Evolutionary Spectral Clustering by Incorporating Temporal Smoothness

[†]Yun Chi [†]Xiaodan Song [‡]Dengyong Zhou[*] [†]Koji Hino [†]Belle L. Tseng
[†]NEC Laboratories America, 10080 N. Wolfe Rd, SW3-350, Cupertino, CA 95014, USA
[‡]NEC Laboratories America, 4 Independence Way, Princeton, NJ 08540, USA
[†]{ychi,xiaodan,hino,belle}@sv.nec-labs.com, [‡]dzhou@nec-labs.com

ABSTRACT

Evolutionary clustering is an emerging research area essential to important applications such as clustering dynamic Web and blog contents and clustering data streams. In evolutionary clustering, a good clustering result should fit the current data well, while simultaneously not deviate too dramatically from the recent history. To fulfill this dual purpose, a measure of *temporal smoothness* is integrated in the overall measure of clustering quality. In this paper, we propose two frameworks that incorporate temporal smoothness in evolutionary spectral clustering. For both frameworks, we start with intuitions gained from the well-known k-means clustering problem, and then propose and solve corresponding cost functions for the evolutionary spectral clustering problems. Our solutions to the evolutionary spectral clustering problems provide more stable and consistent clustering results that are less sensitive to short-term noises while at the same time are adaptive to long-term cluster drifts. Furthermore, we demonstrate that our methods provide the optimal solutions to the relaxed versions of the corresponding evolutionary k-means clustering problems. Performance experiments over a number of real and synthetic data sets illustrate our evolutionary spectral clustering methods provide more robust clustering results that are not sensitive to noise and can adapt to data drifts.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications—*Data mining*; H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*Information filtering*

General Terms

Algorithms, Experimentation, Measurement, Theory

[*]Current address for this author:
Microsoft Research, One Microsoft Way, Redmond, WA 98052, email: dengyong.zhou@microsoft.com

Keywords

Evolutionary Spectral Clustering, Temporal Smoothness, Preserving Cluster Quality, Preserving Cluster Membership, Mining Data Streams

1. INTRODUCTION

In many clustering applications, the characteristics of the objects to be clustered change over time. Very often, such characteristic change contains both long-term trend due to concept drift and short-term variation due to noise. For example, in the blogosphere where blog sites are to be clustered (e.g., for community detection), the overall interests of a blogger and the blogger's friendship network may drift slowly over time and simultaneously, short-term variation may be triggered by external events. As another example, in an ubiquitous computing environment, moving objects equipped with GPS sensors and wireless connections are to be clustered (e.g., for traffic jam prediction or for animal migration analysis). The coordinate of a moving object may follow a certain route in the long-term but its estimated coordinate at a given time may vary due to limitations on bandwidth and sensor accuracy.

These application scenarios, where the objects to be clustered evolve with time, raise new challenges to traditional clustering algorithms. On one hand, the current clusters should depend mainly on the current data features — aggregating all historic data features makes little sense in non-stationary scenarios. On the other hand, the current clusters should not deviate too dramatically from the most recent history. This is because in most dynamic applications, we do not expect data to change too quickly and as a consequence, we expect certain levels of *temporal smoothness* between clusters in successive timesteps. We illustrate this point by using the following example. Assume we want to partition 5 blogs into 2 clusters. Figure 1 shows the relationship among the 5 blogs at time *t-1* and time *t*, where each node represents a blog and the numbers on the edges represent the similarities (e.g., the number of links) between blogs. Obviously, the blogs at time *t-1* should be clustered by Cut I. The clusters at time t are not so clear. Both Cut II and Cut III partition the blogs equally well. However, according to the principle of temporal smoothness, Cut III should be preferred because it is more consistent with recent history (time *t-1*). Similar ideas have long been used in time series analysis [5] where moving averages are often used to smooth out short-term fluctuations. Because similar short-term variances also exist in clustering applications, either due to data noises or due to non-robust behaviors

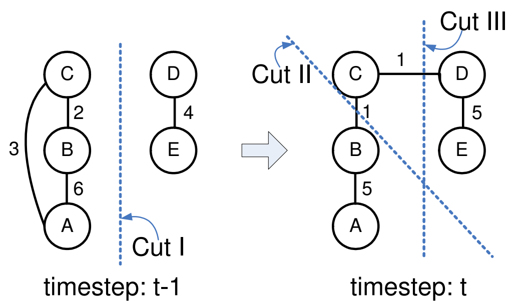

Figure 1: An evolutionary clustering scenario

of clustering algorithms (e.g., converging to different locally suboptimal modes), new clustering techniques are needed to handle evolving objects and to obtain stable and consistent clustering results.

In this paper, we propose two *evolutionary spectral clustering algorithms* in which the clustering cost functions contain terms that regularize temporal smoothness. Evolutionary clustering was first formulated by Chakrabarti et al. [3] where they proposed heuristic solutions to evolutionary hierarchical clustering problems and evolutionary k-means clustering problems. In this paper, we focus on evolutionary *spectral* clustering algorithms under a more rigorous framework. Spectral clustering algorithms have solid theory foundation [6] and have shown very good performances. They have been successfully applied to many areas such as document clustering [22, 15], imagine segmentation [19, 21], and Web/blog clustering [9, 18]. Spectral clustering algorithms can be considered as solving certain graph partition problems, where different graph-based measures are to be optimized. Based on this observation, we define the cost functions in our evolutionary spectral clustering algorithms by using the graph-based measures and derive corresponding (relaxed) optimal solutions. At the same time, it has been shown that these graph partition problems have close connections to different variation of the k-means clustering problems. Through these connections, we demonstrate that our evolutionary spectral clustering algorithms provide solutions to the corresponding evolutionary k-means clustering problems as special cases.

In summary, our main contributions in this paper can be summarized as the following:

1. We propose two frameworks for evolutionary spectral clustering in which the temporal smoothness is incorporated into the overall clustering quality. To the best of our knowledge, our frameworks are the first evolutionary versions of the spectral clustering algorithms.

2. We derive optimal solutions to the *relaxed* versions of the proposed evolutionary spectral clustering frameworks. Because the unrelaxed versions are shown to be NP-hard, our solutions provide both the practical ways of obtaining the final clusters and the upper bounds on the performance of the algorithms.

3. We also introduce extensions to our algorithms to handle the case where the number of clusters changes with time and the case where new data points are inserted and old ones are removed over time.

1.1 Related Work

As stated in [3], evolutionary clustering is a fairly new topic formulated in 2006. However, it has close relationships with other research areas such as clustering data streams, incremental clustering, and constrained clustering.

In clustering data streams, large amount of data that arrive at high rate make it impractical to store all the data in memory or to scan them multiple times. Under such a new data model, many researchers have investigated issues such as how to efficiently cluster massive data set by using limited memory and by one-pass scanning of data [12], and how to cluster evolving data streams under multiple resolutions so that a user can query any historic time period with guaranteed accuracy [1]. Clustering data stream is related to our work in that data in data streams evolve with time. However, instead of the scalability and one-pass-access issues, we focus on how to obtain clusters that evolve smoothly over time, an issue that has not been studied in the above works.

Incremental clustering algorithms are also related to our work. There exists a large research literature on incremental clustering algorithms, whose main task is to efficiently apply dynamic updates to the cluster centers [13], medoids [12], or hierarchical trees [4] when new data points arrive. However, in most of these studies, newly arrived data points have no direct relationship with existing data points, other than that they probably share similar statistical characteristics. In comparison, our study mainly focuses on the case when the similarity among *existing* data points varies with time, although we can also handle insertion and removal of data points over time. In [16], an algorithm is proposed to cluster moving objects based on a novel concept of micro-clustering. In [18], an incremental spectral clustering algorithm is proposed to handles similarity changes among objects that evolve with time. However, the focus of both [16] and [18] is to improve computation efficiency at the cost of lower cluster quality.

