This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication.

Low-rank Kernel Matrix Factorization for Large Scale Evolutionary Clustering

Lijun Wang, Manjeet Rege, Ming Dong, *Member, IEEE*, Yongsheng Ding

Abstract—Traditional clustering techniques are inapplicable to problems where the relationships between data points evolve over time. Not only is it important for the clustering algorithm to adapt to the recent changes in the evolving data, but it also needs to take the historical relationship between the data points into consideration. In this paper, we propose ECKF, a general framework for evolutionary clustering large-scale data based on low-rank kernel matrix factorization. To the best of our knowledge, this is the first work that clusters large evolutionary datasets by the amalgamation of low-rank matrix approximation methods and matrix factorization based clustering. Since the low-rank approximation provides a compact representation of the original matrix, and especially, the near-optimal low-rank approximation can preserve the sparsity of the original data, ECKF gains computational efficiency and hence is applicable to large evolutionary datasets. Moreover, matrix factorization based methods have been shown to effectively cluster high dimensional data in text mining and multimedia data analysis. From a theoretical standpoint, we mathematically prove the convergence and correctness of ECKF, and provide detailed analysis of its computational efficiency (both time and space). Through extensive experiments performed on synthetic and real datasets, we show that ECKF outperforms the existing methods in evolutionary clustering.

Index Terms—Clustering, Low-rank Matrix Approximation, Matrix Decomposition.

1 Introduction

Data clustering in general, is the task of automatically grouping data points into meaningful groups, known as clusters [1], [2]. While clustering has been applied to varied tasks such as information discovery [3]–[5], text mining [6]–[9], Web analysis [10], [11], image grouping [12], [13], image segmentation [14], and bioinformatics [15]–[17], the data in these cases is "static" in nature. In these approaches, a clustering algorithm is applied to a dataset that is previously collected at a particular point in time. This requirement can prove to be a grave restriction for the knowledge discovery process.

In a typical large data mining application, data is not only collected over a period of time, but the relationship between data points can change over that period too. For example, in clustering of social network users for discovering communities [18], [19], it is natural to expect that the association between users will change over a series of time steps. Moreover, this change may either be drastic (e.g., a large of number of active users adding each other) or gradual (e.g., a small set of users slowly become inactive). Consequently, this change should be reflected in the communities discovered at every time step. However, it is important that the clustering does not deviate too much from the recent past due to a sudden change in the relationships between data points. At the same time, for an evolving change in the nature, the clustering algorithm should reflect the corresponding change in the results as well. Stated differently, clustering at a particular time step should be based on the associations between data points (features) at that time step. In addition, the clustering at that time step should not drastically be different from the clustering of the recent past. This actually is a reasonable expectation in clustering of real world datasets. This is mainly because, in a realistic dataset, one typically does not expect to see a sudden change in features at a particular time step. An occurrence of this kind should mostly be due to the existence of noise, and the algorithm should be robust enough to overcome it. This dual objective evolving nature of clustering is radically different from the goal of a traditional clustering algorithm, and falls in the paradigm of *evolutionary clustering* [20]. Coupled with this, the size of the data poses its own set of challenges. It is unreasonable and in many cases even computationally infeasible to re-cluster as the large-scale data evolves.

In this paper, we present a matrix factorization based approach for large-scale evolutionary clustering. Matrix factorization based methods have been shown to effectively cluster high dimensional data in text mining and multimedia data analysis. We propose ECKF, a general model for large-scale **E**volutionary **C**lustering based on low-rank **K**ernel matrix **F**actorization. By integrating low-rank kernel matrix approximation, ECKF partitions extremely large data sets at every time step. In addition, ECKF has desirable properties over other approaches in terms of the accuracy, efficiency, and robustness in evolutionary clustering. A historical cost embedded in the model trades off the benefit of maintaining a consistent clustering over time with the cost of deviating from an accurate representation of the current data. Specifically, given an affinity matrix $S_{n\times n}$ (computed by $S = \phi(A)^T\phi(A)$, where A is the data matrix), ECKF

- *L. Wang and M. Dong are with the Machine Vision Pattern Recognition Lab, Department of Computer Science, Wayne State University, Detroit, MI 48202, USA.*
  E-mail: {ljwang,mdong}@wayne.edu
- *M. Rege is with the Department of Computer Science, Rochester Institute of Technology, Rochester, NY 14623, USA.*
- *Y. Ding is with the College of Information Sciences and Technology, Donghua University, Shanghai 201620, P. R. China.*

Digital Object Indentifier 10.1109/TKDE.2010.258

first computes a low-rank approximation matrix $\tilde{S} = \phi(\tilde{A})^T\phi(\tilde{A})$, as well generates a representative data subspace C_A. Next, the cluster centroids and indicators are obtained through the factorization: $\phi(\tilde{A}) = \phi(C_A)WG^T$, where W is the weight matrix, and G is the cluster indicator matrix. In ECKF, the cluster centroid matrix F equals $\phi(C_A)W$, and each of its vectors is a linear combination of exemplars in the kernel space; for G, each entry indicates the posterior probability of a data point to a certain cluster.

To the best of our knowledge, this is the first work that clusters large evolutionary datasets by the amalgamation of low-rank matrix approximation methods and matrix factorization based clustering. Different from other factorization-based clustering, ECKF works directly in the low-rank subspace, which has highly attractive properties in knowledge discovery such as feature selection, robustness to noise, and as a result, higher clustering accuracy. Since the low-rank approximation is a compact representation of the original matrix, and especially, the near-optimal low-rank approximation can preserve the sparsity of the original data, ECKF can analyze tremendous amounts of data by studying its "sketch", as a result, gain great computational efficiency. By monitoring the low-rank approximation errors at every time step, ECKF analyzes if the underlying structure of the data or the nature of the relationship between the data points has changed over different time steps. Based on this, a decision to either succeed the previous clustering or perform a new clustering is made. From a theoretical perspective, we prove the correctness and convergence of the ECKF algorithm, and provide detailed analysis with respect to its computational efficiency (both time and space).

The rest of the paper is organized as follows. We review the related work in Section 2. The ECKF model formulation and its iterative matrix factorization based evolutionary clustering algorithm are presented in Section 3. Theoretical analysis of ECKF is presented in Section 4. Extensive experimental results performed on synthetic and real world datasets appear in Section 5. Finally, we conclude in Section 6.

2 Related Work

In this Section, we provide some essential background on evolutionary clustering, matrix factorization based clustering, and low-rank matrix approximation, and review related work in the literature.

2.1 Evolutionary Clustering

Evolutionary clustering has been a relatively new topic and was first formulated by Chakrabarti et al. in [20]. They proposed heuristic solutions to evolutionary hierarchical clustering problems and evolutionary k-means clustering problems. Chi et al. [21] extended this work by proposing two evolutionary spectral clustering algorithms by incorporating a measure of *temporal smoothness* in the overall clustering quality. Evolutionary clustering differs from incremental clustering [22], [23] or constraint-based clustering [24], [25], which have been addressed in data mining research previously. Incremental clustering primarily addresses the issue of updating cluster centers [26], medoids [27] or hierarchical trees [28] when new data points arrive. Typically, the "new" arriving data points have no direct relationship with the "old" data points. [29] have proposed an algorithm for clustering moving objects. The spacial-temporal regularities of the moving objects are discovered by using micro-clustering [30]. An incremental spectral clustering algorithm is proposed in [31] to cluster evolving data points. Both these works try to achieve higher computational efficiency by compromising on the clustering quality. Constraint-based clustering on the other hand focuses on bringing in domain specific constraints to guide the clustering algorithm. Either hard constraints such as must-link or cannot-link constraints are enforced on the data points [32], [33] or soft constraints such as *a priori* knowledge [34] is incorporated into the clustering algorithm.

2.2 Matrix factorization based clustering

Matrix factorization based clustering approaches have received increasing attention owing to their applicability to high dimensional datasets. Singular Value Decomposition (SVD) [35] factorizes a matrix with the general form of $X \approx U\Sigma V^T$, where U is a unitary basis consisting of left-singular vectors of X, V is a unitary basis consisting of right-singular vectors of X, and Σ is a diagonal matrix with singular-values on the diagonal. Clustering is achieved by first projecting the data in the singular vector space, and then a traditional clustering algorithm is applied to the data points. In SVD, since matrices U and V are allowed to have negative entries, the projected data might have negative values in spite of the original data being positive. This can prevent the clustering results to be intuitive for applications such as documents or images that have a positive data input. Non-negative Matrix Factorization (NMF) [36], [37] is a linear, non-negative approximate data representation technique. The non-negative data matrix A is factorized into matrices F and G as $A \approx FG^T$, with the constraints that $F \in R^{d\times k}$ and $G \in R^{n\times k}$ are non-negative. Under the assumption of a certain distribution, it has been shown that NMF clustering is equivalent to the (kernel) k-means clustering, probabilistic latent semantic indexing (PLSI) [38] and the Laplacian-based spectral clustering [39]. The matrix F is considered to be a centroid matrix as every column represents a cluster center, while G is the cluster indicator matrix with G_{ik} denoting the posterior probability that data point i belongs to cluster k. Based on additional constraints, numerous variants of NMF based clustering have been proposed [40]–[42].

2.3 Low-rank matrix approximation

Low-rank matrix approximation methods have found profound application in diverse areas due to their ability to extract correlations and remove noise from matrix-structured data [43]. SVD is widely used for low-rank approximation and is known to achieve optimality in terms of the Frobenius norm [35]. The optimal low-rank approximation algorithms require super-linear time and large working sets [35] due to the fact that they perform repeated matrix-vector multiplications. Moreover, the result of SVD is usually dense even if the original matrix is sparse. Near optimal low-rank approximations, on the other hand, preserve sparsity of the data and hence achieve a better computational efficiency. For example, in Algorithm 844 [44], Gram-Schmidt method is applied in the pivoted QR decomposition to compute the sparse low-rank approximation. The algorithm selects one column at a time and with quasi-Gram-Schmidt step, produces a pivoted, Q-less PQR factorization. CUR [45] is another well-known matrix approximation method. It operates by first selecting the representative column and row exemplars as the left and right matrices according to their probability distributions and then computes the middle matrix based on these two matrices. Two other variants of CUR are CMD [46] and Colibri [47] approximation methods. CMD scales each unique sample up based on square root of the number of times it is in the initial subspace. The newly produced unique subspace has the same singular values and left singular vectors as the subspace produced by CUR. As a result, CMD can achieve equal accuracy as CUR but with less computational and space costs. Colibri on the other hand, proposes to solve the problem of incomplete basis. That is, the columns of the initial subspace may be linearly dependent or near duplicates, which is usually seen in a tightly-connected community where all nodes have similar neighbors. Consequently, with the redundant basis, the approximation is not efficient in terms of space. The Colibri algorithms are shown to be very efficient to process both static and dynamic graphs.

TABLE I
Symbol Definitions.

