Large-scale spectral clustering using diffusion coordinates on landmark-based bipartite graphs

Khiem Pham
Department of Computer Science
San José State University
San José, California, USA
khiem.pham@sjsu.edu

Guangliang Chen
Department of Mathematics & Statistics
San José State University
San José, California, USA
guangliang.chen@sjsu.edu

Abstract

Spectral clustering has received a lot of attention due to its ability to separate nonconvex, non-intersecting manifolds, but its high computational complexity has significantly limited its applicability. Motivated by the document-term co-clustering framework by Dhillon (2001), we propose a landmark-based scalable spectral clustering approach in which we first use the selected landmark set and the given data to form a bipartite graph and then run a diffusion process on it to obtain a family of diffusion coordinates for clustering. We show that our proposed algorithm can be implemented based on very efficient operations on the affinity matrix between the given data and selected landmarks, thus capable of handling large data. Finally, we demonstrate the excellent performance of our method by comparing with the state-of-the-art scalable algorithms on several benchmark data sets.

1 Introduction

Given a data set $X = \{x_1, \dots, x_n\} \subset \mathbb{R}^d$ and a similarity function $\delta(\cdot, \cdot)$ such as the Gaussian radial basis function (RBF), spectral clustering (von Luxburg, 2007) first constructs a pairwise similarity matrix

$$W = (w_{ij}) \in \mathbb{R}^{n \times n}, \quad w_{ij} = \delta(x_i, x_j) \qquad (1)$$

and then uses the top eigenvectors of W (after certain kind of normalization) to embed X into a low-dimensional space where k-means is employed to group the data into k clusters. Though mathematically quite simple, spectral clustering can easily adapt to nonconvex geometries and accurately separate various non-intersecting shapes. As a result, it has been successfully applied to many practical tasks, e.g., image segmentation (Shi and Malik, 2000) and document clustering (Dhillon, 2001), often significantly outperforming traditional methods (such as k-means). Furthermore, spectral clustering has a very rich theory (von Luxburg, 2007), with interesting connections to kernel k-means (Dhillon et al., 2004), random walk (Meila and Shi, 2001), graph cut (Shi and Malik, 2000) (and the underlying spectral graph theory (Chung, 1996)), and matrix perturbation analysis (Ng et al., 2001).

However, spectral clustering is known to suffer from a high computational cost associated with the $n \times n$ matrix W, especially when n is large. Consequently, there has been considerable effort to develop fast, approximate algorithms that can handle large data sets (Fowlkes et al., 2004; Yan et al., 2009; Sakai and Imiya, 2009; Wang et al., 2009; Chen and Cai, 2011; Wang et al., 2011; Tasdemir, 2012; Choromanska et al., 2013; Cai and Chen, 2015; Moazzen and Tasdemir, 2016; Chen, 2018). Interestingly, a considerable fraction of them use a landmark set to help reduce the computational complexity of spectral clustering. Specifically, they first find a small set of data representatives (called *landmarks*), $Y = \{y_1, \dots, y_m\} \subset \mathbb{R}^d$ (with $m \ll n$), from the given data in X and then form an affinity matrix between X and Y (see Fig. 1):

$$A = (a_{ij}) \in \mathbb{R}^{n \times m}, \quad a_{ij} = \delta(x_i, y_j). \qquad (2)$$

Afterwards, different scalable methods use the matrix A in different ways to cluster the given data. For example, the column-sampling spectral clustering (cSPEC) algorithm (Wang et al., 2009) regards A as a column-reduced version of W and correspondingly use the left singular vectors of A to approximate the eigenvectors of W. However, they seem to consider only unnormalized spectral clustering, and it is unclear how they extend their technique to normalized spectral clustering (Shi and Malik, 2000; Ng et al., 2001). Another example is the landmark-based spectral

Proceedings of the Twelfth Workshop on Graph-Based Methods for Natural Language Processing (TextGraphs-12), pages 28–37
New Orleans, Louisiana, June 6, 2018.