There is also a large body of work on constrained clustering. In these studies, either hard constraints such as *cannot links* and *must links* [20] or soft constraints such as prior preferences [15] are incorporated in the clustering task. In comparison, in our work the constraints are not given *a priori*. Instead, we set our goal to optimize a cost function that incorporates temporal smoothness. As a consequence, some soft constraints are automatically implied when historic data and clusters are connected with current ones.

Our work is especially inspired by the work by Chakrabarti et al. [3], in which they propose an evolutionary hierarchical clustering algorithm and an evolutionary k-means clustering algorithm. We mainly discuss the latter because of its connection to spectral clustering. Chakrabarti et al. proposed to measure the temporal smoothness by a distance between the clusters at time t and those at time *t-1*. Their cluster distance is defined by (1) pairing each centroid at t to its nearest peer at *t-1* and (2) summing the distances between all pairs of centroids. We believe that such a distance has two weak points. First, the pairing procedure is based on heuristics and it could be unstable (a small perturbation on the centroids may change the pairing dramatically). Second, because it ignores the fact that the *same* data points are to be clustered in both t and *t-1*, this distance may be sensitive to the movement of data points such as shifts and rotations (e.g., consider a fleet of vehicles that move together while the relative distances among them remain the same).

2. NOTATIONS AND BACKGROUND

First, a word about notation. Capital letters, such as W and Z, represent matrices. Lower case letters in vector forms, such as $\vec{v}_i$ and $\vec{\mu}_l$, represent column vectors. Scripted letters, such as $\mathcal{V}$ and $\mathcal{V}_p$, represent sets. For easy presentation, for a given variable, such as W and $\vec{v}_i$, we attach a subscript t, i.e., W_t and $\vec{v}_{i,t}$, to represent the value of the variable at time t. And we use $Tr(W)$ to represent the *trace* of W where $Tr(W) = \sum_i W(i,i)$. In addition, for a matrix $X \in \mathcal{R}^{n \times k}$, we use $span(X)$ to represent the subspace spanned by the columns of X. For vector norms we use the Euclidian norm and for matrix norms we use the Frobenius norm, i.e., $\|W\|^2 = \sum_{i,j} W(i,j)^2 = Tr(W^T W)$.

2.1 The clustering problem

We state the clustering problem in the following way. For a set $\mathcal{V}$ of n nodes, a clustering result is a partition $\{\mathcal{V}_1, \ldots, \mathcal{V}_k\}$ of the nodes in $\mathcal{V}$ such that $\mathcal{V} = \cup_{l=1}^{k} \mathcal{V}_l$ and $\mathcal{V}_p \cap \mathcal{V}_q = \emptyset$ for $1 \le p, q \le k, p \ne q$. A partition (clustering result) can be equivalently represented as an n-by-k matrix Z whose elements are in $\{0,1\}$ where $Z(i,j) = 1$ if only if node i belongs to cluster j. Obviously, $Z \cdot \vec{1}_k = \vec{1}_n$, where $\vec{1}_k$ and $\vec{1}_n$ are k-dimensional and n-dimensional vectors of all ones. In addition, we can see that the columns of Z are orthogonal. Furthermore, we normalize Z in the following way: we divide the l-th column of Z by $\sqrt{|\mathcal{V}_l|}$ to get $\tilde{Z}$, where $|\mathcal{V}_l|$ is the size of $\mathcal{V}_l$. Note that the columns of $\tilde{Z}$ are orthonormal, i.e., $\tilde{Z}^T \tilde{Z} = I_k$.

2.2 K-means clustering

The *k-means clustering problem* is one of the most widely-studied clustering problems. Assume the i-th node in $\mathcal{V}$ can be represented by an m-dimensional feature vector $\vec{v}_i \in \mathcal{R}^m$, then the k-means clustering problem is to find a partition $\{\mathcal{V}_1, \ldots, \mathcal{V}_k\}$ that minimizes the following measure

$$KM = \sum_{l=1}^{k} \sum_{i \in \mathcal{V}_l} \|\vec{v}_i - \vec{\mu}_l\|^2 \qquad (1)$$

where $\vec{\mu}_l$ is the centroid (mean) of the l-th cluster, i.e., $\vec{\mu}_l = \sum_{j \in \mathcal{V}_l} \vec{v}_j / |\mathcal{V}_l|$.

A well-known algorithm to the k-means clustering problem is the so called *k-means algorithm* in which after initially randomly picking k centroids, the following procedure is repeated until convergence: all the data points are assigned to the clusters whose centroids are nearest to them, and then the cluster centroids are updated by taking the average of the data points assigned to them.

2.3 Spectral clustering

The basic idea of spectral clustering is to cluster based on the eigenvectors of a (possibly normalized) *similarity matrix* W defined on the set of nodes in $\mathcal{V}$. Very often W is positive semi-definite. Commonly used similarities include the inner product of the feature vectors, $W(i,j) = \vec{v}_i^T \vec{v}_j$, the diagonally-scaled Gaussian similarity, $W(i,j) = \exp(-(\vec{v}_i - \vec{v}_j)^T diag(\vec{\gamma})(\vec{v}_i - \vec{v}_j))$, and the affinity matrices of graphs.

Spectral clustering algorithms usually solve graph partitioning problems where different graph-based measures are to be optimized. Two popular measures are to maximize the *average association* and to minimize the *normalized cut* [19]. For two subsets, $\mathcal{V}_p$ and $\mathcal{V}_q$, of the node set $\mathcal{V}$ (where $\mathcal{V}_p$ and $\mathcal{V}_q$ do not have to be disjoint), we first define the *association* between $\mathcal{V}_p$ and $\mathcal{V}_q$ as $assoc(\mathcal{V}_p, \mathcal{V}_q) = \sum_{i \in \mathcal{V}_p, j \in \mathcal{V}_q} W(i,j)$ Then we can write the k-way average association as

$$AA = \sum_{l=1}^{k} \frac{assoc(\mathcal{V}_l, \mathcal{V}_l)}{|\mathcal{V}_l|} \qquad (2)$$

and the k-way normalized cut as

$$NC = \sum_{l=1}^{k} \frac{assoc(\mathcal{V}_l, \mathcal{V} \backslash \mathcal{V}_l)}{assoc(\mathcal{V}_l, \mathcal{V})} \qquad (3)$$

where $\mathcal{V} \backslash \mathcal{V}_l$ is the complement of $\mathcal{V}$. For consistency, we further define the *negated average association* as

$$NA = Tr(W) - AA = Tr(W) - \sum_{l=1}^{k} \frac{assoc(\mathcal{V}_l, \mathcal{V}_l)}{|\mathcal{V}_l|} \qquad (4)$$

where, as will be shown later, NA is always non-negative if W is positive semi-definite. In the remaining of the paper, instead of maximizing AA, we equivalently aim to minimize NA, and as a result, all the three objective functions, KM, NA and NC are to be minimized.

Finding the optimal partition Z for either the negated average association or the normalized cut is NP-hard [19]. Therefore, in spectral clustering algorithms, usually a relaxed version of the optimization problem is solved by (1) computing eigenvectors X of some variations of the similarity matrix W, (2) projecting all data points to $span(X)$, and (3) applying the k-means algorithm to the projected data points to obtain the clustering result. While it may seem nonintuitive to apply spectral analysis and then again use the k-means algorithm, it has been shown that such procedures have many advantages such as they work well in the cases when the data points are not linearly separable [17]. The focus of our paper is in step (1). For steps (2) and (3) we follow the standard procedures in traditional spectral clustering and thus will not give more details on them.

3. EVOLUTIONARY SPECTRAL CLUSTERING—TWO FRAMEWORKS

In this section we propose two frameworks for evolutionary spectral clustering. We first describe the basic idea.