Symbol	Definition
$A_{d\times n}$	data matrix with the size of $d \times n$
$\tilde{A}_{d\times n}$	the approximation of the matrix A
$A^T, C^T, R^T ...$	the transpose of a matrix
$A(i,j)$	the entry(i,j) of A
$A(i,:)$	the i^{th} row of A
$A(:,j)$	the j^{th} column of A
c	the number of the selected columns
k_c	the number of the clusters
C_A	the representative column matrix of A
R_A	the right matrix of $\tilde{A}$
U_A	the middle matrix of $\tilde{A}$
I_c	the index set of the subspace ($C_A = A(:, I_c)$)
$W_{c\times k_c}$	the weight matrix with the size of $c \times k_c$
$G_{k_c\times n}$	the indicator matrix with the size of $k_c \times n$
$S_{n\times n}$	affinity matrix with the size of $n \times n$
$\tilde{S}$, S^T, $S(i,:)$, $S(:,j)$	similar definitions as for A
$C_{n\times c}$, $U_{c\times c}$, $R_{c\times n}$, I_c	similar definitions as for A
$X^{(t)}$	any matrix at t time step

3 Evolutionary Clustering with Low-rank Kernel Matrix Factorization

We now present ECKF, the kernel matrix factorization based framework to cluster large-scale evolving data (all symbols used are listed in Table I). At every time step, ECKF first gets a low-rank approximation of the affinity matrix. Next, we perform the factorization in a kernel space to yield the clustering.

3.1 Data dynamics and low-rank matrix approximation

One of the challenges of large-scale real world evolutionary datasets is the dynamic nature. Over different time steps, the data size (insertion and removal of samples) or the data structure (insertion and removal of clusters) might change. A low-rank approximation can efficiently detect the dynamics of data by examining the approximation errors over time [46].

The family of Colibri methods [47], i.e., *Colibri-S* (the version for static data) and *Colibri-D* (for dynamic data), compute the low-rank approximation to a matrix with a non-redundant subspace, and are proved to lose no accuracy compared to the best competitors, e.g., CUR [45] and CMD [46], while achieving significant savings in space and time. *Colibri-D* is specially designed for evolving data, which can quickly update the approximating subspace by leveraging the "smoothness" or similarity between two consecutive time steps. Moreover, for the same accuracy, *Colibri-D* is provably better or equal compared to CUR, CMD and *Colibri-S* in terms of speed. We employ the Colibri methods for efficiently selecting the representative subspace and generating the compact approximation of data similarities.

Given the affinity matrix S, *Colibri-S* first selects a column initial subspace C_0 by using the biased sampling method [45]. Then, a unique and independent subspace C is formed with an iterative procedure. Before starting this process, we initialize C with $C = C_0(:,1)$ and the core matrix U with $U = (C^TC)^{-1}$, where $(C^TC)^{-1}$ is the Moore-Penrose pseudo-inverse of the square matrix C^TC. In the following iteration, the i^{th} column in C_0 is checked to see if it is linearly dependent on the current columns of C. If not, this column is appended to C and the core matrix U is updated; else, this sample is discarded. Finally, C is obtained by eliminating all the redundant columns from C_0. Given the subspace matrix C, the approximation to the affinity matrix can be computed by $\tilde{S} = C(C^TC)^{-1}C^TS$. As proved in [47], the core matrix U satisfies $U = (C^TC)^{-1}$, so R is defined as C^TS in the algorithm, and the final approximation to S is $\tilde{S} = CUR$. In *Colibri-D*, given the updated matrix $S^{(t+1)}$, the columns in the initial subspace $C_0^{(t)}$ first split into two sets: one is for the unchanged samples at both time steps, and another represents the changed samples

between two time steps or the redundant samples at time t. Then, *Colibri-D* copies the samples of the first set to $C^{(t+1)}$ and checks those of the second set in the same way as *Colibri-S*. With more columns added in $C^{(t+1)}$, the core matrix $U^{(t+1)}$ is updated simultaneously. Finally, $R^{(t+1)}$ is computed by $R^{(t+1)} = {C^{(t+1)}}^T S^{(t+1)}$. Both methods guarantee the exemplars in the subspace are unique and linearly independent.

The common metric to measure the approximation error is the sum-square-error (SSE), defined as $\text{SSE} = \|S - \tilde{S}\|_F$, where $\|.\|_F$ is the Frobenius norm. The SSE value is typically dependent on the rank used in the approximation. In ECKF, with the fixed initial subspace size, SSE is computed with the same rank even though subspaces used may be smaller than the initial one. Suppose at $t-1$ time step, the low-rank approximation $\tilde{S}^{(t-1)}$, approximation error $\text{SSE}^{(t-1)}$ and the initial subspace $C_0^{(t-1)}$ are given. We first use *Colibri-D* based on $C_0^{(t-1)}$ to compute the current approximation $\tilde{S}^{(t)}$, and obtain the approximation error $\text{SSE}^{(t)}$. Compared with $\text{SSE}^{(t-1)}$, the change of $\text{SSE}^{(t)}$ can be simply evaluated by $\frac{|SSE^{(t-1)} - SSE^{(t)}|}{\min(SSE^{(t-1)}, SSE^{(t)})}$, predicting the following cases:

1) $\text{SSE}^{(t)}$ is almost the same as $\text{SSE}^{(t-1)}$: There are two possibilities: First, $n^{(t)} == n^{(t-1)}$, i.e., no nodes are added or deleted; Second, $n^{(t-1)} \neq n^{(t)}$, some nodes highly correlated with the existing samples are inserted or removed.
2) $\text{SSE}^{(t)}$ changes in a small range: This means although the data has evolved, the structure of the data has not changed.
3) $\text{SSE}^{(t)}$ increases significantly: This clearly indicates that the last initial subspace is not appropriate to be used as the representative subspace for computing the approximation at the current time. In other words, we need to perform resampling to obtain more exemplars for the new or existing clusters.

In the first case, the lack of change in SSE indicates that there was not much change in data as well. Consequently, we simply succeed the last clustering results without performing a new partition on the current data. In the last two cases, we need to get a new partition to track the groupings of evolving data. To estimate the cluster number, we examine the gaps between consecutive eigenvalues of the exemplar similarities. We determine if there are some clusters to be deleted by checking the volume of each cluster on the current data based on the last cluster membership. If a cluster has very small volume compared with that of the last time, we determine that cluster needs to be removed. Hence, in the second case, we use the new cluster number to yield the clustering. In the third case, we use Colibri-S to do re-sampling, as a result, we get a new representative subspace $C_0^{(t)}$ and "accurate" approximation. Then, a new partition is executed on the current data with the new estimated cluster number.

3.2 Model Formulation

We define the overall cost function of ECKF as the sum of snapshot quality and historical cost. To achieve a smooth clustering, we solve this problem by maximizing the clustering quality of the current snapshot and minimizing the historical cost as,

$$J = \min_{W^{(t)} \geq 0, G^{(t)} \geq 0} [\alpha \cdot \|\phi(\tilde{A})^{(t)} - \phi(C_A)^{(t)} W^{(t)} {G^{(t)}}^T\|_F^2 + (1-\alpha) \cdot \|\phi(\tilde{A})^{(t-1)} - \phi(C_A)^{(t-1)} W^{(t-1)} {G^{(t)}}^T\|_F^2], \quad (1)$$

where α is a user-defined parameter that trades-off between two costs. Notice the low-rank matrix approximation $\tilde{A}$ and the representative data subspace C_A in the kernel space are used in our clustering model, which leads to great improvement in clustering accuracy, as well as efficiency in both time and space.

To solve the above optimization problem, we propose an iterative algorithm to get $W^{(t)}$ and $G^{(t)}$. The updating rules are obtained by using the auxiliary functions and the optimization theory [42]. We let, $P_1 = {P^{(t)}}^T \tilde{S}^{(t)}$, $P_2 = P_1'$, $P_3 = {P^{(t)}}^T \tilde{S}^{(t)} P^{(t)}$, $P_4 = (\tilde{S}^{(t-1)})^T P^{(t-1)}$, and $P_5 = {P^{(t-1)}}^T \tilde{S}^{(t-1)} P^{(t-1)}$, where $\tilde{S}^{(t)}$ is the approximation of the affinity matrix at t, $P^{(t)}$ is the t time permutation matrix (${P^{(t)}}^T \tilde{S}^{(t)} = \phi(C_A)^{(t)T} \phi(\tilde{A})^{(t)}$), and similar definitions are made for $\tilde{S}^{(t-1)}$ and $P^{(t-1)}$, We can then split each matrix into the positive and negative parts, as follows,

$$P_i^+ = (|P_i| + P_i)/2; \quad P_i^- = (|P_i| - P_i)/2,\ i \in \{1,2,3,4,5\},$$

and derive the updating rules as,

$$W_{(i,h)}^{(t)} \leftarrow W_{(i,h)}^{(t)} \sqrt{\frac{((P_1)^+ G^{(t)} + (P_3)^- W^{(t)} {G^{(t)}}^T G^{(t)})_{(i,h)}}{((P_1)^- G^{(t)} + (P_3)^+ W^{(t)} {G^{(t)}}^T G^{(t)})_{(i,h)}}}, \quad (2)$$

$$G_{(i,h)}^{(t)} \leftarrow G_{(i,h)}^{(t)} \sqrt{\frac{(M_1 + N_1 + M_3 + N_3)_{(i,h)}}{(M_2 + N_2 + M_4 + N_4)_{(i,h)}}}, \quad (3)$$

where

$$M_1 = \alpha {P_2}^+ W^{(t)}, \qquad M_3 = \alpha G^{(t)} {W^{(t)}}^T {P_3}^- W^{(t)},$$
$$M_2 = \alpha {P_2}^- W^{(t)}, \qquad M_4 = \alpha G^{(t)} {W^{(t)}}^T {P_3}^+ W^{(t)},$$
$$N_1 = (1-\alpha) {P_4}^+ W^{(t-1)},$$
$$N_2 = (1-\alpha) {P_4}^- W^{(t-1)},$$
$$N_3 = (1-\alpha) G^{(t)} {W^{(t-1)}}^T {P_5}^- W^{(t-1)},$$
$$N_4 = (1-\alpha) G^{(t)} {W^{(t-1)}}^T {P_5}^+ W^{(t-1)}.$$

Note that in Equation (1), the subspaces between two time steps ($\phi(C_A)^{(t)}$ and $\phi(C_A)^{(t-1)}$) are not necessarily the same in our model. Consequently, the weights between the two steps will not correspond to each other. To solve the optimization problem by using matrix factorization, we must correspond $\phi(\tilde{A})^{(t-1)}$ to $\phi(\tilde{A})^{(t)}$ to guarantee they have the same size. The same holds true for $\phi(C_A)^{(t)} W^{(t)}$ and $\phi(C_A)^{(t-1)} W^{(t-1)}$. From Equations (2) and (3), we can see that the operators are based on the

elements of the matrices. That is, if we can correspond the nodes and clusters at the current time with those at the last time, we will arrive at our goal. Specifically, assume the cluster numbers are $k^{(t-1)}$ and $k^{(t)}$ at $t-1$ and t time steps, respectively, when

1) $k^{(t-1)} < k^{(t)}$, the extra clusters at t time must be computed only on the current data, so we let $W^{(t-1)} = [W^{(t-1)}, \quad \mathbf{0}_{c\times(k^{(t-1)}+1:k^{(t)})}]$.
2) $k^{(t-1)} > k^{(t)}$, we simply remove the deleted cluster weight vectors in $W^{(t-1)}$.