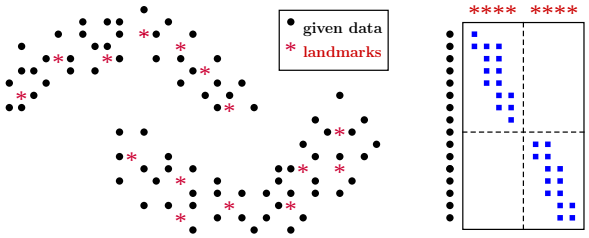

Figure 1: Illustration of landmark-based spectral clustering. Left: given data (in black color) and selected landmarks (in red); right: the affinity matrix A between the two sets of points with the blue squares indicating the largest entries in each row of A. Here, we assume that both the given data and the landmarks have been sorted according to the true clusters, so as to reveal the approximately block diagonal structure of A).

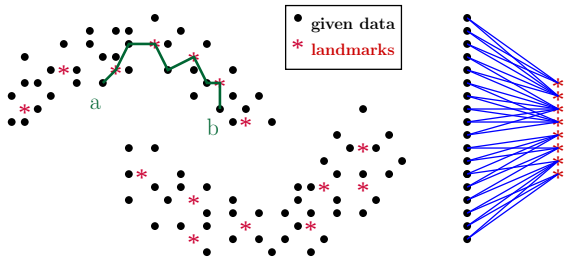

Figure 2: A bipartite graph whose two components are the given data (in black) and a landmark set learned from it (in red). Here, we form edges between each given data point and its two closest landmark points. Initially, no landmark point is connected to the two points a and b. However, if one simulates a random walk on the bipartite graph, then through a sequence of steps between the two components, a, b will be identified as belonging to the same cluster.

clustering (LSC) algorithm (Cai and Chen, 2015) which uses a row-sparsified version of the matrix A as approximate sparse representations of the input data while bypassing the expensive dictionary learning and sparse coding tasks. It then applies the L_1 normalization to each row of A, followed by a square-root L_1 column normalization. This method empirically works quite well but clearly there is a gap between its sparse coding motivation and the actual implementation. A third example is the k-means-based approximate spectral clustering (KASP) algorithm (Yan et al., 2009) which first applies the k-means algorithm to partition the given data into m small clusters and then performs spectral clustering to divide their centroids (which are the landmark points) into k groups. Next, they extend the clustering of the landmarks to the original data by performing 1 nearest neighbor (1NN) classification. This algorithm runs very fast, but is sensitive to the k-means clusters as it aggressively reduces the given data to a small set of centroids.

In this work we propose a novel landmark-based scalable spectral clustering approach by adapting the co-clustering framework by Dhillon (Dhillon, 2001) for landmark-based clustering and combining it with diffusion maps (Coifman and Lafon, 2006). Specifically, with the given data $X = \{x_i\}$ and a selected landmark set $Y = \{y_j\}$, we first construct a bipartite graph G_2 with X and Y being the two parts, and form edges between each x_i and its s nearest neighbors y_j in the landmark set with weights $a_{ij} = \delta(x_i, y_j)$. We then compute the transition probabilities for all the vertices of G_2 and use them to define a random walk on the bipartite graph, which (when being iterated forward) further generates a diffusion process on G_2. We expect the resulting diffusion coordinates to be able to capture the global geometry of the clusters at different scales and, as a result, the connectivity of each cluster will be significantly strengthened (see Fig. 2). We will show that the diffusion coordinates may be computed directly from the $n \times m$ matrix $A = (a_{ij})$. Lastly, we propose three different ways to use the diffusion coordinates for clustering the data in X (depending on the length of the random walk).

The rest of the paper is organized as follows. First, in Section 2, we review some necessary background. We then present our methodology in Section 3. Experiments are conducted in Section 4 to test our proposed algorithms. Finally, we conclude the paper in Section 5.

2 Background

In this section, we first review the Normalized Cut (Ncut) algorithm (Shi and Malik, 2000) and its connections to random walk (Meila and Shi, 2001) and diffusion maps (Coifman and Lafon, 2006). Next, we will review the co-clustering framework in the setting of documents data by Dhillon (2001).