3.1 Basic Idea

We define a general cost function to measure the quality of a clustering result on evolving data points. The function contains two costs. The first cost, *snapshot cost* (CS), only measures the snapshot quality of the current clustering result with respect to the *current* data features, where a higher snapshot cost means worse snapshot quality. The second cost, *temporal cost* (CT), measures the temporal smoothness in terms of the goodness-of-fit of the current clustering result with respect to either *historic* data features or *historic* clustering results, where a higher temporal cost means worse temporal smoothness. The overall cost function[1] is defined as a linear combination of these two costs:

$$Cost = \alpha \cdot CS + \beta \cdot CT \qquad (5)$$

[1]Our general cost function is equivalent to the one defined in [3], differing only by a constant factor and a negative sign.

where $0 \leq \alpha \leq 1$ is a parameter assigned by the user and together with $\beta(= 1 - \alpha)$, they reflect the user's emphasis on the snapshot cost and temporal cost, respectively.

In both frameworks that we propose, for a current partition (clustering result), the snapshot cost *CS* is measured by the clustering quality when the partition is applied to the current data. The two frameworks are different in how the temporal cost *CT* is defined. In the first framework, which we name PCQ for *preserving cluster quality*, the current partition is applied to *historic data* and the resulting cluster quality determines the temporal cost. In the second framework, which we name PCM for *preserving cluster membership*, the current partition is directly compared with the *historic partition* and the resulting difference determines the temporal cost.

In the discussion of both frameworks, we first use the k-means clustering problem, Equation (1), as a motivational example and then formulate the corresponding evolutionary spectral clustering problems (both *NA* and *NC*). We also provide the optimal solutions to the relaxed versions of the evolutionary spectral clustering problems and show how they relate back to the evolutionary k-means clustering problem. In addition, in this section, we focus on a special case where the number of clusters does not change with time and neither does the number of nodes to be clustered. We will discuss the more general cases in the next section.

3.2 Preserving Cluster Quality (PCQ)

In the first framework, PCQ, the temporal cost is expressed as how well the current partition clusters historic data. We illustrate this through an example. Assume that two partitions, Z_t and Z'_t, cluster the current data at time t equally well. However, to cluster historic data at time *t-1*, the clustering performance using partition Z_t is better than using partition Z'_t. In such a case, Z_t is preferred over Z'_t because Z_t is *more consistent with historic data*. We formalize this idea for the k-means clustering problem using the following overall cost function

$$\begin{aligned} Cost_{KM} &= \alpha \cdot CS_{KM} + \beta \cdot CT_{KM} \qquad (6)\\ &= \alpha \cdot KM_t\big|_{Z_t} + \beta \cdot KM_{t-1}\big|_{Z_t} \\ &= \alpha \cdot \sum_{l=1}^{k} \sum_{i \in \mathcal{V}_{l,t}} \|\vec{v}_{i,t} - \vec{\mu}_{l,t}\|^2 \\ &\quad + \beta \cdot \sum_{l=1}^{k} \sum_{i \in \mathcal{V}_{l,t}} \|\vec{v}_{i,t-1} - \vec{\mu}_{l,t-1}\|^2 \end{aligned}$$

where $\big|_{Z_t}$ means "evaluated by the partition Z_t, where Z_t is computed at time t" and $\vec{\mu}_{l,t-1} = \sum_{j \in \mathcal{V}_{l,t}} \vec{v}_{j,t-1}/|\mathcal{V}_{l,t}|$. Note that in the formula of CT_{KM}, the inner summation is over all data points in $\mathcal{V}_{l,t}$, the clusters at time t. That is, although the feature values used in the summation are those at time *t-1* (i.e., $\vec{v}_{i,t-1}$'s), the partition used is that at time t (i.e., Z_t). As a result, this cost $CT_{KM} = KM_{t-1}\big|_{Z_t}$ penalizes those clustering results (at t) that do not fit well with recent historic data (at *t-1*) and therefore promotes temporal smoothness of clusters.

3.2.1 Negated Average Association

We now formulate the PCQ framework for evolutionary spectral clustering. We start with the case of negated average association. Following the idea of Equation (6), at time t, for a given partition Z_t, a natural definition of the overall cost is

$$\begin{aligned} Cost_{NA} &= \alpha \cdot CS_{NA} + \beta \cdot CT_{NA} \qquad (7)\\ &= \alpha \cdot NA_t\big|_{Z_t} + \beta \cdot NA_{t-1}\big|_{Z_t} \end{aligned}$$

The above cost function is almost identical to Equation (6), except that the cluster quality is measured by the negated average association *NA* rather than the k-means *KM*.

Next, we derive a solution to minimizing $Cost_{NA}$. First, it can be easily shown that the negated average association defined in Equation (4) can be equivalently written as

$$NA = Tr(W) - Tr(\tilde{Z}^T W \tilde{Z}) \qquad (8)$$

Therefore[2] we write the overall cost (7) as

$$\begin{aligned} Cost_{NA} &= \alpha \cdot [Tr(W_t) - Tr(\tilde{Z}_t^T W_t \tilde{Z}_t)] \qquad (9)\\ &\quad + \beta \cdot [Tr(W_{t-1}) - Tr(\tilde{Z}_t^T W_{t-1} \tilde{Z}_t)] \\ &= Tr(\alpha W_t + \beta W_{t-1}) - Tr\left[\tilde{Z}_t^T (\alpha W_t + \beta W_{t-1}) \tilde{Z}_t\right] \end{aligned}$$

Notice that the first term $Tr(\alpha W_t + \beta W_{t-1})$ is a constant independent of the clustering partitions and as a result, minimizing $Cost_{NA}$ is equivalent to maximizing the trace $Tr[\tilde{Z}_t^T (\alpha W_t + \beta W_{t-1}) \tilde{Z}_t]$, subject to $\tilde{Z}_t$ being a normalized indicator matrix (*cf* Sec 2.1). Because maximizing the average association is an NP-hard problem, finding the solution $\tilde{Z}_t$ that minimizes $Cost_{NA}$ is also NP-hard. So following most spectral clustering algorithms, we relax $\tilde{Z}_t$ to $X_t \in \mathcal{R}^{n \times k}$ with $X_t^T X_t = I_k$. It is well-known [11] that one solution to this relaxed optimization problem is the matrix X_t whose columns are the k eigenvectors associated with the top-k eigenvalues of matrix $\alpha W_t + \beta W_{t-1}$. Therefore, after computing the solution X_t we can project the data points into $span(X_t)$ and then apply k-means to obtain a solution to the evolutionary spectral clustering problem under the measure of negated average association. In addition, the value $Tr(\alpha W_t + \beta W_{t-1}) - Tr\left[X_t^T (\alpha W_t + \beta W_{t-1}) X_t\right]$ provides a lower bound on the performance of the evolutionary clustering problem.

Moreover, Zha et al. [22] have shown a close connection between the k-means clustering problem and spectral clustering algorithms — they proved that if we put the m-dimensional feature vectors of the n data points in $\mathcal{V}$ into an m-by-n matrix $A = (\vec{v}_1, \ldots, \vec{v}_n)$, then

$$KM = Tr(A^T A) - Tr(\tilde{Z}^T A^T A \tilde{Z}) \qquad (10)$$

Comparing Equations (10) and (8), we can see that the k-means clustering problem is a special case of the negated average association spectral clustering problem, where the similarity matrix W is defined by the inner product $A^T A$. As a consequence, our solution to the *NA* evolutionary spectral clustering problem can also be applied to solve the k-means evolutionary clustering problem in the PCQ framework, i.e., under the cost function defined in Equation (6).

[2]Here we can show that *NA* is positive semi-definite: We have $\tilde{Z}^T \tilde{Z} = I_k$ and $Tr(W) = \sum_{i=1}^{n} \lambda_i$ where λ_i's are the eigenvalues of W ordered by decreasing magnitude. Therefore, by Fan's theorem [10], which says that $\max_{X \in \mathcal{R}^{n \times k}, X^T X = I_k} Tr(X^T W X) = \sum_{j=1}^{k} \lambda_k$, we can derive from (8) that $NA \geq \sum_{j=k+1}^{n} \lambda_j \geq 0$ if W is positive semi-definite.