To correspond the nodes between two time steps, we construct a permutation matrix $P_{t-1,t}$, i.e., one is assigned to an element that corresponds to a node in the current data in each row of $P_{t-1,t}$, thus $\tilde{S}^{(t-1)}P_{t-1,t}^T$ and $\tilde{S}^{(t)}$ have equal sizes. Consequently, in Equation (3), we only rewrite

$$N_1 = (1-\alpha)P_{t-1,t}{P_4}^+W^{(t-1)},$$
$$N_2 = (1-\alpha)P_{t-1,t}{P_4}^-W^{(t-1)}.$$

3.3 Algorithm Derivation

In ECKF, we initialize the variable factors using the previous clustering results instead of using random values. This strategy not only smooths the clustering results between consecutive time steps, but also improves the clustering efficiency.

Suppose the two time steps data, $S^{(t-1)}$ and $S^{(t)}$, are given, as well as the clustering results at $t-1$ time step, including $\tilde{S}^{(t-1)}$, $I_c^{(t-1)}$, $W^{(t-1)}$ and $G^{(t-1)}$. Now, we need to obtain $W^{(t)}$ and $G^{(t)}$, as well as the new representative subspace and approximation. We first use Colibri methods to get $I_c^{(t)}$ and $\tilde{S}^{(t)}$, as well as two disjoint subsets $I_a^{(t)}$ and $I_b^{(t)}$ ($I_c^{(t)} = I_a^{(t)} \cup I_b^{(t)}$), so that the elements in $I_a^{(t)}$ correspond to those unchanged selected samples from $t-1$ to t, while the items in $I_b^{(t)}$ correspond to those changed or unselected samples between the two time steps. To obtain these subsets, we only need to compare the corresponding columns in $S^{(t-1)}$ and $S^{(t)}$.

Now, we make a reasonable assumption: if a cluster centroid is a weighted combination of unchanged exemplars at $t-1$ time step, then it is also a cluster centroid at time t. Hence, we initialize $W^{(t)}$ as follows: for each column of $W^{(t)}$, we check the non-zero values to see if the indices of their corresponding exemplars are all in $I_a^{(t)}$; if they are, we copy the corresponding weights from $W^{(t-1)}$ to $W^{(t)}$; otherwise, we initialize the vector with random values. Actually, we can update $W^{(t)}$ more softly in this way: for a column of $W^{(t)}$, if all the large non-zero values, for example the ones greater than 0.2, correspond to unchanged exemplars, we will copy the relevant weights of $W^{(t-1)}$ to $W^{(t)}$; otherwise, we initialize it with random values. In particular, if a large non-zero weight corresponds to an exemplar that is not selected at t time step, we just ignore it. That is, in the proposed algorithm, we only examine the exemplars at t^{th} subspace. We initialize $G^{(t)}$ according to $W^{(t)}$: if $W^{(t)}$ is succeeded from $W^{(t-1)}$, then we pick a set of data points belonging to that cluster at $t-1$ step and copy the corresponding indicators from $G^{(t-1)}$ to $G^{(t)}$.

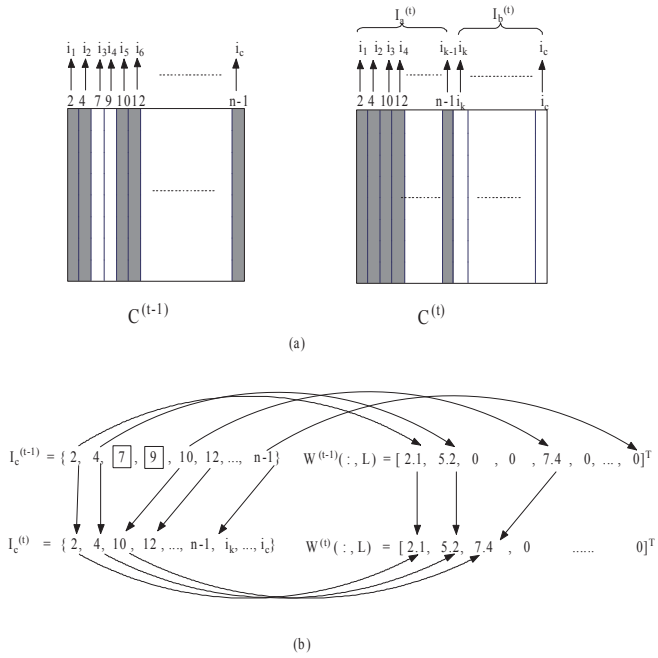


Fig. 1. An example of fast updating weights: (a) shows the process of splitting the index set $I_c^{(t)}$ into $I_a^{(t)}$ ($\{2, 4, 10, 12, ..., n-1\}$) and $I_b^{(t)}$ ($\{7, 9, ...\}$), where gray cells represent unchanged columns and blank cells represent changed columns; (b) shows the process of updating $W^{(t)}$ according to $I_a^{(t)}$ and $W^{(t-1)}$.

Algorithm 1 Fast Updating

INPUT: $W^{(t-1)}$, $G^{(t-1)}$, $I_a^{(t)}$ and $k_c^{(t)} \in \mathbf{Z}^+$ s.t. $1 \leq k_c^{(t)} \leq n^{(t)}$
OUTPUT: $W^{(t)}$, $G^{(t)}$
METHOD:

1) Initialize $W^{(t)}$ and $G^{(t)}$ with random values
2) Update $W^{(t)}$ and $G^{(t)}$:

$$\begin{aligned}
&for\ L = 1 : k_c^{(t)}\\
&\quad if\{i_k | W^{(t-1)}(i_k, L) \neq 0\} \subseteq I_a^{(t)}\\
&\qquad W^{(t)}(:, L) \leftarrow W^{(t-1)}(:, L)\\
&\qquad I_L = \{i_k | Indicator(G^{(t-1)})_{i_k} == L\}\\
&\qquad G^{(t)}(I_L, :) = G^{(t-1)}(I_L, :)\\
&\quad end\\
&end
\end{aligned}$$

3) Return $W^{(t)}$ and $G^{(t)}$.

In Figure 1, we provide an example to explain the updating process. At first, we compute $C^{(t)}$ using Colibri methods. Then we get $I_a^{(t)}$ ($\{2, 4, 10, 12, ..., n-1\}$) and $I_b^{(t)}$ ($\{7, 9, ...\}$) according to $S^{(t-1)}$ and $S^{(t)}$. As observed from Figure 1, we learn in the L^{th} vector of $W^{(t-1)}$, the

Algorithm 2 ECKF

INPUT: $\tilde{S}^{(t-1)}$, $I_c^{(t-1)}$, $W^{(t-1)}$, $G^{(t-1)}$, $S^{(t)}$, $k_c^{(t-1)} \in \mathbf{Z}^+$ s.t. $1 \leq k_c^{(t-1)} \leq n^{(t-1)}$
OUTPUT: $\tilde{S}^{(t)}$, $W^{(t)}$, $G^{(t)}$, $I_c^{(t)}$ and $k_c^{(t)} \in \mathbf{Z}^+$
METHOD:

1) Use the Colibri method to get $\tilde{S}^{(t)}$, $I_c^{(t)}$, $I_a^{(t)}$ and $I_b^{(t)}$;
2) If we need to do a new partition, go to 3 and 4; else, let $W^{(t)} = W^{(t-1)}$ and $G^{(t)} = G^{(t-1)}$, and return;
3) Determine the cluster number $k_c^{(t)}$;
4) Construct permutation matrices $P^{(t-1)}$ and $P^{(t)}$;
5) Initialize $W^{(t)}$ and $G^{(t)}$ using the Fast Update algorithm;
6) Correspond the nodes and clusters between two time steps and obtain $W^{(t-1)}$ and $P_{t-1,t}$;
7) Iterate by using the updating rules defined in Equations (2) and (3) until convergence.

weights of columns $\{2, 4, 10\}$ are non-zeros, and these exemplars belong to $I_a^{(t)}$, so we update the L^{th} vector of $W^{(t)}$ by copying the relevant weights of $W^{(t-1)}(:, L)$ to $W^{(t)}(:, L)$. Since L^{th} column vector of $W^{(t)}$ is succeeded from $W^{(t-1)}$ and it corresponds to a cluster, we select the set of data points (indexed by I_L) belonging to L^{th} cluster at $t-1$ time and let $G^{(t)}(I_L, :) = G^{(t-1)}(I_L, :)$. In Algorithm 1, we provide the details of the initialization algorithm, while Algorithm 2 presents the complete kernel based evolutionary clustering algorithm.

4 Theoretical Analysis

4.1 Correctness

Proposition 1 (Correctness of ECKF): Given the objective function of Equation (1), the constrained solution satisfies KKT complementary conditions under the updating rules in Equations (2) - (3).

Proof: To solve the optimization problem, we introduce the Lagrangian function

$$\begin{aligned}
&L(W^{(t)}, G^{(t)}, \lambda_1, \lambda_2) \\
&= \alpha \|\phi(\tilde{A})^{(t)} - \phi(C_A)^{(t)} W^{(t)} {G^{(t)}}^T\|_F^2 + (1-\alpha)\|\phi(\tilde{A})^{(t-1)} \\
&\quad - \phi(C_A)^{(t-1)} W^{(t-1)} {G^{(t)}}^T\|_F^2 - \mathrm{Tr}(\lambda_1 {W^{(t)}}^T) \\
&\quad - \mathrm{Tr}(\lambda_2 {G^{(t)}}^T) \\
&= \alpha \cdot \mathrm{Tr}[\tilde{S}^{(t)} - (\tilde{S}^{(t)})^T P^{(t)} W^{(t)} {G^{(t)}}^T - G^{(t)} {W^{(t)}}^T {P^{(t)}}^T \tilde{S}^{(t)} \\
&\quad + G^{(t)} {W^{(t)}}^T {P^{(t)}}^T \tilde{S}^{(t)} P^{(t)} W^{(t)} {G^{(t)}}^T] + (1-\alpha) \cdot \mathrm{Tr}[\tilde{S}^{(t-1)} \\
&\quad - (\tilde{S}^{(t-1)})^T P^{(t-1)} W^{(t-1)} {G^{(t)}}^T - G^{(t)} {W^{(t-1)}}^T {P^{(t-1)}}^T \\
&\quad \cdot \tilde{S}^{(t-1)} + G^{(t)} {W^{(t-1)}}^T {P^{(t-1)}}^T \tilde{S}^{(t-1)} P^{(t-1)} W^{(t-1)} {G^{(t)}}^T] \\
&\quad - \mathrm{Tr}(\lambda_1 {W^{(t)}}^T) - \mathrm{Tr}(\lambda_2 {G^{(t)}}^T) \qquad (4)
\end{aligned}$$

where $\tilde{S}^{(t)} = \phi(\tilde{A})^{(t)^T} \phi(\tilde{A})^{(t)}$, λ_1 and λ_2 are Lagrangian multipliers with nonnegative values, which constrain the nonnegativity of $W^{(t)}$ and $G^{(t)}$ respectively. This function satisfies KKT complementary conditions. By setting the gradients, i.e., $\frac{\partial L}{\partial W^{(t)}}$ and $\frac{\partial L}{\partial G^{(t)}}$, as zeros, we obtain the following equations from the complementary conditions:

$$\begin{aligned}
&[-2\alpha \cdot {P^{(t)}}^T \tilde{S}^{(t)} G^{(t)} + 2\alpha \cdot {P^{(t)}}^T \tilde{S}^{(t)} P^{(t)} W^{(t)} {G^{(t)}}^T G^{(t)}]_{(i,h)} \\
&\cdot {W^{(t)}}_{(i,h)} = \lambda_1 {W^{(t)}}_{(i,h)} = 0, \qquad (5)
\end{aligned}$$

$$\begin{aligned}
&[-2\alpha \cdot (\tilde{S}^{(t)})^T P^{(t)} W^{(t)} + 2\alpha \cdot G^{(t)} {W^{(t)}}^T {P^{(t)}}^T \tilde{S}^{(t)} P^{(t)} W^{(t)} \\
&- 2(1-\alpha) \cdot (\tilde{S}^{(t-1)})^T P^{(t-1)} W^{(t-1)} + 2(1-\alpha) \cdot G^{(t)} {W^{(t-1)}}^T \\
&{P^{(t-1)}}^T \tilde{S}^{(t-1)} P^{(t-1)} W^{(t-1)}]_{(i,h)} {G^{(t)}}_{(i,h)} = \lambda_2 {G^{(t)}}_{(i,h)} \\
&= 0. \qquad (6)
\end{aligned}$$