2.1 The Ncut algorithm

Given a data set $X = \{x_1, x_2, ..., x_n\} \subset \mathbb{R}^d$ and a notion of similarity δ, we may construct a weighted graph G by using the data points in X as vertices and assigning an edge between any two points x_i, x_j with associated weight $w_{ij} = \delta(x_i, x_j)$. Let $W = (w_{ij}) \in \mathbb{R}^{n \times n}$, which is the weight matrix of G. The degree of x_i is the total

edge weight at that vertex: $d_i = \sum_j w_{ij}$, which measures the connectivity of x_i. The diagonal matrix $D = \text{diag}(d_1, \ldots, d_n)$ is called the degree matrix. The Laplacian of the graph G is defined as $L = D - W$, which is a positive semidefinite matrix. One way to normalize L is the following

$$L_{rw} = D^{-1}L = I - D^{-1}W. \tag{3}$$

For any subset of vertices $S \subset X$, we define the cut between S and its complement $\bar{S}$ as

$$\text{cut}(S, \bar{S}) = \sum_{x_i \in S} \sum_{x_j \in \bar{S}} w_{ij}. \tag{4}$$

The volume of S is defined as the total connectivity of the vertices in S:

$$\text{vol}(S) = \sum_{x_i \in S} d_i. \tag{5}$$

The Ncut algorithm (Shi and Malik, 2000) finds k clusters from the given data X by minimizing the following objective function:

$$\text{Ncut}(S_1, \ldots, S_k) = \sum_{\ell=1}^{k} \frac{cut(S_\ell, \bar{S}_\ell)}{vol(S_\ell)} \tag{6}$$

over all possible partitions $X = S_1 \cup \cdots \cup S_k$. Such a formulation of clustering is very convenient, however, the resulting optimization problem is intractable due to its combinatorial nature. Fortunately, by using a continuous relaxation, the above problem is reduced to finding the bottom $k-1$ eigenvectors of the normalized graph Laplacian L_{rw} (corresponding to its smallest positive eigenvalues)[1]:

$$V = [v_1 \mid \cdots \mid v_{k-1}] \in \mathbb{R}^{n \times (k-1)}. \tag{7}$$

One then regards the rows of V as a low-dimensional embedding of the original data and clusters them by using k-means.

2.2 Random walk and diffusion maps

Let $P = D^{-1}W$, whose row sums are all one. We can then write $L_{rw} = I - P$. Algebraically, the bottom eigenvectors of L_{rw} are just the top eigenvectors of P. However, since P is row-stochastic, it can be used as a transition probability matrix to define a random walk on the graph G (Meila and Shi, 2001). Under such a model, clustering can be interpreted as a way of finding a partition of the graph such that the random walk stays long inside the clusters and rarely moves between them.

If the random walk is moved forward for many iterations, then a diffusion process is generated on the graph G. For every integer $\alpha \geq 1$, P^α is the α-step transition matrix. Different values of α integrate the local connectivity information of the graph at different scales, with larger α yielding more global descriptions. The rows of P^α define a family of discrete distributions, one at each vertex of G, and are called the α-step diffusion coordinates (Coifman and Lafon, 2006). For practical purposes, it suffices to use low-dimensional approximations of them (by exploiting the fast decay of the α-th power of the spectrum of P):

$$V^{(\alpha)} = [\lambda_1^\alpha v_1 \mid \cdots \mid \lambda_p^\alpha v_p] \in \mathbb{R}^{n \times p}, \tag{8}$$

where $p \in \mathbb{Z}^+$ and λ_i, v_i are the largest eigenvalues and eigenvectors of P (excluding the eigenvalue 1 and associated eigenvector). In this work, we use the rows of $V^{(\alpha)}$ as an embedding of the data for clustering purposes.

