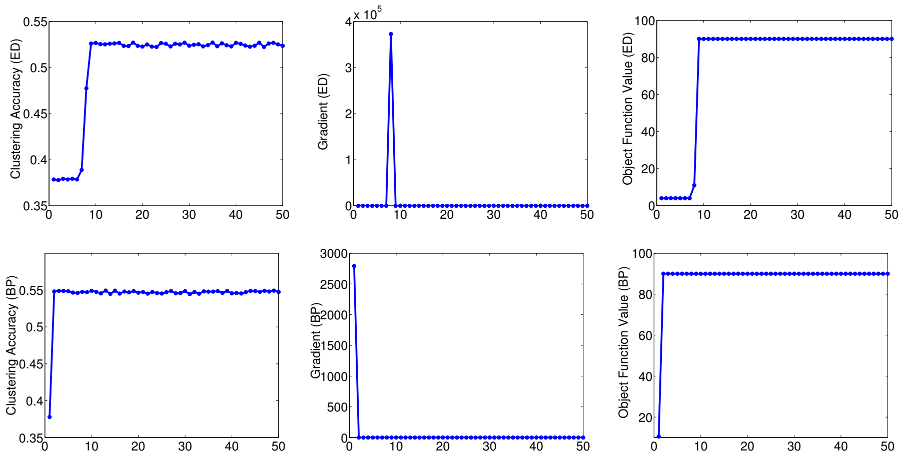

Fig. 5. Experimental results for EEG dataset

Table 3. Clustering results on *EEG* dataset

	CHAC	SHAC	AHAC	Our method
Euclidean	0.4778	0.3556	0.3556	0.5222
BP	0.4556	0.3556	0.4222	0.5444

From the above experiments we can see that for HAC methods, if we adopt different distance measures (nearest, furthest, average neighbor distance), the final clustering results may become dramatically different. Moreover, these methods may always fail to find the correct cluster number, which makes us to set this number manually. On the contrary, our spectral decomposition based clustering method can discover the right cluster number automatically, and in most cases the final performance achieved by our approach will be better than HAC methods.

4 Conclusion and Discussion

In this paper we present a new spectral decomposition based time series clustering framework. The theoretical analysis guarantees the feasibility of our approach, and the effectiveness of which has been shown by experiments.

The problem remained for us is that the spectral decomposition is time consuming. Fortunately the affinity matrix is Hermitian, and usually sparse. Thus we can use the subspace method like the Lanczos method, Arnoldi method to solve the large eigenproblems [6]. We believe that our approach will be promising and it may have potential usage in many data mining problems.

Acknowledgements

This work is supported by the project (60475001) of the National Natural Science Foundation of China.

References

1. R. O. Duda, P. E. Hart, D. G. Stork. Pattern Classification. 2nd edition. New York, John Wiley & Sons, Inc. 2001.
2. P. Smyth. Clustering Sequences with Hidden Markov Models. In Advances in Neural Information Processing 9 (NIPS'97), MIT Press, 1997. 1997.
3. S. Zhong, J. Ghosh. A Unified Framework for Model-based Clustering. Journal of Machine Learning Research 4 (2003) 1001-1037. 2003.
4. F. M. Porikli. Clustering Variable Length Sequences by Eigenvector Decomposition Using Hmm. International Workshop on Structural and Syntactic Pattern Recognition, (SSPR'04). 2004.
5. H. Zha, X. He, C. Ding, H. Simon, M. Gu. Spectral Relaxation for K-means Clustering. Advances in Neural Information Processing Systems 14 (NIPS'01). pp. 1057-1064, Vancouver, Canada. 2001.
6. G. H. Golub, C. F. Van Loan. Matrix Computation. 2nd ed. Johns Hopkins University Press, Baltimore. 1989.
7. G. Das, D. Gunopulos, and H. Mannila. Finding Similar Time Series, In proceedings of the First European Symposium on Principles of Data Mining and Knowledge Discovery, LNCS 1263, pp. 88-100. 1999.
8. M. Maila and J. Shi. A Random Walks View of Spectral Segmentation. International Workshop on AI and STATISTICS (AISTATS) 2001. 2001.
9. A. Y. Ng, M. I. Jordan, Y. Weiss. On Spectral Clustering: Analysis and an Algorithm. In Advances in Neural Information Processing Systems 14 (NIPS'01), pages 849–856, Vancouver, Canada, MIT Press. 2001.
10. J. Huang, P. C. Yuen, W. S. Chen and J. H. Lai. Kernel Subspace LDA with Optimized Kernel Parameters on Face Recognition. Proceedings of the Sixth IEEE International Conference on Automatic Face and Gesture Recognition (FGR04). 2004.
11. A. Panuccio, M. Bicego, V. Murino. A Hidden Markov Model-based approach to sequential data clustering. Structural, Syntactic and Statistical Pattern Recognition (SSPR'02), LNCS 2396. 2002.
12. F. Wang, C. Zhang. Boosting GMM and Its Two Applications. To be appeared in the 6th International Workshop on Multiple Classifier Systems (MCS'05). 2005.
13. Z. Lin, C. Zhang, Enhancing Classification by Perceptual Characteristic for the P300 Speller Paradigm. In Proceedings of the 2nd International IEEE EMBS Special Topic Conference on Neural Engineering (NER'05). 2005.
14. R. Agrawal, C. Faloutsos, A. Swami. Efficient Similarity Search In Sequence Databases. Proceedings of the 4th International Conference of Foundations of Data Organization and Algorithms (FODO'93). 1993.