These are fixed point equations, and the solutions must eventually converge to a stationary point. From the above two equations, we derive another two equal equations:

$$\begin{aligned}
&[-2\alpha \cdot {P^{(t)}}^T \tilde{S}^{(t)} G^{(t)} + 2\alpha \cdot {P^{(t)}}^T \tilde{S}^{(t)} P^{(t)} W^{(t)} {G^{(t)}}^T G^{(t)}]_{(i,h)} \\
&{W^{(t)}}^2_{(i,h)} = 0, \qquad (7)
\end{aligned}$$

$$\begin{aligned}
&[-2\alpha \cdot (\tilde{S}^{(t)})^T P^{(t)} W^{(t)} + 2\alpha \cdot G^{(t)} {W^{(t)}}^T {P^{(t)}}^T \tilde{S}^{(t)} P^{(t)} W^{(t)} \\
&- 2(1-\alpha) \cdot (\tilde{S}^{(t-1)})^T P^{(t-1)} W^{(t-1)} + 2(1-\alpha) \cdot G^{(t)} {W^{(t-1)}}^T \\
&{P^{(t-1)}}^T \tilde{S}^{(t-1)} P^{(t-1)} W^{(t-1)}]_{(i,h)} {G^{(t)}}^2_{(i,h)} = 0. \qquad (8)
\end{aligned}$$

The constrained solution with updating rules in Equations (2) and (3) satisfy Equations (7) and (8), so it satisfies the KKT fixed point condition. The proof is completed. □

4.2 Convergence

Proposition 2 (Convergence of ECKF): The object function of Equation (1) is monotonically decreasing under the updating rules in Equations (2) - (3).

Proof: We construct auxiliary functions to prove that Equation (1) decreases monotonically under the updating rules.

An auxiliary function $Z(X^{t+1}, X^t)$ should satisfy the two conditions:

$$Z(X^{t+1}, X^t) \geq J(X^{t+1}), \qquad Z(X^t, X^t) = J(X^t), \quad (9)$$

for any X^{t+1} and X^t. We define,

$$X^{t+1} = \min_X Z(X, X^t). \qquad (10)$$

Then, we obtain the following equations:

$$J(X^t) = Z(X^t, X^t) \geq Z(X^{t+1}, X^t) \geq J(X^{t+1}). \quad (11)$$

Thus, with a proper auxiliary function, $J(X^t)$ is decreasing monotonically. Now, we construct the auxiliary functions with respect to $W^{(t)}$ and $G^{(t)}$.

Let,

$$X = W^{(t)},$$
$$B = \phi(C_A)^{(t)^T}\phi(\tilde{A})^{(t)}G^{(t)} = P^{(t)^T}\tilde{S}^{(t)}G^{(t)},$$
$$H = \phi(C_A)^{(t)^T}\phi(C_A)^{(t)} = P^{(t)^T}\tilde{S}^{(t)}P^{(t)},$$
$$Q = G^{(t)^T}G^{(t)},$$

then the objective function with $W^{(t)}$ is,

$$J(X) = \text{Tr}(-2\alpha \cdot X^T B + \alpha \cdot X^T HXQ). \tag{12}$$

Since B and H are mixed-sign matrices, we rewrite $J(X)$ by splitting each mixed-sign matrix into positive and negative parts:

$$J(X) = \text{Tr}(-2\alpha \cdot X^T B^+ + 2\alpha \cdot X^T B^- + \alpha \cdot X^T H^+ XQ - \alpha \cdot X^T H^- XQ). \tag{13}$$

According to the Proposition 3 in [42], we find the upper bounds for each item in Equation (13) by,

$$a \leq (a^2+b^2)/2b \quad \text{for} \quad \text{Tr}(X^T B^-);$$
$$\text{Tr}(D^T EDF) \leq \sum_{i=1}^{n}\sum_{p=1}^{k}\frac{(ED^TF)_{ip}D_{ip}^2}{D_{ip}^T} \text{ for } \text{Tr}(X^T H^+ XQ);$$
$$1+\log z \leq z \quad \text{for} \quad \text{Tr}(X^T B^+),\ \text{Tr}(X^T H^- XQ).$$

Then, we construct an auxiliary function for $J(X)$ as:

$$Z(X', X) = -2\sum_{ik} B_{ik}^+ X'_{ik}(1 + log\frac{X_{ik}}{X'_{ik}}) + \sum_{ik} B_{ik}^- \frac{X_{ik}^2 + X_{ik}'^2}{X'_{ik}} + \sum_{ik}\frac{(H^+X'Q)_{ik}X_{ik}^2}{X'_{ik}} - \sum_{ijkl} H_{ij}^- X'_{jk}Q_{kl}X'_{il}(1 + log\frac{X_{jk}X_{il}}{X'_{jk}X'_{il}}), \tag{14}$$

where the subscription ik is the index of an element in the matrix. To get its global minimum, we take $\frac{\partial Z(X',X)}{\partial X_{ik}} = 0$ and get Equation (2).

Similarly, when

$$X = G^{(t)}$$
$$B_1 = (\phi(\tilde{A})^{(t)})^T\phi(C_A)^{(t)}W^{(t)} = (\tilde{S}^{(t)})^T P^{(t)}W^{(t)},$$
$$H_1 = W^{(t)^T}\phi(C_A)^{(t)^T}\phi(C_A)^{(t)}W^{(t)} = W^{(t)^T}P^{(t)^T}\tilde{S}^{(t)}P^{(t)}W^{(t)},$$
$$B_2 = (\phi(\tilde{A})^{(t-1)})^T\phi(C_A)^{(t-1)}W^{(t-1)} = (\tilde{S}^{(t-1)})^T P^{(t-1)}W^{(t-1)},$$
$$H_2 = W^{(t-1)^T}\phi(C_A)^{(t-1)^T}\phi(C_A)^{(t-1)}W^{(t-1)} = W^{(t-1)^T}P^{(t-1)^T}\tilde{S}^{(t-1)}P^{(t-1)}W^{(t-1)},$$

the objective function with $G^{(t)}$ is,

$$J(X) = \alpha \cdot \text{Tr}(-2X^T B_1^+ + 2X^T B_1^- + XH_1^+ X^T - XH_1^- X^T) + (1-\alpha)\cdot \text{Tr}(-2X^T B_2^+ + 2X^T B_2^- + XH_2^+ X^T - XH_2^- X^T). \tag{15}$$

So, its auxiliary function is

$$Z(X', X)$$
$$= -2\alpha\sum_{ik} B_{1ik}^+ X'_{ik}(1 + log\frac{X_{ik}}{X'_{ik}}) + \alpha\sum_{ik} B_{1ik}^- \frac{X_{ik}^2 + X_{ik}'^2}{X'_{ik}}$$
$$+ \alpha\sum_{ik}\frac{(H_1^+X')_{ik}X_{ik}^2}{X'_{ik}} - \alpha\sum_{ikl}(H_{1kl}^- X'_{ik}X'_{il}(1 + log\frac{X_{ik}X_{il}}{X'_{ik}X'_{il}})$$
$$- 2(1-\alpha)\sum_{ik} B_{2ik}^+ X'_{ik}(1 + log\frac{X_{ik}}{X'_{ik}}) + (1-\alpha)\sum_{ik}(B_{2ik}^- \frac{X_{ik}^2 + X_{ik}'^2}{X'_{ik}}) + (1-\alpha)\sum_{ik}\frac{(H_2^+X')_{ik}X_{ik}^2}{X'_{ik}} - (1-\alpha)$$
$$\sum_{ikl}(H_{2kl}^- X'_{ik}X'_{il}(1 + log\frac{X_{ik}X_{il}}{X'_{ik}X'_{il}})). \tag{16}$$

By taking $\frac{\partial Z(X',X)}{\partial X_{ik}} = 0$, we get the updating rule of Equation (3) for $G^{(t)}$. The proof is completed. □

In the ECKF model, the input is required to be a semi-positive definite matrix so that it can be used as the kernel matrix. Assuming the affinity matrix is semi-positive definite, we prove the computed low-rank approximation is also semi-positive definite in Proposition 3.

Proposition 3: Given an semi-positive definite matrix X, the approximation $\tilde{X}$ computed by Colibri methods is semi-positive definite.

Proof: According to Colibri methods, if we concatenate all the sampled columns into C, the approximation is computed by:

$$\tilde{X} = C(C^TC)^{-1}C^TX, \tag{17}$$

where $(C^TC)^{-1}$ is the Moore-Penrose pseudo-inverse of (C^TC). This equation indicates, $\tilde{X}$ is the projection of X into the column space of C. Similarly, we can construct $\tilde{X}$ as the projection of X into the row space of R, then we have $\tilde{X} = XR^T(RR^T)^{-1}R$. Since X is semi-positive definite, we have $X = X^T$, so $C = R^T$, and we obtain:

$$\tilde{X} = C(C^TC)^{-1}C^TX = XR^T(RR^T)^{-1}R = XC(C^TC)^{-1}C^T. \tag{18}$$

Let $B = C(C^TC)^{-1}C^T$, then we get:

$$(C^TC) \quad \text{is semi-positive definite}$$
$$\Longrightarrow (C^TC)^{-1} \quad \text{is semi-positive definite,}$$
$$\Longrightarrow C(C^TC)^{-1}C^T \quad \text{is semi-positive definite,}$$

since for any vector $\mathbf{v}$, $\mathbf{v}[C(C^TC)^{-1}C^T]\mathbf{v}^T = \mathbf{y}(C^TC)^{-1}\mathbf{y}^T \geq 0$. Hence, B is semi-positive definite. Then, Equation (18) is rewritten as $\tilde{X} = BX = XB$. According to the property of a semi-positive definite matrix, if X and B are both semi-positive definite and $BX = XB$, then BX and XB are also semi-positive definite. Thus, $\tilde{X}$ is semi-positive definite. The proof is completed. □

4.3 Time and Space Complexity

To cluster a large dataset, efficiency in both space and speed is essential. In Algorithm 2, the near-optimal matrix approximation is highly efficient, much faster than SVD [47]. In the decomposition step, the computation is actually done using the three small matrices, C, U and R. This is also the basis for our time analysis.