2.3 Dhillon's co-clustering framework

Dhillon (2001) proposed a spectral clustering based co-clustering framework in the setting of documents clustering. Specifically, given a document-term frequency matrix $A \in \mathbb{R}^{n \times m}$, they first construct a bipartite graph G_2 whose two parts are the documents and terms, respectively, and then use A to define a weight matrix for G_2 as follows:

$$W = \begin{pmatrix} & A \\ A^t & \end{pmatrix} \in \mathbb{R}^{(n+m)\times(n+m)}. \tag{9}$$

The two empty blocks of W are zero matrices of appropriate sizes (which have been omitted for simplicity), indicating no connection among documents or terms. Afterwards, they apply the Ncut algorithm (along with the above weight matrix W) to co-cluster documents and terms, and they derived an efficient way to implement Ncut solely based on operations on the $n \times m$ matrix A (thus effectively avoiding the larger matrix W in all calculations). We review their derivation below.

We start with some definitions. First, let D_1, D_2 be two diagonal matrices consisting (resp.) of the row and column sums of A:

$$D_1 = \text{diag}(A\,1), \quad D_2 = \text{diag}(A^t\,1). \tag{10}$$

[1] It can be easily shown that, for any weighted graph, the smallest eigenvalue of L_{rw} is $\lambda_0 = 0$, with associated eigenvector $v_0 = (1, \ldots, 1)^t$. This eigenpair is skipped by the Ncut algorithm.

Here, and in what follows, we abuse notation to use 1 to denote the column vector (with appropriate dimension) with all entries equal to one. Next, we define three different normalized version of A:

$$\widetilde{A}_1 = D_1^{-1} A, \quad \widetilde{A}_2 = A D_2^{-1}, \tag{11}$$

$$\widetilde{A} = D_1^{-1/2} A D_2^{-1/2}. \tag{12}$$

It is easy to see that $\widetilde{A}_1, \widetilde{A}_2$ are respectively row- and column-stochastic, while $\widetilde{A}$ is closely related to both of them in the following ways:

$$\widetilde{A}_1 = D_1^{-1/2} \widetilde{A} D_2^{1/2}, \tag{13}$$

$$\widetilde{A}_2 = D_1^{1/2} \widetilde{A} D_2^{-1/2}. \tag{14}$$

The degree matrix of the bipartite graph is

$$D = \mathrm{diag}(W\,1) = \begin{pmatrix} D_1 & \\ & D_2 \end{pmatrix}, \tag{15}$$

from which we may obtain the following transition probability matrix for the bipartite graph:

$$P = D^{-1}W = \begin{pmatrix} & \widetilde{A}_1 \\ \widetilde{A}_2^t & \end{pmatrix}. \tag{16}$$

The following result, first proved by Dhillon (2001), indicates the close connection between the eigenvalue decomposition of P and the SVD of $\widetilde{A}$.

Lemma 1. *Let $v_1 \in \mathbb{R}^n$, $v_2 \in \mathbb{R}^m$ and $v = (v_1; v_2) \in \mathbb{R}^{n+m}$. Then v is an eigenvector of P if and only if $\widetilde{v}_1 = D_1^{1/2} v_1$ and $\widetilde{v}_2 = D_2^{1/2} v_2$ are a pair of left/right singular vectors of $\widetilde{A}$.*

Proof. Suppose v is an eigenvector of P corresponding to some eigenvalue λ, that is, $Pv = \lambda v$, or equivalently,

$$\begin{bmatrix} & \widetilde{A}_1 \\ \widetilde{A}_2^t & \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = \lambda \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}, \tag{17}$$

From this we obtain

$$\widetilde{A}_1 v_2 = \lambda v_1, \quad \widetilde{A}_2^t v_1 = \lambda v_2 \tag{18}$$

and also (after plugging in (13) and (14))

$$\widetilde{A} D_2^{1/2} v_2 = \lambda D_1^{1/2} v_1, \tag{19}$$

$$\widetilde{A}^t D_1^{1/2} v_1 = \lambda D_2^{1/2} v_2. \tag{20}$$

This shows that $D_1^{1/2} v_1$ and $D_2^{1/2} v_2$ are the left and right singular vectors of $\widetilde{A}$ (corresponding to the same singular value λ). It is easy to verify that the converse is also true. □

Therefore, to obtain an eigenvector of P, we just need to first perform the SVD of $\widetilde{A}$ to find a pair of its left and right singular vectors

$$\widetilde{v}_1 = D_1^{1/2} v_1, \quad \widetilde{v}_2 = D_2^{1/2} v_2, \tag{21}$$

and then apply the following formula

$$v = \begin{pmatrix} v_1 \\ v_2 \end{pmatrix} = \begin{pmatrix} D_1^{-1/2} \widetilde{v}_1 \\ D_2^{-1/2} \widetilde{v}_2 \end{pmatrix} = D^{-1/2} \widetilde{v}, \tag{22}$$

where $\widetilde{v} = (\widetilde{v}_1; \widetilde{v}_2) \in \mathbb{R}^{n+m}$.