3.2.2 Normalized Cut

For the normalized cut, we extend the idea of Equation (6) similarly. By replacing the *KM* Equation (6) with *NC*, we define the overall cost for evolutionary normalized cut to be

$$Cost_{NC} = \alpha \cdot CS_{NC} + \beta \cdot CT_{NC} \tag{11}$$
$$= \alpha \cdot NC_t\big|_{Z_t} + \beta \cdot NC_{t-1}\big|_{Z_t}$$

Shi et al. [19] have proved that computing the optimal solution to minimize the normalized cut is NP-hard. As a result, finding an indicator matrix Z_t that minimizes $Cost_{NC}$ is also NP-hard. We now provide an optimal solution to a relaxed version of the problem. Bach et al. [2] proved that for a given partition Z, the normalized cut can be equivalently written as

$$NC = k - Tr\left[Y^T\left(D^{-\frac{1}{2}}WD^{-\frac{1}{2}}\right)Y\right] \tag{12}$$

where D is a diagonal matrix with $D(i,i) = \sum_{j=1}^{n} W(i,j)$ and Y is any matrix in $\mathcal{R}^{n\times k}$ that satisfies two conditions: (a) the columns of $D^{-1/2}Y$ are piecewise constant with respect to Z and (b) $Y^TY = I_k$. We remove the constraint (a) to get a relaxed version for the optimization problem

$$Cost_{NC} \approx \alpha \cdot k - \alpha \cdot Tr\left[X_t^T\left(D_t^{-\frac{1}{2}}W_tD_t^{-\frac{1}{2}}\right)X_t\right] \tag{13}$$
$$+ \beta \cdot k - \beta \cdot Tr\left[X_t^T\left(D_{t-1}^{-\frac{1}{2}}W_{t-1}D_{t-1}^{-\frac{1}{2}}\right)X_t\right]$$
$$= k - Tr\left[X_t^T\left(\alpha D_t^{-\frac{1}{2}}W_tD_t^{-\frac{1}{2}} + \beta D_{t-1}^{-\frac{1}{2}}W_{t-1}D_{t-1}^{-\frac{1}{2}}\right)X_t\right]$$

for some $X_t \in \mathcal{R}^{n\times k}$ such that $X_t^TX_t = I_k$. Again we have a trace maximization problem and a solution is the matrix X_t whose columns are the k eigenvectors associated with the top-k eigenvalues of matrix $\alpha D_t^{-\frac{1}{2}}W_tD_t^{-\frac{1}{2}} + \beta D_{t-1}^{-\frac{1}{2}}W_{t-1}D_{t-1}^{-\frac{1}{2}}$. And again, after obtaining X_t, we can further project data points into $span(X_t)$ and then apply the k-means algorithm to obtain the final clusters.

Moreover, Dhillon et al. [8] have proved that the normalized cut approach can be used to minimize the cost function of a weighted kernel k-means problem. As a consequence, our evolutionary spectral clustering algorithm can also be applied to solve the evolutionary version of the weighted kernel k-means clustering problem.

3.2.3 Discussion on the PCQ Framework

The PCQ evolutionary clustering framework provides a data clustering technique similar to the moving average framework in time series analysis, in which the short-term fluctuation is expected to be smoothed out. The solutions to the PCQ framework turn out to be very intuitive — the historic similarity matrix is scaled and combined with current similarity matrix and the new combined similarity matrix is fed to traditional spectral clustering algorithms.

Notice that one assumption we have used in the above derivation is that the temporal cost is determined by data at time *t-1* only. However, the PCQ framework can be easily extended to cover longer historic data by including similarity matrices W's at older time, probably with different weights (e.g., scaled by an exponentially decaying factor to emphasize more recent history).

3.3 Preserving Cluster Membership (PCM)

The second framework of evolutionary spectral clustering, PCM, is different from the first framework, PCQ, in how the temporal cost is measured. In this second framework, the temporal cost is expressed as the difference between the current partition and the historic partition. We again illustrate this by an example. Assume that two partitions, Z_t and Z'_t, cluster current data at time t equally well. However, when compared to the historic partition Z_{t-1}, Z_t is much more similar to Z_{t-1} than Z'_t is. In such a case, Z_t is preferred over Z'_t because Z_t is *more consistent with historic partition*.

We first formalize this idea for the evolutionary k-means problem. For convenience of discussion, we write the current partition as $Z_t = \{\mathcal{V}_{1,t}, \ldots, \mathcal{V}_{k,t}\}$ and the historic partition as $Z_{t-1} = \{\mathcal{V}_{1,t-1}, \ldots, \mathcal{V}_{k,t-1}\}$. Now we want to define a measure for the difference between Z_t and Z_{t-1}. Comparing two partitions has long been studied in the literatures of classification and clustering. Here we use the traditional chi-square statistics [14] to represent the distance between two partitions

$$\chi^2(Z_t, Z_{t-1}) = n\left(\sum_{i=1}^{k}\sum_{j=1}^{k}\frac{|\mathcal{V}_{ij}|^2}{|\mathcal{V}_{i,t}|\cdot|\mathcal{V}_{j,t-1}|} - 1\right)$$

where $|\mathcal{V}_{ij}|$ is the number of nodes that are both in $\mathcal{V}_{i,t}$ (at time t) and in $\mathcal{V}_{j,t-1}$ (at time *t-1*). Actually, in the above definition, the number of clusters k does not have to be the same at time t and *t-1*, and we will come back to this point in the next section. By ignoring the constant shift of -1 and the constant scaling n, we define the temporal cost for the k-means clustering problem as

$$CT_{KM} = -\sum_{i=1}^{k}\sum_{j=1}^{k}\frac{|\mathcal{V}_{ij}|^2}{|\mathcal{V}_{i,t}|\cdot|\mathcal{V}_{j,t-1}|} \tag{14}$$

where the negative sign is because we want to *minimize* CT_{KM}. The overall cost can be written as

$$Cost_{KM} = \alpha \cdot CS_{KM} + \beta \cdot CT_{KM} \tag{15}$$
$$= \alpha \cdot \sum_{l=1}^{k}\sum_{i\in\mathcal{V}_{l,t}}\|\vec{v}_{i,t} - \vec{\mu}_{l,t}\|^2 - \beta \cdot \sum_{i=1}^{k}\sum_{j=1}^{k}\frac{|\mathcal{V}_{ij}|^2}{|\mathcal{V}_{i,t}|\cdot|\mathcal{V}_{j,t-1}|}$$

3.3.1 Negated Average Association

Recall that in the case of negated average association, we want to maximize $NA = Tr(\tilde{Z}^TW\tilde{Z})$ where $\tilde{Z}$ is further relaxed to continuous-valued X, whose columns are the k eigenvectors associated with the top-k eigenvalues of W. So in the PCM framework, we shall define a distance $dist(X_t, X_{t-1})$ between X_t, a set of eigenvectors at time t, and X_{t-1}, a set of eigenvectors at time *t-1*. However, there is a subtle point — for a solution $X \in \mathcal{R}^{n\times k}$ that maximizes $Tr(X^TWX)$, any $X' = XQ$ is also a solution, where $Q \in \mathcal{R}^{k\times k}$ is an arbitrary orthogonal matrix. This is because $Tr(X^TWX) = Tr(X^TWXQQ^T) = Tr((XQ)^TWXQ) = Tr(X'^TWX')$. Therefore we want a distance $dist(X_t, X_{t-1})$ that is invariant with respect to the rotation Q. One such solution, according to [11], is the norm of the difference between two projection matrices, i.e.,

$$dist(X_t, X_{t-1}) = \frac{1}{2}\|X_tX_t^T - X_{t-1}X_{t-1}^T\|^2 \tag{16}$$

which essentially measures the distance between $span(X_t)$ and $span(X_{t-1})$. Furthermore in Equation (16), the number of columns in X_t does not have to be the same as that in X_{t-1} and we will discuss this in the next section.