For matrix decomposition, we first need to compute P_i $(i \in 1, 2, 3, 4, 5)$ with the following time:

$$P_1, P_2, P_3 : \mathcal{O}({c_1}^2 n_1),$$
$$P_4, P_5 : \mathcal{O}({c_0}^2 n_0),$$

where c_1 and c_0 are the sizes of data subspaces at t and $t-1$ time steps, respectively; and n_1 and n_0 are the sizes of the input matrices at t and $t-1$. Then, we need to compute $W^{(t)}$ and $G^{(t)}$ in Equations (2) and (3). With one iteration the time is,

$$W^{(t)} : \mathcal{O}(c_1 n_1 k_c),$$
$$G^{(t)} : \mathcal{O}(cnk_c),$$

where k_c is the cluster number at t time step, $c = max(c_1, c_0)$ and $n = max(n_1, n_0)$. P_i $(i \in 1, 2, 3, 4, 5)$ is computed only once, but $W^{(t)}$ and $G^{(t)}$ have to be calculated l times, so the overall time complexity is $\mathcal{O}(c^2 n + lcnk_c)$.

Regarding the space complexity, ECKF needs $2cn + c^2$ units to store C, U and R, and needs $c_1 k_c$ and $n_1 k_c$ units for $W^{(t)}$ and $G^{(t)}$, respectively. In addition, the temporal storage for computing P_i and updating $W^{(t)}$ and $G^{(t)}$ requires $\mathcal{O}(cn)$ units. Since $c \ll n$, the total space used is $\mathcal{O}(cn)$.

The space requirement of storing the approximation and that of running the clustering algorithm are of the same order. That is, the overall space complexity of ECKF is the same as the storage requirement for the approximation matrix. Thus, we use the relative storage of the input matrix as the space cost in our experiments. Moreover, we also note that the size of data subspace c plays an important role in determining the time and space costs. The lower c is, the less running time and space ECKF uses.

5 Experiments and Results

In this Section, we evaluate the performance of ECKF on both synthetic and two real-world evolving datasets in terms of clustering accuracy, time and space costs by comparing it with leading evolutionary clustering methods. In addition, we investigate the results gained by the low-rank approximation, which indicate the change of data structure. All algorithms were implemented using MATLAB 7. The experiment was performed on a machine with a 3.0GHz Intel Xeon CPU, 3.0GB RAM and the Windows XP operating system. All the results reported are averaged over 10 runs.

5.1 Evaluation methods

ECKF performs low-rank approximation by employing the Colibri methods [47]. Based on the specification of the parameter c_{max}, which is the maximum volume of the subspace, Colibri methods can produce a unique and linearly independent subspace. In our experiments, we set $c_{max} = 200$ for both synthetic and real datasets. To study the performance of the low-rank approximation, we examine the reconstruction error:

$$\text{SSE} = \|S - \tilde{S}\|_F = \sqrt{\sum_{i,j} (S(i,j) - \tilde{S}(i,j))^2}. \quad (19)$$

When the dataset is extremely large, direct computation of SSE by using Equation (19) is expensive. In this case, we employ the approximation method [46] to estimate SSE:

$$\tilde{\text{SSE}} = \sqrt{\frac{n^2}{|\mathcal{C}|} \sum_{(i,j)\in\mathcal{C}} (S(i,j) - \tilde{S}(i,j))^2}, \quad (20)$$

where $\mathcal{C}$ is a set of randomly selected sample entries, and n is the size of S.

We compare the proposed method with evolutionary spectral clustering [21] (ENC) and accumulated K-means (AccKM, the accumulation of the history and present data is used as input for k-means). To show the effectiveness of incorporating historical knowledge in the ECKF model, we also compare it with the kernel matrix factorization based clustering (KMF), which executes the algorithm on the present time data only (Notice that ECKF and KMF will gain the same results on the first time data).

All our comparisons are conducted using the following evaluation metrics:

1) Normalized mutual information (NMI) [48] computed from the confusion matrix based on the true and predicted cluster labels,

$$NMI = \frac{\sum_{h=1}^{k^{(a)}} \sum_{l=1}^{k^{(b)}} n_{h,l} log(\frac{n \cdot n_{h,l}}{n_h^{(a)} n_l^{(b)}})}{\sqrt{(\sum_{h=1}^{k^{(a)}} n_h^{(a)} log\frac{n_h^{(a)}}{n})(\sum_{l=1}^{k^{(b)}} n_l^{(b)} log\frac{n_l^{(b)}}{n})}}, \quad (21)$$

where $k^{(a)}$ and $k^{(b)}$ are cluster numbers in the true and predicted clusterings, $n_{h,l}$ denotes the element in the confusion matrix, $n_h^{(a)}$ is the number of objects in the h^{th} cluster according to the true clustering and $n_l^{(b)}$ has similar definition. NMI ranges in $[0, 1]$. A high NMI value indicates that the predicted clustering matches the ground truth well.

2) KMCost [21], a value computed by K-means cost function:

$$\text{KMCost} = \alpha \sum_{l=1}^{k_c} \sum_{i \in \mathcal{V}_{l,t}} \| \vec{v}_{i,t} - \vec{\mu}_{l,t} \|^2 + (1-\alpha) \sum_{l=1}^{k_c} \sum_{i \in \mathcal{V}_{l,t}} \|\vec{v}_{i,t-1} - \vec{\mu}_{l,t-1}\|^2, \quad (22)$$

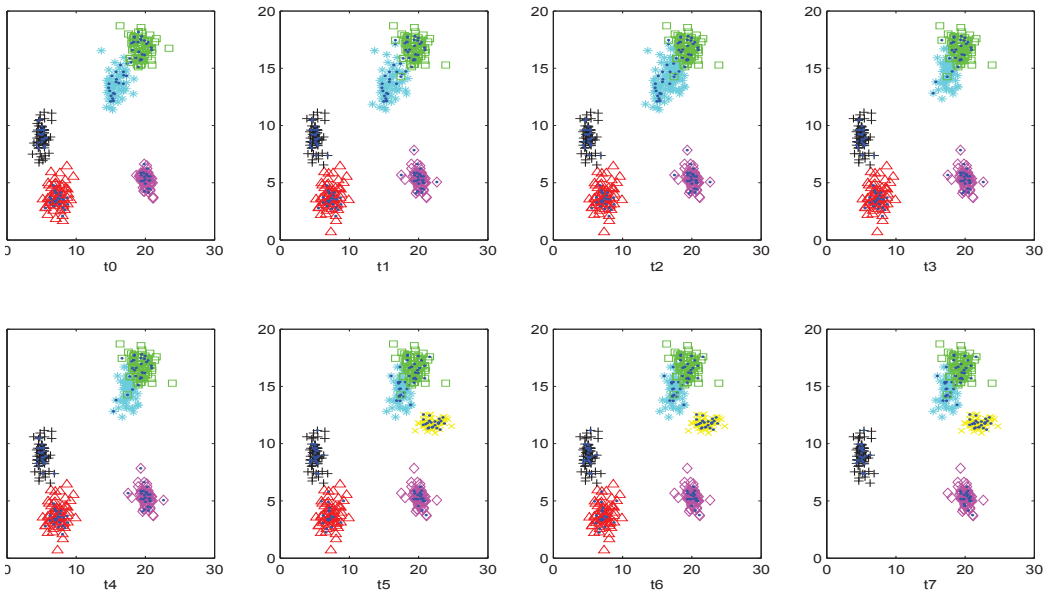


Fig. 2. Exemplar selection for evolving synthetic data (eight time steps from t0-t7): blue dots inside each cluster denote the selected exemplars.

where $\vec{v}_{i,t}$ denotes a node in the l^{th} cluster at t time, $\vec{\mu}_{l,t}$ is the mean of that cluster, and the similar definitions are made for $\vec{v}_{i,t-1}$ and $\vec{\mu}_{l,t-1}$. Notice that for the second term, we evaluate t time clustering on the $t-1$ data. A lower value indicates better clustering.

3) Time cost: We use the inline functions of MATLAB, tic and toc, to compute the running time.
4) Space cost: We compute the storage of the input matrix as the space cost (normalized based on the required storage units of the original data matrix). For ECKF, the input is the low-rank approximations of two period similarity matrices, thus the space cost is given by,

$$\text{SPCost} = \frac{N_1 + N_2}{\text{NNZ}(S^{(t)}) + \text{NNZ}(S^{(t-1)})}, \tag{23}$$
$$N_1 = \text{NNZ}(C^{(t)}) + \text{NNZ}(U^{(t)}) + \text{NNZ}(R^{(t)}),$$
$$N_2 = \text{NNZ}(C^{(t-1)}) + \text{NNZ}(U^{(t-1)}) + \text{NNZ}(R^{(t-1)})$$

where $\text{NNZ}(.)$ presents the number of non-zero entries in a matrix, For KMF, the input is the approximation of present data, so the space cost is $\text{SPCost} = \frac{\text{NNZ}(C^{(t)})+\text{NNZ}(U^{(t)})+\text{NNZ}(R^{(t)})}{\text{NNZ}(S)}$. Both, ENC and AccKM, take S as the input, and their normalized space cost is one.

5.2 Synthetic data

We evaluated the performance of ECKF on evolving synthetic data by considering the following cases: adding noise, inserting new nodes to existing clusters, deleting nodes, inserting new clusters, removing clusters and duplicating snapshots (in this case, our algorithm can automatically detect duplicates and copy the last clustering as the current one).

We now describe the synthetic data we generated consisting of eight time steps. For the first time step (i.e., $t0$), we sampled data points from five two-dimensional Gaussian distributions $\mathcal{N}(\mu_i, \Sigma_i)$, $i \in \{1..5\}$, where Σ_i is restricted to a diagonal matrix. Specifically, each entry of μ_i has a value randomly chosen from 0 to 20, and that of Σ_i ranging from 1 to 2. We sampled 60 data points from each distribution and obtained a five-cluster dataset with size 2×300. For $t1$, we randomly selected 50 data points from each cluster and added zero-mean gaussian noise. For $t2$, we selected one cluster and inserted some new data points into it. For $t3$, we selected one cluster and deleted some nodes from it. For $t4$, we generated a duplicate snapshot of $t3$. For $t5$, we generated a group of nodes from a gaussian distribution in the same way as for the former clusters, and inserted it as a new cluster. For $t6$, we duplicated the last snapshot again. For $t7$, we selected a cluster and deleted it. In Figure 2, each time step data is shown with each group having a unique color and shape. The affinity matrix is computed by the kernel $S(i,j) = e^{\frac{-\|A(:,i)-A(:,j)\|^2}{\sigma}}$, where $\sigma = 0.05 \times max(A(:))$.