Lastly, to complete the co-clustering task, one just stacks the top $k-1$ eigenvectors of P as columns to form an embedding matrix $\mathbb{V} \in \mathbb{R}^{(n+m)\times(k-1)}$ and applies k-means to its rows to group the documents and terms simultaneously.

3 Methodology

In the previous section we reviewed the co-clustering framework by Dhillon (2001) which employs the Ncut algorithm (Shi and Malik, 2000) to partition a bipartite graph consisting of documents and terms by directly working on the document-term matrix. In this section, we extend their work in two ways. First, we adapt their bipartite graph model for landmark-based clustering by using instead the given data and a selected landmark set as its two parts. Second, we simulate a diffusion process on the bipartite graph to gather global information about the graph, however, our focus is still on clustering the given data for which we will introduce several ways of using such a bipartite graph model.

3.1 Derivation of diffusion coordinates on a bipartite graph

Given a data set, $X = \{x_1, \ldots, x_n\} \subset \mathbb{R}^d$, to be partitioned into k clusters, we select from X a set of landmark points, $Y = \{y_1, \ldots, y_m\} \subset \mathbb{R}^d$ (with $m \ll n$), by some sampling method, such as uniform sampling or k-means clustering. Let $A \in \mathbb{R}^{n\times m}$ be the affinity matrix between X and Y, computed by using a pre-specified similarity function δ as in (2). Like LSC (Cai and Chen, 2015), we preserve only the largest s ($s \ll m$) entries in each row of A, but our motivation is to focus on the most similar landmark points for each given data point. We then construct a bipartite graph $G_{X,Y}$ with X, Y as its two parts and a weight matrix W of the form in (9) but based on the affinity matrix A defined above. Again, the

two empty blocks of W indicate no connection inside each component of $G_{X,Y}$. Also, the s-sparse rows of A imply that each point in X is only connected to its s nearest neighbors in Y on the bipartite graph. See Fig. 2 for an illustration of our bipartite graph model.

Using the computational framework laid out in Subsec. 2.3, we may easily obtain various quantities like $\widetilde{A}_1, \widetilde{A}_2, \widetilde{A}, D, P$. In particular, we may use the transition matrix P to define a random walk on the bipartite graph and run it forward continuously to generate a diffusion process. The α-step transition matrix, P^α has the following form.

Lemma 2. *Let $\alpha \geq 1$ be any integer.*
(1) If $\alpha = 2q$ is even, then

$$P^\alpha = \begin{pmatrix} \left(\widetilde{A}_1 \widetilde{A}_2^t\right)^q & \\ & \left(\widetilde{A}_2^t \widetilde{A}_1\right)^q \end{pmatrix} \qquad (23)$$

(2) If $\alpha = 2q+1$ is odd, then

$$P^\alpha = \begin{pmatrix} & \left(\widetilde{A}_1 \widetilde{A}_2^t\right)^q \widetilde{A}_1 \\ \left(\widetilde{A}_2^t \widetilde{A}_1\right)^q \widetilde{A}_2^t & \end{pmatrix} \qquad (24)$$

This result indicates that after an even number of steps, the random walk (no matter in which component of the bipartite graph it is initiated) is always back to the original component, so that the original bipartite graph becomes two disconnected subgraphs, while after an odd number of steps, the random walk always ends in the other component, and hence the graph remains bipartite. See Fig. 3 for an illustration. Also, when $\alpha = 2q$, the α-step random walk on the bipartite graph $G_{X,Y}$ is equivalent to a q-step random walk within each component of $G_{X,Y}$ with the transition matrix $\widetilde{A}_1 \widetilde{A}_2^t$ or $\widetilde{A}_2^t \widetilde{A}_1$. This implies that for even integers α, one should focus on the two components X, Y of the bipartite graph separately and in principle may use the corresponding blocks of P^α as new transition matrices for clustering the two sets of data individually. In contrast, for odd integers α, one still needs to consider the two components X, Y together as a bipartite graph and simultaneously cluster the original data and the landmark set.