By using this distance to quantify the temporal cost, we derive the total cost for the negated average association as

$$\begin{aligned}
&Cost_{NA} = \alpha \cdot CS_{NA} + \beta \cdot CT_{NA} \qquad (17)\\
=&\alpha \cdot \Big[Tr(W_t) - Tr(X_t^T W_t X_t)\Big] + \frac{\beta}{2} \cdot \|X_t X_t^T - X_{t-1}X_{t-1}^T\|^2\\
=&\alpha \cdot \Big[Tr(W_t) - Tr(X_t^T W_t X_t)\Big] +\\
&\frac{\beta}{2} Tr\left(X_t X_t^T - X_{t-1}X_{t-1}^T\right)^T\left(X_t X_t^T - X_{t-1}X_{t-1}^T\right)\\
=&\alpha \cdot \Big[Tr(W_t) - Tr(X_t^T W_t X_t)\Big] +\\
&\frac{\beta}{2} Tr(X_t X_t^T X_t X_t^T - 2X_t X_t^T X_{t-1}X_{t-1}^T + X_{t-1}X_{t-1}^T X_{t-1}X_{t-1}^T)\\
=&\alpha \cdot \Big[Tr(W_t) - Tr(X_t^T W_t X_t)\Big] + \beta k - \beta Tr\left(X_t^T X_{t-1}X_{t-1}^T X_t\right)\\
=&\alpha \cdot Tr(W_t) + \beta \cdot k - Tr\Big[X_t^T(\alpha W_t + \beta X_{t-1}X_{t-1}^T)X_t\Big]
\end{aligned}$$

Therefore, an optimal solution that minimizes $Cost_{NA}$ is the matrix X_t whose columns are the k eigenvectors associated with the top-k eigenvalues of the matrix $\alpha W_t + \beta X_{t-1}X_{t-1}^T$. After getting X_t, the following steps are the same as before.

Furthermore, it can be shown that the un-relaxed version of the distance defined in Equation (16) for spectral clustering is equal to that defined in Equation (15) for k-means clustering by a constant shift. That is, it can be shown (*cf.* Bach et al. [2]) that

$$\frac{1}{2}\|\tilde{Z}_t\tilde{Z}_t^T - \tilde{Z}_{t-1}\tilde{Z}_{t-1}^T\|^2 = k - \sum_{i=1}^{k}\sum_{j=1}^{k}\frac{|\mathcal{V}_{ij}|^2}{|\mathcal{V}_{i,t}| \cdot |\mathcal{V}_{j,t-1}|} \qquad (18)$$

As a result, the evolutionary spectral clustering based on negated average association in the PCM framework provides a relaxed solution to the evolutionary k-means clustering problem defined in the PCM framework, i.e., Equation (15).

3.3.2 Normalized Cut

It is straightforward to extend the PCM framework from the negated average association to normalized cut as

$$\begin{aligned}
Cost_{NC} &= \alpha \cdot CS_{NC} + \beta \cdot CT_{NC} \qquad (19)\\
&= \alpha \cdot k - \alpha \cdot Tr\left[X_t^T\left(D_t^{-\frac{1}{2}}W_t D_t^{-\frac{1}{2}}\right)X_t\right]\\
&\quad + \frac{\beta}{2} \cdot \|X_t X_t^T - X_{t-1}X_{t-1}^T\|^2\\
&= k - Tr\left[X_t^T\left(\alpha D_t^{-\frac{1}{2}}W_t D_t^{-\frac{1}{2}} + \beta X_{t-1}X_{t-1}^T\right)X_t\right]
\end{aligned}$$

Therefore, an optimal solution that minimizes $Cost_{NC}$ is the matrix X_t whose columns are the k eigenvectors associated with the top-k eigenvalues of the matrix $\alpha D_t^{-\frac{1}{2}}W_t D_t^{-\frac{1}{2}} + \beta X_{t-1}X_{t-1}^T$. After obtaining X_t, the subsequent steps are the same as before.

It is worth mentioning that in the PCM framework, $Cost_{NC}$ has an advantage over $Cost_{NA}$ in terms of the ease of selecting an appropriate α. In $Cost_{NA}$, the two terms CS_{NA} and CT_{NA} are of different scales — CS_{NA} measures a sum of variances and CT_{NA} measures some probability distribution. Consequently, this difference needs to be considered when choosing α. In contrast, for $Cost_{NC}$, because the CS_{NC} is normalized, both $D_t^{-\frac{1}{2}}W_t D_t^{-\frac{1}{2}}$ and $X_{t-1}X_{t-1}^T$ have the same 2-norms scale, for both matrices have $\lambda_{max} = 1$. Therefore, the two terms CS_{NC} and CT_{NC} are comparable and α can be selected in a straightforward way.

3.3.3 Discussion on the PCM Framework

In the PCM evolutionary clustering framework, all historic data are taken into consideration (with different weights) — X_t partly depends on X_{t-1}, which in turn partly depends on X_{t-2} and so on. Let us look at two extreme cases. When α approaches 1, the temporal cost will become unimportant and as a result, the clusters are computed at each time window independent of other time windows. On the other hand, when α approaches 0, the eigenvectors in all time windows are required to be identical. Then the problem becomes a special case of the higher-order singular value decomposition problem [7], in which singular vectors are computed for the three modes (the rows of W, the columns of W, and the timeline) of a data tensor $\mathcal{W}$ where $\mathcal{W}$ is constructed by concatenating W_t's along the timeline.

In addition, if the similarity matrix W_t is positive semi-definite, then $\alpha D_t^{-\frac{1}{2}}W_t D_t^{-\frac{1}{2}} + \beta X_{t-1}X_{t-1}^T$ is also *positive semi-definite* because both $D_t^{-\frac{1}{2}}W_t D_t^{-\frac{1}{2}}$ and $X_{t-1}X_{t-1}^T$ are positive semi-definite.

3.4 Comparing Frameworks PCQ and PCM

Now we compare the PCQ and PCM frameworks. For simplicity of discussion, we only consider time slots t and *t-1* and ignore older history.

In terms of the temporal cost, PCQ aims to maximize $Tr(X_t^T W_{t-1} X_t)$ while for PCM, $Tr(X_t^T X_{t-1}X_{t-1}^T X_t)$ is to be maximized. However, these two are closely connected. By applying the eigen-decomposition on W_{t-1}, we have

$$X_t^T W_{t-1} X_t = X_t^T (X_{t-1}, X_{t-1}^{\perp})\Lambda_{t-1}(X_{t-1}, X_{t-1}^{\perp})^T X_t$$

where Λ_{t-1} is a diagonal matrix whose diagonal elements are the eigenvalues of W_{t-1} ordered by decreasing magnitude, and X_{t-1} and $X_{t-1}^{\perp}$ are the eigenvectors associated with the first k and the residual $n-k$ eigenvectors of W_{t-1}, respectively. It can be easily verified that both $Tr(X_t^T W_{t-1} X_t)$ and $Tr(X_t^T X_{t-1}X_{t-1}^T X_t)$ are maximized when $X_t = X_{t-1}$ (or more rigorously, when $span(X_t) = span(X_{t-1})$). The differences between PCQ and PCM are (a) if the eigenvectors associated with the smaller eigenvalues (other than the top k) are considered and (b) the level of penalty when X_t deviates from X_{t-1}. For PCQ, all the eigenvectors are considered and their deviations between time t and *t-1* are penalized according to the corresponding eigenvalues. For PCM, rather than all eigenvectors, only the first k eigenvectors are considered and they are treated equally. In other words, in the PCM framework, other than the historic cluster membership, all details about historic data are ignored.