We first examine the results of the low-rank approximating procedure. As can be seen from Figure 2, the exemplars (representative points) selected by ECKF illustrated by blue dots, nicely represent the underlying data distribution, i.e., no outliers are picked. In addition, we have also plotted SSEs gained from Colibri methods

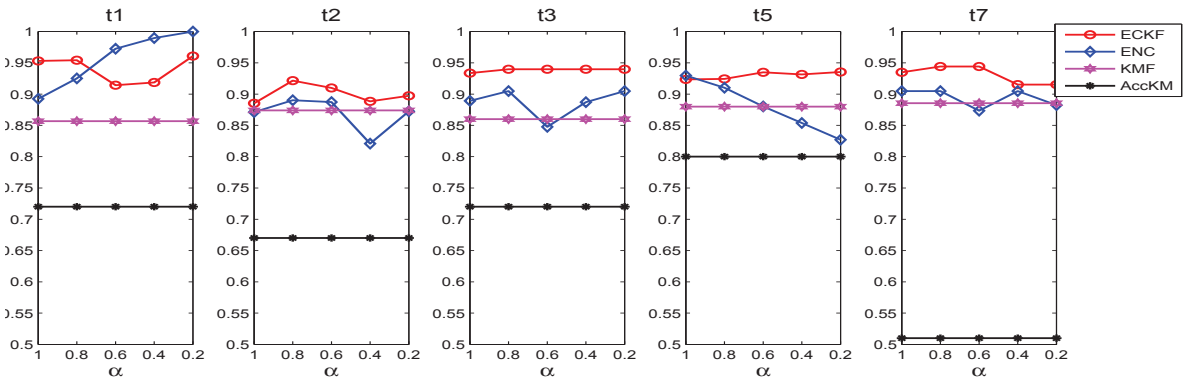
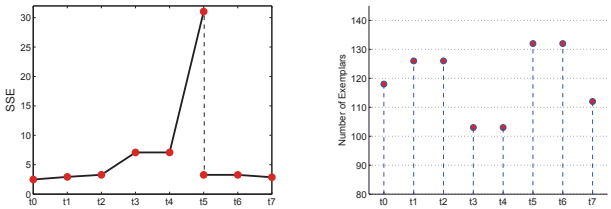


Fig. 4. Comparison of NMI among four methods with different values of α ranging in $[0.2, 1]$.



(a) (b)

Fig. 3. Low-rank approximation on synthetic datasets: (a) shows approximation error at each time step, and (b) shows the number of selected exemplars at each time step.

TABLE II
The gaps between top eigenvalues and numbers of the deleted cluster on synthetic data.

Time Steps	$t0$	$t1$	$t2$	$t3$	$t5$	$t7$
Eigenvalue Gaps	***13.81***	***15.18***	***15.18***	***15.18***	***15.50***	***29.60***
	148.08	***157.24***	***135.32***	***151.34***	***15.18***	***34.27***
	11.14	***14.29***	***16.73***	***20.19***	***151.34***	***60.21***
	4.08	***16.37***	***19.48***	***68.44***	***2.82***	***87.90***
	84.37	***55.52***	***54.02***	***17.13***	***17.37***	***20.26***
	6.14	*6.50*	*14.26*	*1.82*	***68.44***	*16.16*
	11.79	*8.99*	*19.10*	*0.77*	*17.13*	*19.41*
	7.39	*0.77*	*0.77*	*8.96*	*1.82*	*7.43*
	.	.	.	.	.	.
	.	.	.	.	.	.
	.	.	.	.	.	.
Deleted clusters	*0*	*0*	*0*	*0*	*0*	*1*

in Figure 3(a). As can be observed, SSE changes with the evolving nature of the data. Specifically, it increases slightly when noise is added at $t1$. There is almost no change in SSE when highly correlated nodes are inserted into an existing cluster at $t2$. When some data points are deleted at $t3$, it increases since the subspace is shrunk and some of the exemplars are removed. As $t4$ is a duplicate snapshot of $t3$, SSE remains unchanged. At $t5$, SSE first increases significantly due to the addition of a new cluster but comes back to the original range once the subspace is reconstructed based on the total data. Like the case of $t4$ before, SSE remains unchanged at $t6$. At $t7$, although a cluster is removed, SSE does not change because the subspace for the existing clusters is unchanged, and it is enough to estimate the approximation.

To further examine the subspace, we plotted the number of exemplars for each time step in Figure 3(b). We can see that when noise or new nodes are added, the subspace grows, such as in the case of $t1$, $t2$ and $t5$. On the other hand, when nodes are deleted like $t3$ and $t7$, the subspace shrinks. Hence, the size of subspace reflects the complexity of data structure.

To estimate the cluster number, we compute the gaps between consecutive top eigenvalues, i.e., eignvalue$^{(i)}$ − eignvalue$^{(i+1)}$, where $i = 1, 2, ...$, and detect the deleted clusters (the cluster on the current data with less than 30 data points based on the last cluster membership is determined to be deleted) at the time steps where a clustering is performed, as shown in Table II. We notice, the gaps are larger for the top eigenvalues, and become smaller with the eigenvalue decreasing. According to the change in SSE and the number of the deleted clusters, the data structure at the first five time steps (from $t0$ to $t4$) does not change, indicating the cluster number remains the same. From the gaps between eigenvalues, we notice the index of the last gap that is larger than a threshold (20 in our experiment) is almost the same, meaning the cluster number doesn't change at these time steps. At $t3$, we see the index of the last gap greater than 20 is 4, however, based on the change in SSE (almost no change) and the deleted cluster (no deleted cluster), we determine the cluster number should be 5, the same as that for the last time. At $t5$, we determine the cluster number is 6 according to the index of the last gap greater than 20, which is consistent with the prediction of SSE that a significant increase of SSE may be resulted by new inserted clusters. At $t7$, SSE takes almost no change, and one cluster is deleted, so the cluster number is 1 less than that at the last step, i.e., 5. From the computed eigenvalues, the index of the last gap greater than 20 is 5.

Next, we study the performance of ECKF for clustering. In the low-rank approximation results, SSE remain unchanged at $t4$ and $t6$ indicating that we do not need to

perform new clustering at these time steps. Thus, in the rest of our experiments, we only report the clustering results on the remaining data. To examine the effect of parameter α, in Figure 4 we report NMIs with α going from 1 to 0.2. A higher value of α indicates lower incorporation of historical knowledge into clustering. Since the results obtained by KMF and AccKM are independent to α, we plot them as flat lines. ECKF gains the highest NMIs in most cases, while AccKM gets the worst results. Though the results by both ECKF and ENC fluctuate with the change of α, ECKF gains the best results in all values of α at most time steps except for $t1$. This is because, ECKF integrates more historical knowledge such as the last approximation results and last clustering results, which is much more effective than ENC that only incorporates the historical data. In general, KMF underperforms compared to both ECKF and ENC, which indicates incorporation of historical information in clustering evolving data is essential. In Figure 5, we show the comparison of four methods with respect to NMI for $\alpha = 0.8$. It is clear that ECKF outperforms the other methods by having highest NMIs at evolving time steps.

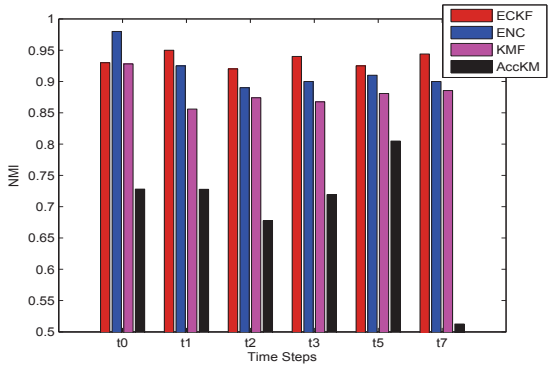


Fig. 5. Comparison of NMI among four methods with $\alpha = 0.8$ at each time step.

5.3 Real data

To evaluate the performance of ECKF on real world data, we selected two publicly available social network datasets. The first dataset is collected from Pubmed [1] database. To generate a set of evolving datasets, we used an author-paper structure to denote each time data. We extracted records published between $1990 - 2009$ on the topics of "drug abuse", and eliminated the ones without authors and abstracts. We also removed the authors who published less than 5 papers in this period. Then, we processed the records by extracting the abstracts, removing common words and stemming them by Porter's suffix-stripping algorithm [50]. After this, we had 11,250 authors, 51,684 papers and 63,017 words. Next, we constructed an author-paper matrix for each year by denoting each author with a word frequency vector. Finally, we had 20 groups of datasets, shown in Table III. The affinity matrix is computed by $S = A^T * A$. To make it more sparse, we made the entries less than 0.005 to be zeros. The second dataset we have used is the Enron Email corpus [49] [2], which contains email messages collected from 150 senior executives in the Enron corporation. We used a user-message structure to denote each time data with every user corresponding to an email address. A user's message in one time step is the concatenation of all the sent and received emails. First, we selected a part of data from the period between April 2001 to January 2002 and regarded each month as a time step. After the elimination of users who had sent or received less than 10 emails, we had $75,532$ users, and $68,588$ email messages. From all these email messages, common stop words were first removed followed by stemming using Porter's suffix-stripping algorithm, resulting in a vocabulary of $357,274$ words. Next, we constructed a user-message matrix for each month by denoting a user appearing in a month with a word frequency vector. The details of the 10 groups of datasets we obtained in the end are given in Table IV. For this set of evolving data, most of datasets contain over $5,000$ nodes. Hence, instead of computing S by $A^T * A$, due to both computational and space limitations, we adopted a different approach. We executed the low-rank approximation algorithm directly on the data matrix: $A \approx C_A U_A R_A$. Then, the approximation of the affinity matrix was constructed by,

$$\begin{aligned}\tilde{S} &= (C_A U_A R_A)^T (C_A U_A R_A) \\ &= R_A^T U_A^T C_A^T C_A U_A R_A \\ &= CUR\end{aligned}$$

where $C = R_A^T$, $U = U_A^T C_A^T C_A U_A$ and $R = R_A$. Since, ENC requires the affinity matrix and not its approximation as the input, we only evaluated ECKF, KMF, and AccKM on this dataset. On both real-world datasets, we executed ECKF, ENC, and AccKM with $\alpha = 0.8$.

In Figure 6(a), we plot SSEs gained by Colibri methods on the Pubmed datasets. The peaks before 1997 indicate the changes in the structure of the data. Especially during 1992 to 1996, every year a new subspace needs to

TABLE III
Pubmed Datasets .

Year	*Authors*	*Papers*	*Year*	*Authors*	*Papers*
1990	*2126*	*1886*	*2000*	*4059*	*2683*
1991	*2205*	*1813*	*2001*	*4097*	*2646*
1992	*2419*	*1880*	*2002*	*4371*	*2867*
1993	*2820*	*2101*	*2003*	*4637*	*3123*
1994	*2938*	*2181*	*2004*	*4896*	*3209*
1995	*3160*	*2219*	*2005*	*5122*	*3493*
1996	*3273*	*2260*	*2006*	*5146*	*3688*
1997	*3464*	*2376*	*2007*	*5248*	*4006*
1998	*3727*	*2466*	*2008*	*4899*	*3852*
1999	*3852*	*2542*	*2009*	*954*	*393*

1. http://www.ncbi.nlm.nih.gov/pubmed/

2. http://www.cs.cmu.edu/∼enron/

TABLE IV
Enron Email Datasets.

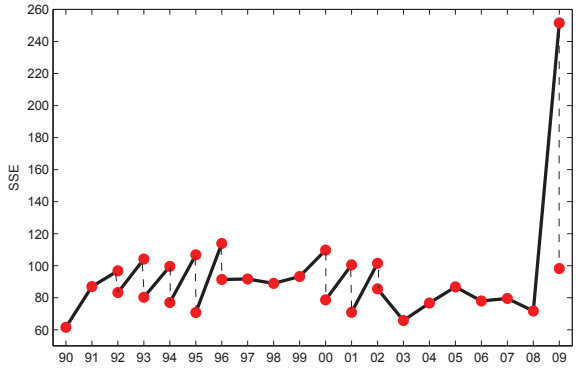
Time steps	*Number of Users*	*Number of Messages*
2001-04	*5942*	*14348*
2001-05	*7481*	*16986*
2001-06	*6072*	*11128*
2001-07	*3777*	*6980*
2001-08	*4815*	*7490*
2001-09	*6670*	*9888*
2001-10	*9795*	*29556*
2001-11	*9246*	*23441*
2001-12	*4557*	*9471*
2002-01	*2865*	*11948*

be created. In the period 1996 to 1999, the data structure remains relatively steady, and the subspace at each time step succeeds from the previous one. A similar pattern of peaks happens from 2000 to 2002, and stability thereafter appears until 2008. The sudden spike in the SSE in 2009 suggests that the subspace in this year is significantly different from that in the previous year.

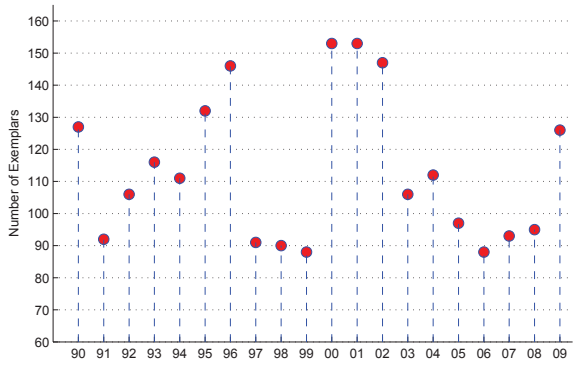
In Figure 6(b), we have plotted the number of exemplars to show the complexity of data structure at each time step. The pattern observed here supports the one observed in Figure 6(a). That is, before 1997, the subspace grows almost every year, indicating that the data structure is becoming more complex. Thereafter, during 1997 to 1999, the subspace shrinks. Again, from 2000 to 2002, the data becomes more complex, followed by a reduction in the complexity after 2003.

The lack of much change in SSE during 1997 to 1999 and in 2007, means that we do not need a new clustering on these datasets. In Figure 7, we have plotted the number of clusters across all time steps. As discussed before, since the data is more complex prior to 2002, the number of clusters is more compared with after 2002. Figure 8 performs a head-to-head comparison of ECKF, ENC, KMF, and AccKM on the remaining datasets with respect to KMCost values. Clearly, among the four methods, ECKF gains the lowest KMCost values on all the evolving data, followed by AccKM. The KMCost values of ENC are zeros during the period of 2003 to 2007, where the data contains over 4500 nodes. A zero-value here indicates ENC fails to cluster the data due to insufficient memory. All through out, KMF generally gets the highest KMCosts. This clearly demonstrates the benefits of incorporating historical knowledge into the matrix factorization based clustering. The exemplars are able to rule out noisy data points, hence making the ECKF framework very robust.

Next, we report the results on the Enron datasets. Figure 9(a) shows the approximation errors over the time steps of ten months. As we can see, the structure of the data changes considerably in most months, except for the periods of June to July and August to September in 2001. In particular, in December 2001, the data changes significantly indicated by the highest point in the figure. In Figure 9(b), we have shown the number of selected exemplars for each month. Note that, before



(a)



(b)

Fig. 6. Low-rank approximation on Pubmed datasets: (a) shows approximation error at each time step, and (b) shows the number of selected exemplars at each time step.

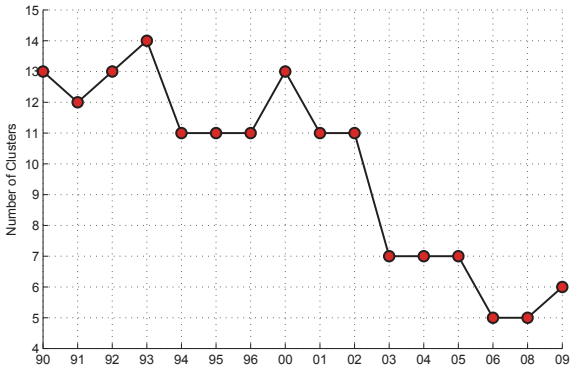


Fig. 7. Number of clusters on Pubmed Datasets.

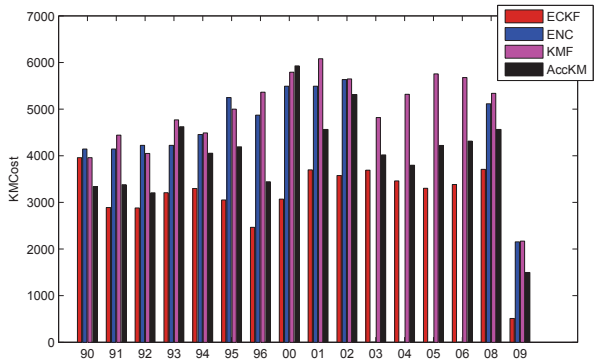
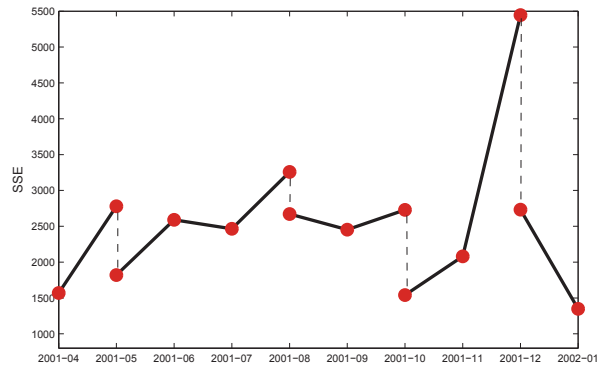
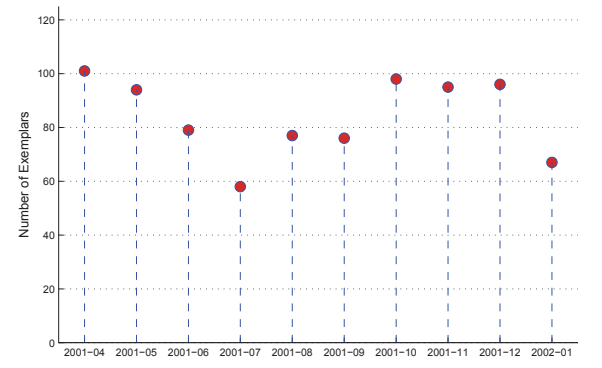


Fig. 8. Comparison of KMCost between ECKF, ENC, KMF, and AccKM on Pubmed datasets.

July 2001, the number decreases every month, meaning the structure is becoming less complex, while after that, it increases for the opposite reason. This is verified by Table IV, where before July 2001, the numbers of users and messages decrease every month, and increase after August 2001.



(a)



(b)

Fig. 9. Low-rank approximation on Enron datasets: (a)shows approximation error at each time step, and (b) shows the number of selected exemplars at each time step.

Based on the low-rank approximation results, we do not perform a new clustering on the data in July 2001. In Figure 10, we have plotted the number of clusters on the Enron datasets. As noticed, the value of cluster number drops before June 2001, and then grows, which is consistent with the curve of the number of exemplars in Figure 9(b). In Figure 11, we have compared the KMCost values obtained by ECKF, KMF, and AccKM. As we see, ECKF gains the lowest KMCost values on all the envolving Enron datasets.

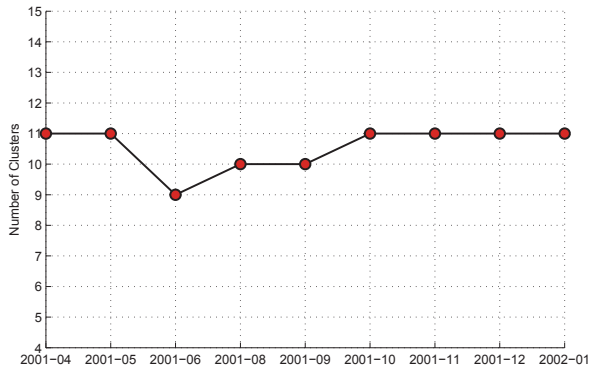


Fig. 10. Number of clusters on Enron Datasets.

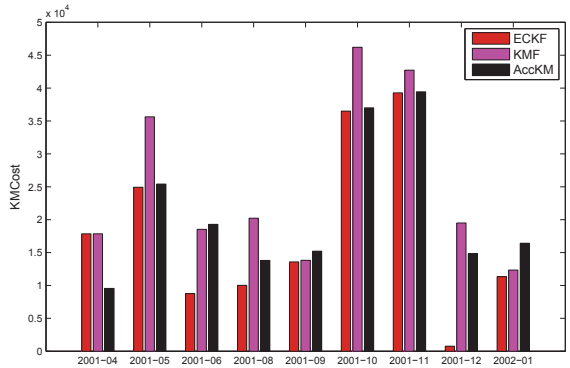


Fig. 11. Comparison of KMCost between ECKF, KMF, and AccKM on Enron datasets.

In Table V, we have compared the efficiency of the four algorithms in terms of execution time and storage. With regards to speed, ENC is the fastest with KMF, ECKF, and AccKM occupying second, third, and fourth place, respectively. Although ECKF requires more time than ENC and KMF, as seen from the experiments, it is able to process extremely large datasets, having almost $10,000$ nodes within 200 seconds. From the storage perspective, ECKF leads the way as it requires the least memory, followed by KMF, and the last being ENC and AccKM which require the most.

TABLE V
Comparisons of efficiency on two real-world datasets: the values are the average of results on all time step data.

Pubmed Datasets		
	Time(s)	*Space*
ECKF	*136.33*	*0.13*
ENC	*6.65*	*1*
KMF	*48.34*	*0.14*
AccKM	*72.615*	*1*
Enron Datasets		
	Time(s)	*Space*
ECKF	*96.44*	*0.32*
KMF	*53.16*	*0.33*
AccKM	*314.52*	*1*

6 Conclusions

We presented a mathematically-rigorous theoretical framework for **E**volutionary **C**lustering based on low-rank **K**ernel matrix **F**actorization (ECKF). By integrating low-rank kernel matrix approximation, ECKF can partition extremely large data sets at every time step. ECKF works directly in the low-rank subspace, which has highly attractive properties such as feature selection, robustness to noise, and as a result, higher clustering accuracy. Empirically, we demonstrated that ECKF outperforms existing evolutionary clustering methods in terms of the clustering accuracy and efficiency through extensive experiments performed on synthetic as well as publicly available real datasets.

Acknowledgements

This research was partially funded by U. S. National Science Foundation under grants IIS-0713315 and CNS-0751045, and by the 21st Century Jobs Fund Award, State of Michigan, under grant 06-1-P1-0193.

References

[1] A. K. Jain, M. N. Murty, and P. J. Flynn, "Data clustering: A review," *ACM Computing Surveys*, pp. 264–323, 1999.

[2] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*, Wiley-Interscience, New York, second edition, 2001.

[3] N. Bouguila and D. Ziou, "High-dimensional unsupervised selection and estimation of a finite generalized dirichlet mixture model based on minimum message length," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 10, pp. 1716–1731, Oct. 2007.

[4] Mohamed Bouguessa and Shengrui Wang, "Mining projected clusters in high-dimensional spaces," *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 4, pp. 507–522, 2009.

[5] D. Perera, J. Kay, I. Koprinska, K. Yacef, and O.R. Zaiane, "Clustering and sequential pattern mining of online collaborative learning data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 6, pp. 759–772, June 2009.

[6] S. Marinai, E. Marino, and G. Soda, "Font adaptive word indexing of modern printed documents," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 8, pp. 1187–1199, Aug. 2006.