Although by keeping only historic cluster membership, PCM introduces more information loss, there may be benefits in other aspects. For example, the *CT* part in the PCM framework does not necessarily have to be temporal cost — it can represent any prior knowledge about cluster membership. For example, we can cluster blogs purely based on interlinks. However, other information such as the content of the blogs and the demographic data about the bloggers may provide valuable prior knowledge about cluster membership that can be incorporated into the clustering. The PCM framework can handle such information fusion easily.

4. GENERALIZATION

There are two assumptions in the PCQ and the PCM framework proposed in the last section. First, we assumed that the number of clusters remains the same over all time. Second, we assumed that the same set of nodes is to be clustered in all timesteps. Both assumptions are too restrictive in many applications. In this section, we extend our frameworks to handle the issues of variation in cluster numbers and insertion/removal of nodes over time.

4.1 Variation in Cluster Numbers

In our discussions so far, we have assumed that the number of clusters k does not change with time. However, keeping a fixed k over all time windows is a very strong restriction. Determining the number of clusters is an important research problem in clustering and there are many effective methods for selecting appropriate cluster numbers (e.g., by thresholding the gaps between consecutive eigenvalues). Here we assume that the number of cluster k at time t has been determined by one of these methods and we investigate what will happen if the cluster number k at time t is different from the cluster number k' at time *t-1*.

It turns out that both the PCQ and the PCM frameworks can handle variations in cluster number already. In the PCQ framework, the temporal cost is expressed by *historic data* themselves, not by *historic clusters* and therefore the computation at time t is independent of the cluster number k' at time *t-1*. In the PCM framework, as we have mentioned, the partition distance (Equation 14) and the subspace distance (Equation 16) can both be used without change when the two partitions have different numbers of clusters. As a result, both of our PCQ and PCM frameworks can handle variations in the cluster numbers.

4.2 Insertion and Removal of Nodes

Another assumption that we have been using is that the number of nodes in $\mathcal{V}$ does not change with time. However, in many applications the data points to be clustered may vary with time. In the blog example, very often there are old bloggers who stop blogging and new bloggers who just start. Here we propose some heuristic solutions to handle this issue.

4.2.1 Node Insertion and Removal in PCQ

For the PCQ framework, the key is $\alpha W_t + \beta W_{t-1}$. When old nodes are removed, we can simply remove the corresponding rows and columns from W_{t-1} to get $\tilde{W}_{t-1}$ (assuming $\tilde{W}_{t-1}$ is $n_1 \times n_1$). However, when new nodes are inserted at time t, we need to add entries to $\tilde{W}_{t-1}$ and to extended it to $\hat{W}_{t-1}$, which has the same dimension as W_t (assuming W_t is $n_2 \times n_2$). Without lost of generality, we assume that the first n_1 rows and columns of W_t correspond to those nodes in $\tilde{W}_{t-1}$. We propose to achieve this by defining

$$\hat{W}_{t-1}=\begin{bmatrix}\tilde{W}_{t-1} & E_{t-1}\\ E_{t-1}^T & F_{t-1}\end{bmatrix} \text{ for } \begin{cases} E_{t-1} = \frac{1}{n_1}\tilde{W}_{t-1}\vec{1}_{n_1}\vec{1}_{n_2-n_1}^T \\ F_{t-1} = \frac{1}{n_1^2}\vec{1}_{n_1}^T\tilde{W}_{t-1}\vec{1}_{n_1}\vec{1}_{n_2-n_1}\vec{1}_{n_2-n_1}^T \end{cases}$$

Such a heuristic has the following good properties, whose proofs are skipped due to the space limitation.

PROPERTY 1. *(1) $\hat{W}_{t-1}$ is positive semi-definite if W_{t-1} is. (2) In $\hat{W}_{t-1}$, for each existing node v_{old}, each newly inserted node v_{new} looks like an average node in that the similarity between v_{new} and v_{old} is the same as the average similarity between any existing node and v_{old}. (3) In $\hat{W}_{t-1}$, the similarity between any pair of newly inserted nodes is the same as the average similarity among all pairs of existing nodes.*

We can see that these properties are appealing when no prior knowledge is given about the newly inserted nodes.

4.2.2 Node Insertion and Removal in PCM

For the PCM framework, when old nodes are removed, we remove the corresponding rows from X_{t-1} to get $\tilde{X}_{t-1}$ (assuming $\tilde{X}_{t-1}$ is $n_1 \times k$). When new nodes are inserted at time t, we extend $\tilde{X}_{t-1}$ to $\hat{X}_{t-1}$, which has the same dimension as X_t (assuming X_t is $n_2 \times k$) as follows

$$\hat{X}_{t-1} = \begin{bmatrix}\tilde{X}_{t-1}\\ G_{t-1}\end{bmatrix} \text{ for } G_{t-1} = \frac{1}{n_1}\vec{1}_{n_2-n_1}\vec{1}_{n_1}^T\tilde{X}_{t-1} \tag{20}$$

That is, we insert new rows as the row average of $\tilde{X}_{t-1}$. After obtaining $\hat{X}_{t-1}$, we replace the term $\beta X_{t-1}X_{t-1}^T$ with $\beta\hat{X}_{t-1}(\hat{X}_{t-1}^T\hat{X}_{t-1})^{-1}\hat{X}_{t-1}^T$ in Equations (17) and (19).

Such a heuristic has the following good property, whose proof is skipped due to the space limit.

PROPERTY 2. *Equation (20) corresponds to for each newly inserted nodes, assigning to it a prior clustering membership that is approximately proportional to the size of the clusters at time t-1.*

5. EXPERIMENTAL STUDIES

In this section, we report experimental studies based on both synthetic data sets and a real blog data set.

5.1 Synthetic Data

First, we use several experiments on synthetic data sets to illustrate the good properties of our algorithms.

5.1.1 NA-based Evolutionary Spectral Clustering

In this subsection, we show three experimental studies based on synthetic data. In the first experiment, we demonstrate a stationary case where data variation is due to a zero-mean noise. In the second experiment, we show a non-stationary case where there are concept drifts. In the third experiment, we show a case where there is a large difference between the PCQ and PCM frameworks.

By using the k-means algorithm, we design two baselines. The first baseline, which we call ACC, accumulates all historic data before the current timestep t and applies the k-means algorithm on the aggregated data. The second baseline, which we call IND, independently applies the k-means algorithm on the data in only timestep t and ignore all historic data before t.

For our algorithms, we use the *NA*-based PCQ and PCM algorithms, because of the equivalence between the *NA*-based spectral clustering problem and the k-means clustering problem (Equation (10)). We choose to use $W = A^TA$ in the *NA*-based evolutionary spectral clustering and compare its results with that of the k-means baseline algorithms. For a fair comparison, we use the *KM* defined for the k-means clustering problem (i.e., Equation (1)) as the measure for performance, where a smaller *KM* value is better.

The data points to be clustered are generated in the following way. 800 two-dimensional data points are initially positioned as described in Figure 2(a) at timestep 1. As can be seen, there are roughly four clusters (the data were actually generated by using four Gaussian distributions centered in the four quadrants). Then in timesteps 2 to 10, we perturb the initial positions of the data points by adding different noises according to the experimental setup. Unless stated otherwise, all experiments are repeated 50 times with different random seeds and the average performances are reported.