[7] M. Bulacu and L. Schomaker, "Text-independent writer identification and verification using textural and allographic features," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 4, pp. 701–717, April 2007.

[8] Liping Jing, Michael K. Ng, and Joshua Zhexue Huang, "An entropy weighting k-means algorithm for subspace clustering of high-dimensional sparse data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 19, no. 8, pp. 1026–1041, 2007.

[9] K.M. Hammouda and M.S. Kamel, "Hierarchically distributed peer-to-peer document clustering and cluster summarization," *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 5, pp. 681–698, May 2009.

[10] N.K. Papadakis, D. Skoutas, K. Raftopoulos, and T.A. Varvarigou, "Stavies: a system for information extraction from unknown web data sources through automatic web wrapper generation using clustering techniques," *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 12, pp. 1638–1652, Dec. 2005.

[11] S. G. Petridou, V. A. Koutsonikola, A. I. Vakali, and G. I. Papadimitriou, "Time-aware web users' clustering," *IEEE Transactions on Knowledge and Data Engineering*, vol. 20, no. 5, pp. 653–667, May 2008.

[12] G. J. Bloy, "Blind camera fingerprinting and image clustering," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 3, pp. 532–534, March 2008.

[13] Manjeet Rege, Ming Dong, and Jing Hua, "Graph theoretical framework for simultaneously integrating visual and textual features for efficient web image clustering," in *Proceeding of the 17th international conference on World Wide Web*, New York, NY, USA, 2008, pp. 317–326, ACM.

[14] Jianbo Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 8, pp. 888–905, 2000.

[15] I.S. Dhillon, Yuqiang Guan, and B. Kulis, "Weighted graph cuts without eigenvectors a multilevel approach," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 11, pp. 1944–1957, Nov. 2007.

[16] M. Vignes and F. Forbes, "Gene clustering via integrated markov models combining individual and pairwise features," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 6, no. 2, pp. 260–270, April-June 2009.

[17] P. C. H. Ma and K. C. C. Chan, "A novel approach for discovering overlapping clusters in gene expression data," *IEEE Transactions on Biomedical Engineering*, vol. 56, no. 7, pp. 1803–1809, July 2009.

[18] Nan Du, Bin Wu, Xin Pei, Bai Wang, and Liutong Xu, "Community detection in large-scale social networks," in *Proceedings of the 9th WebKDD and 1st SNA-KDD 2007 workshop on Web mining and social network analysis*, 2007, pp. 16–25.

[19] Jianhua Ruan and Weixiong Zhang, "An efficient spectral algorithm for network community discovery and its applications on biological and social networks," in *IEEE International Conference on Data Mining*, Los Alamitos, CA, USA, 2007, vol. 0, pp. 643–648, IEEE Computer Society.

[20] Deepayan Chakrabarti, Ravi Kumar, and Andrew Tomkins, "Evolutionary clustering," in *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2006, pp. 554–560.

[21] Yun Chi, Xiaodan Song, Dengyong Zhou, Koji Hino, and Belle L. Tseng, "Evolutionary spectral clustering by incorporating temporal smoothness," in *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2007, pp. 153–162.

[22] H. Prehn and G. Sommer, "An adaptive classification algorithm using robust incremental clustering," in *Proceedings of the 18th International Conference on Pattern Recognition*, 2006, vol. 1, pp. 896–899.

[23] W. Pedrycz and Keun-Chang Kwak, "The development of incremental models," *IEEE Transactions on Fuzzy Systems*, vol. 15, no. 3, pp. 507–518, June 2007.

[24] Zhengdong Lu and M.A. Carreira-Perpinan, "Constrained spectral clustering through affinity propagation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* , June 2008, pp. 1–8.

[25] P.K. Mallapragada, Rong Jin, and A.K. Jain, "Active query selection for semi-supervised clustering," in *Proceedings of the 19th International Conference on Pattern Recognition*, Dec. 2008, pp. 1–4.

[26] C. Gupta and R. Grossman, "Genic: A single pass generalized incremental algorithm for clustering," in *proc. of SIAM Int. Conf. on Data Mining*, 2004.

[27] S. Guha, N. Mishra, R. Motwani, and L. OCallaghan, "Clustering data streams," in *proc. of IEEE Symposium on Foundations of Computer Science*, 2000.

[28] M. Charikar, C. Chekuri, T. Feder, and R. Motwani, "Incremental clustering and dynamic information retrieval," in *proc. of the 29th STOC Conference*, 1997.

[29] Y. Li, J. Han, , and J. Yang, "Clustering moving objects," in *proc. of the 10th ACM SIGKDD Conference*, 2004.
[30] T. Zhang, R. Ramakrishnan, and M. Livny, "Birch: An efficient data clustering method for very large databases," in *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 1996, pp. 103–114.
[31] H. Ning, W. Xu, Y. Chi, Y. Gong, and T. Huang, "Incremental spectral clustering with application to monitoring of evolving blog communities," in *proc. of SIAM Int. Conf. on Data Mining*, 2007.
[32] K. Wagstaff, C. Cardie, S. Rogers, and S. Schroedl, "Constrained k-means clustering with background knowledge," in *proc. of ICML Conference*, 2001.
[33] Yanhua Chen, Manjeet Rege, Ming Dong, and Jing Hua, "Incorporating user provided constraints into document clustering," in *proc. of IEEE ICDM*, 2007.
[34] X. Ji and W. Xu, "Document clustering with prior knowledge," in *proc. of ACM SIGIR*, 2006.
[35] Gene H. Golub and Charles F. Van Loan, *Matrix computations (3rd ed.)*, Johns Hopkins University Press, Baltimore, Maryland, USA, 1996.
[36] D. D. Lee and H. S. Seung, "Learning the parts of objects by non-negative matrix factorization.," *Nature*, vol. 401, no. 6755, pp. 788–791, October 1999.
[37] Daniel D. Lee and H. Sebastian Seung, "Algorithms for non-negative matrix factorization," in *The Neural Information Processing Systems*, 2000, pp. 556–562.
[38] Tao Li and Chris Ding, "The relationships among various nonnegative matrix factorization methods for clustering," in *Proceedings of the IEEE International Conference on Data Mining*, 2006, vol. 0, pp. 362–371.
[39] C. Ding, Xiaofeng He, and Horst D. Simon, "On the equivalence of nonnegative matrix factorization and spectral clustering," in *Proceedings of the SIAM International Conference of Data Mining*, 2005, pp. 606–610.
[40] Patrik O. Hoyer, "Non-negative matrix factorization with sparseness constraints," *The Journal of Machine Learning Research*, vol. 5, pp. 1457–1469, 2004.
[41] Chris Ding, Tao Li, Wei Peng, and Haesun Park, "Orthogonal nonnegative matrix t-factorizations for clustering," in *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2006, pp. 126–135.
[42] Chris Ding, Tao Li, and Michael I. Jordan, "Convex and semi-nonnegative matrix factorizations," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Los Alamitos, CA, USA, 2008, vol. 99, IEEE Computer Society.
[43] Dimitris Achlioptas and Frank Mcsherry, "Fast computation of low-rank matrix approximations," *Journal of the ACM*, vol. 54, no. 2, pp. 9, 2007.
[44] Michael W. Berry, Shakhina A. Pulatova, and G. W. Stewart, "Algorithm 844: Computing sparse reduced-rank approximations to sparse matrices," *ACM Transactions on Mathematical Software*, vol. 31, no. 2, pp. 252–269, 2005.
[45] Petros Drineas, Ravi Kannan, and Michael W. Mahoney, "Fast monte carlo algorithms for matrices iii: Computing a compressed approximate matrix decomposition," *SIAM Journal on Computing*, vol. 36, pp. 184–206, 2006.
[46] Jimeng Sun, Yinglian Xie, Hui Zhang, and Christos Faloutsos, "Less is more: Sparse graph mining with compact matrix decomposition," *Statistical Analysis and Data Mining*, vol. 1, no. 1, pp. 6–22, 2008.
[47] Hanghang Tong, Spiros Papadimitriou, Jimeng Sun, Philip S. Yu, and Christos Faloutsos, "Colibri: fast mining of large static and dynamic graphs," in *Proceeding of the 14th ACM international conference on Knowledge discovery and data mining*, 2008, pp. 686–694.
[48] Alexander Strehl, Joydeep Ghosh, and Claire Cardie, "Cluster ensembles - a knowledge reuse framework for combining multiple partitions," *Journal of Machine Learning Research*, vol. 3, pp. 583–617, 2002.
[49] Bryan Klimt and Yiming Yang, "The enron corpus: A new dataset for email classification research," in *Proceeding of the European Conference on Machine Learning*, 2004, pp. 217–226.
[50] M. F. Porter. "An algorithm for suffix stripping," *Program*, vol. 14, no. 3, pp. 130–137, 1980.

Lijun Wang received her MS from School of Computer Science and Technology, Harbin Institute of Technology, P. R. China in 2004. She is currently a PhD student in the Machine Vision and Pattern Recognition Laboratory of the Department of Computer Science, Wayne State University, Detroit, MI. Her research interests lie in the areas of Data Mining, Information Retrieval, Computer Vision, Graph theory, and Multimedia Analysis.

Manjeet Rege is currently an Assistant Professor of Computer Science at Rochester Institute of Technology. He received the PhD in Computer Science from Wayne State University, Detroit, MI. His research interests lie in the areas of Data Mining, Multimedia Analysis, and Information Retrieval. His research has been published in various international conferences and journals such as IEEE International Conference on Data Mining, ACM International Conference on Multimedia, Data Mining and Knowledge Discovery Journal etc. He also serves regularly on the program committees of various international conferences and workshops.

Ming Dong received his BS degree from Shanghai Jiao Tong University, Shanghai, P.R. China in 1995 and his PhD degree from the University of Cincinnati, Ohio, in 2001, both in electrical engineering. He is currently an associate professor of Computer Science and the Director of the Machine Vision and Pattern Recognition Laboratory. His research interests include pattern recognition, data mining, and multimedia analysis. His research is currently supported by the NSF and State of Michigan. He has published over 70 technical articles, many in premium journals and conferences such as IEEE Trans. on Pattern Analysis and Machine Intelligence, IEEE Trans. on Visualization and Computer Graphics, IEEE Trans. on Neural Networks, IEEE Trans. on Computers, IEEE Trans. on Fuzzy Systems, IEEE ICDM, IEEE CVPR, ACM Multimedia, and WWW. He is currently an associate editor of IEEE Trans. on Neural Networks, Pattern Analysis and Applications (Springer) and was on the editorial board of International Journal of Semantic Web and Information Systems, 2005-2006. He also serves as a program committee member for many related conferences. He is a member of the IEEE.

Yongsheng Ding received the Ph.D. degree in Electrical Engineering from Donghua University, Shanghai, China. He is currently a full professor at College of Information Sciences and Technology, Donghua University. He serves as a senior member of Institute of Electrical and Electronics Engineers (IEEE). He has published more than 300 technical papers, and four research monograph/advanced textbooks entitled DNA Computing and Soft Computing (2002), Computational Intelligence (2004), Natural Computing and Network Intelligence (2008), Bio-Network based Intelligent Control and Optimization (2010). His scientific interests include computational intelligence, pattern recognition, network intelligence, nature-inspired technologies, bio-computing and bio-informatics, intelligent decision-making, and digitized textile and garment technology.