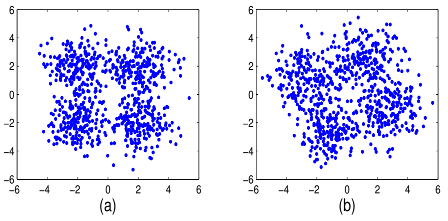

Figure 2: (a) The initial data point positions and (b) A typical position in the non-stationary case

In the first experiment, for timesteps 2 through 10, we add an i.i.d. Gaussian noise following $N(0, 0.5)$ to the initial positions of the data points. We use this data to simulation a stationary situation where the concept is relatively stable but there exist short-term noises. In Figures 3(a) and 3(b), we report the snapshot cost CS_{KM} and the temporal cost CT_{KM} for the two baselines and for our algorithms (with $\alpha = 0.9$ for both PCQ and PCM) from timesteps 1 to 10. For both costs, a lower value is better. As can be seen from the figure, the ACC baseline has low temporal smoothness but very high snapshot cost, whereas the IND baseline has the low snapshot cost but very high temporal cost. In comparison, our two algorithms show low temporal cost at the price of a little increase in snapshot cost. The overall cost $\alpha \cdot CS_{KM} + \beta \cdot CT_{KM}$ is given in Figure 3(c). As can be seen, the ACC baseline has the worst overall performance and our algorithms improve a little over the IND baseline. In addition, Figure 3(d) shows the degree of cluster change over time as defined in Equation (18). We can see that as

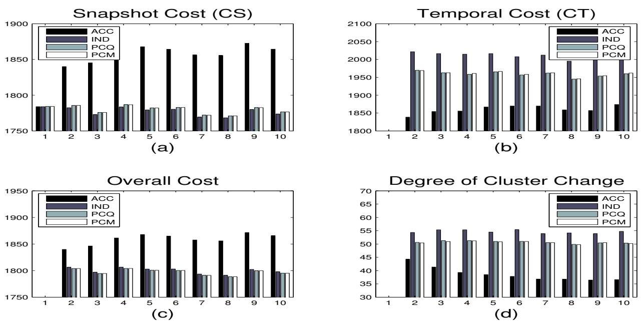

Figure 3: The performance for the stationary synthetic data set, which shows that PCQ and PCM result in low temporal cost at a price of a small increase in snapshot cost

expected, the cluster membership change using our frameworks is less dramatic than that of the IND baseline, which takes no historic information into account.

Next, for the same data set, we let α increase from 0.2 to 1 with a step of 0.1. Figure 4 shows the average snapshot cost and the temporal cost over all 10 timesteps under different settings of α. As we expected, when α increases, to emphasize more on the snapshot cost, we get better snapshot quality at the price of worse temporal smoothness. This result demonstrates that our frameworks are able to control the tradeoff between the snapshot quality and the temporal smoothness.

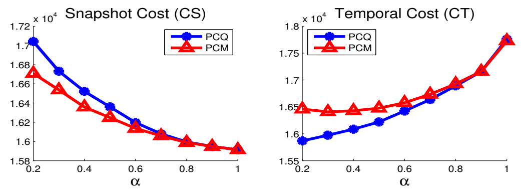

Figure 4: The tradeoff between snapshot cost and temporal cost, which can be controlled by α

In the second experiment, we simulate a non-stationary situation. At timesteps 2 through 10, before adding random noises, we first rotate all data points by a small random angle (with zero mean and a variance of $\pi/4$). Figure 2(b) shows the positions of data points in a typical timestep. Figure 5 gives the performance of the four algorithms. As can be seen, while the performance of our algorithms and the IND baseline has little change, the performance of the ACC baseline becomes very poor. This result shows that if an aggregation approach is used, we should not aggregate the data features in a non-stationary scenario — instead, we should aggregate the similarities among data points.

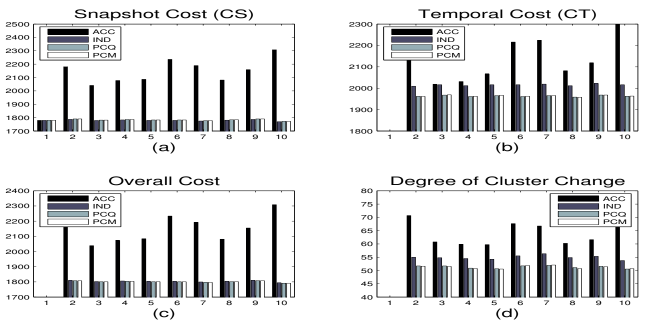

Figure 5: Performance for a non-stationary synthetic data set, which shows that aggregating data features does not work

In the third experiment, we show a case where the PCQ and PCM frameworks behave differently. We first generate data points using the procedure described in the first experiment (the stationary scenario), except that this time we generate 60 timesteps for a better view. This time, instead of 4 clusters, we let the algorithms partition the data into 2 clusters. From Figure 2(a) we can see that there are obviously two possible partitions, a horizonal cut or a vertical cut at the center, that will give similar performance where the performance difference will mainly be due to short-term noises. Figure 6 shows the degree of cluster membership change over the 60 timesteps in one run (for obvious reasons, no averaging is taken in this experiment). As can be seen, the cluster memberships of the two baselines jump around, which shows that they are not robust to noise in this case. Also can be seen, the cluster membership of the PCM algorithm varies much lesser than that of the PCQ algorithm. The reason for this difference is that switching the partition from the horizontal cut to the vertical cut will introduce much higher penalty to PCM than to PCQ — PCM is directly penalized by the change of eigenvectors, which corresponds to the change of cluster membership; for PCQ, the penalty is indirectly acquired from historic data, not historic cluster membership.

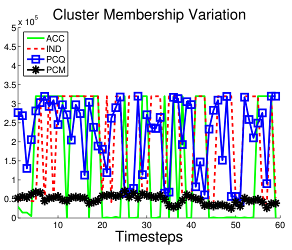

Figure 6: A case where PCM is more robust vs PCQ

5.1.2 *NC-based Evolutionary Spectral Clustering*

It is difficult to compare the *NC*-based evolutionary spectral clustering with the k-means clustering algorithm. Instead, in this experiment, we use a toy example in the 2-dimensional Euclidean space with only 4 timesteps (as shown in Figure 7) to compare the non-evolutionary version (upper panels, with $\alpha = 1$) and the evolutionary version (lower panels, with $\alpha = 0.9$) of the *NC*-based evolutionary spectral clustering algorithms. Figure 7 gives the clustering results with the correct cluster numbers provided to the algorithm. As can be seen, for the non-evolutionary version, at timestep 2, the two letters "D"'s are confused because they move too near to each other. At timestep 4, due to the change of cluster number, part of the newly introduced letter "0" is confused with the second "D". Neither happens to the evolutionary version, in which the temporal smoothness is taken into account.

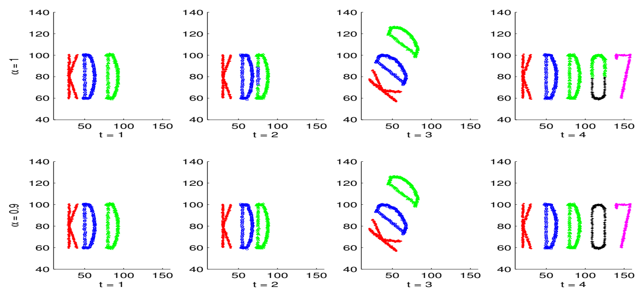

Figure 7: A toy example that demonstrates our evolutionary spectral clustering is more robust and can handle changes of cluster number

As a conclusion, these experiments based on synthetic data sets demonstrate that compared to traditional clustering methods, our evolutionary spectral clustering algorithms can provide clustering results that are more stable and consistent, less sensitive to short-term noise, and adaptive to long-term trends.

5.2 Real Blog Data

The real blog data was collected by an NEC in-house blog crawler. Due to space limit, we will not describe how the data was collected and refer interested readers to [18] for details. This NEC blog data set contains 148,681 entry-to-entry links among 407 blogs crawled during 63 consecutive weeks, between July 10th in 2005 and September 23rd in 2006. By looking at the *contents* of the blogs, we discovered two main groups: a set of 314 blogs with technology focus and a set of 93 blogs with politics focus. Figure 8 shows the blog graph for this NEC data set, where the nodes are blogs (with different labels depending on their group member) and the edges are interlinks among blogs (obtained by aggregating all entry-to-entry links).

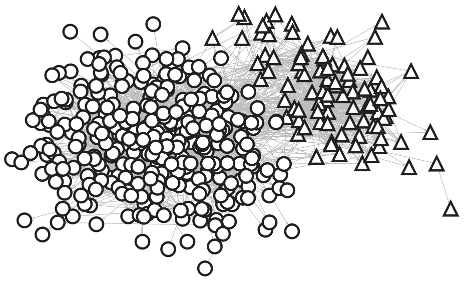

Figure 8: The blog graph for the NEC data set

One application of clustering blogs is to discover communities. Since we already have the ground truth of the two communities based on content analysis, we start by running the clustering algorithms with $k = 2$. The data is prepared in this way: each week corresponds to a timestep; all the entry-to-entry links in a week are used to construct an affinity matrix for the blogs of that week (i.e., those blogs that are relevant to at least one entry-to-entry link in that week); and the affinity matrix is used as the similarity matrix W in the clustering algorithms. For baselines, we again use ACC and IND, except that this time the normalized cut algorithm is used. For our algorithms, we use the *NC*-based PCQ and PCM. Figures 9(a),(b), and (c) give the CS_{NC}, CT_{NC}, and $Cost_{NC}$ for the two baseline algorithms and the PCM algorithm (to make the figures readable, we did not plot the results for PCQ, which are similar to those of PCM,

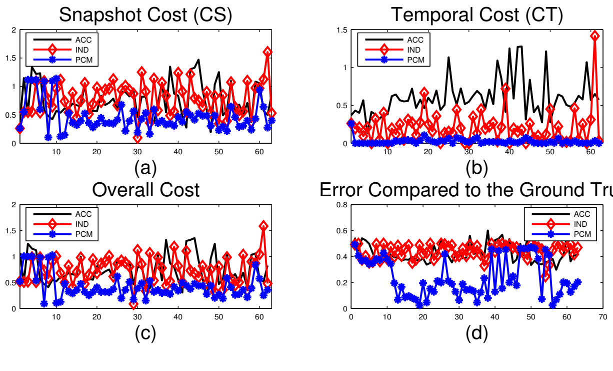

Figure 9: The performance on the NEC data, which shows that evolutionary spectral clustering clearly outperforms non-evolutionary ones

as shown in Table 1). In Figure 9(d), we show the error between the cluster results and the ground truth obtained from content analysis, where the error is the distance between partitions defined in Equation (18). As can be seen from these figures, the evolutionary spectral clustering has the best performance in all four measures. The high snapshot cost of IND was surprising to us. We believe this could be due to the non-robustness of the normalized cut package (which we obtained from the homepage of the first author of [19]). In addition, note that CT_{NC} is usually smaller than CS_{NC} because CT_{NC} is computed over those nodes that are active in both t and $t\text{-}1$ and such nodes are usually less than those that are active at t. This is also one of the reasons for the high variation of the curves.

Table 1: Performance under Different Cluster Numbers for the Blog Data Set

		ACC	IND	NC_PCQ	NC_PCM
k=2	CS	0.76	0.79	0.68	**0.46**
	CT	0.59	0.20	0.10	**0.06**
	Total Cost	0.74	0.73	0.63	**0.42**
k=3	CS	1.22	1.53	1.12	**1.07**
	CT	0.98	0.22	0.24	**0.02**
	Total Cost	1.21	1.43	1.06	**0.98**
k=4	CS	1.71	2.05	**1.70**	1.71
	CT	1.40	0.18	0.39	**0.03**
	Total Cost	1.69	1.89	1.59	**1.57**

In addition, we run the algorithms under different cluster numbers and report the performance in Table 1, where the best results among the same category are in bold face. Our evolutionary clustering algorithms always give more stable and consistent cluster results than the baselines where the historic data is totally ignored or totally aggregated.

6. CONCLUSION

There are new challenges when traditional clustering techniques are applied to new data types, such as streaming data and Web/blog data, where the relationship among data evolves with time. On one hand, because of long-term concept drifts, a naive approach based on aggregation will not give satisfactory cluster results. On the other hand, short-term variations occur very often due to noise. Preferably the cluster results should not change dramatically over short time and should exhibit temporal smoothness. In this paper, we propose two frameworks to incorporate temporal smoothness in evolutionary spectral clustering. In both frameworks, a cost function is defined where in addition to the traditional cluster quality cost, a second cost is introduced to regularize the temporal smoothness. We then derive the (relaxed) optimal solutions for solving the cost functions. The solutions turn out to have very intuitive interpretation and have forms analogous to traditional techniques used in time series analysis. Experimental studies demonstrate that these new frameworks provide cluster results that are both stable and consistent in the short-term and adaptive in the long run.

7. ACKNOWLEDGMENTS

We thank Shenghuo Zhu, Wei Xu, and Kai Yu for the inspiring discussions, and thank Junichi Tatemura for helping us prepare the data sets.

8. REFERENCES

[1] C. C. Aggarwal, J. Han, J. Wang, and P. S. Yu. A framework for clustering evolving data streams. In *Proc. of the 12th VLDB Conference*, 2003.

[2] F. R. Bach and M. I. Jordan. Learning spectral clustering, with application to speech separation. *Journal of Machine Learning Research*, 7, 2006.

[3] D. Chakrabarti, R. Kumar, and A. Tomkins. Evolutionary clustering. In *Proc. of the 12th ACM SIGKDD Conference*, 2006.

[4] M. Charikar, C. Chekuri, T. Feder, and R. Motwani. Incremental clustering and dynamic information retrieval. In *Proc. of the 29th STOC Conference*, 1997.

[5] C. Chatfield. *The Analysis of Time Series: An Introduction.* Chapman & Hall/CRC.

[6] F. R. K. Chung. *Spectral Graph Theory.* American Mathematical Society, 1997.

[7] L. De Lathauwer, B. De Moor, and J. Vandewalle. A multilinear singular value decomposition. *SIAM J. on Matrix Analysis and Applications*, 21(4), 2000.

[8] I. S. Dhillon, Y. Guan, and B. Kulis. Kernel k-means: spectral clustering and normalized cuts. In *Proc. of the 10th ACM SIGKDD Conference*, 2004.

[9] C. Ding and X. He. K-means clustering via principal component analysis. In *Proc. of the 21st ICML Conference*, 2004.

[10] K. Fan. On a theorem of weyl concerning eigenvalues of linear transformations. In *Proc. Natl. Acad. Sci.*, 1949.

[11] G. Golub and C. V. Loan. *Matrix Computations.* Johns Hopkins University Press, third edition, 1996.

[12] S. Guha, N. Mishra, R. Motwani, and L. O'Callaghan. Clustering data streams. In *IEEE Symposium on Foundations of Computer Science*, 2000.

[13] C. Gupta and R. Grossman. Genic: A single pass generalized incremental algorithm for clustering. In *SIAM Int. Conf. on Data Mining*, 2004.

[14] L. J. Hubert and P. Arabie. Comparing partitions. *Journal of Classification*, 2, 1985.

[15] X. Ji and W. Xu. Document clustering with prior knowledge. In *SIGIR*, 2006.

[16] Y. Li, J. Han, and J. Yang. Clustering moving objects. In *Proc. of the 10th ACM SIGKDD Conference*, 2004.

[17] A. Ng, M. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In *NIPS*, 2001.

[18] H. Ning, W. Xu, Y. Chi, Y. Gong, and T. Huang. Incremental spectral clustering with application to monitoring of evolving blog communities. In *SIAM Int. Conf. on Data Mining*, 2007.

[19] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 22(8), 2000.

[20] K. Wagstaff, C. Cardie, S. Rogers, and S. Schroedl. Constrained K-means clustering with background knowledge. In *Proc. 18th ICML Conference*, 2001.

[21] Y. Weiss. Segmentation using eigenvectors: A unifying view. In *ICCV '99: Proceedings of the International Conference on Computer Vision-Volume 2*, 1999.

[22] H. Zha, X. He, C. H. Q. Ding, M. Gu, and H. D. Simon. Spectral relaxation for k-means clustering. In *NIPS*, 2001.