The actual algorithm we will propose uses a family of diffusion coordinates (corresponding to different time steps of the diffusion process) to embed the input data and/or the landmark points into low-dimensional spaces for clustering by k-means. The next result shows that one may obtain such diffusion coordinates directly from the matrix $\widetilde{A}$ (defined in (12)).

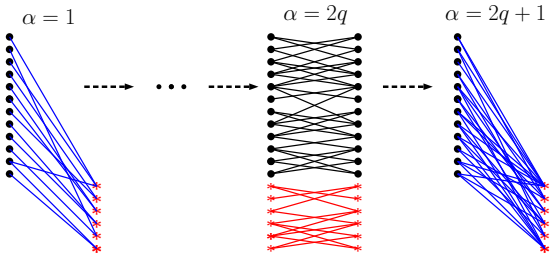

Figure 3: Simulation of a diffusion process on a bipartite graph. Left: initial bipartite graph; middle: the resulting graph after an even number of random walk iterations; right: the resulting graph after an odd number of iterations.

Theorem 1. *Let $\alpha, p \geq 1$ be two integers. The p-dimensional diffusion coordinates for $G_{X,Y}$ at time step α are*

$$\mathbb{V}^{(\alpha)} = D^{-1/2} \widetilde{V} \Lambda^\alpha \in \mathbb{R}^{(n+m)\times p}, \qquad (25)$$

where $\Lambda = diag(\lambda_1, \ldots, \lambda_p) \in \mathbb{R}^{p\times p}$ contains the largest p singular values of $\widetilde{A}$ (excluding the singular value 1), and $\widetilde{V} \in \mathbb{R}^{(n+m)\times p}$ the corresponding pairs of left and right singular vectors of $\widetilde{A}$ (one pair in each column).

Proof. This is a direct consequence of Lemma 1 combined with the formula in (8). □

Remark. We fix $p = k - 1$ (where k is the number of clusters) in the rest of the paper (so as to be consistent with the Ncut algorithm (Shi and Malik, 2000)), but other values of p could be used too (e.g. those determined based on the actual power decay of the eigenvalues of P).

Remark. If we extend α to zero in (25), then we can obtain the embedding used by Dhillon (2001). This shows that our work extends (Dhillon, 2001).

Remark. When α is even, the top $n \times p$ and bottom $m \times p$ blocks of $\mathbb{V}^{(\alpha)}$, denoted as $\mathbb{V}_X^{(\alpha)}, \mathbb{V}_Y^{(\alpha)}$, may be used separately as diffusion coordinates for the two sets of data X, Y.

3.2 Proposed algorithm

We present several different ways to use the α-step diffusion coordinates $\mathbb{V}^{(\alpha)} \in \mathbb{R}^{(n+m)\times(k-1)}$ in (25), computed from a landmark-based bipartite graph $G_{X,Y}$, for clustering the input data X.

We consider the following two cases:
(1) α even: In this case, the initial bipartite graph becomes two disjoint subgraphs corresponding to

Algorithm 1 Landmark-based Bipartite Diffusion Maps (LBDM)

Input: Data set $X = \{x_1, x_2, ..., x_n\} \subset \mathbb{R}^d$, # clusters k, similarity function δ, landmark selection method, # landmarks m, # nearest landmarks s, # diffusion steps α, clustering method: direct or landmark (for even α), or co-clustering (for odd α).

Output: A partition of X into k clusters.

1: Find m landmarks $Y = \{y_1, \ldots, y_m\} \subset \mathbb{R}^d$ using the given method.
2: Form the affinity matrix A between the input data X and their respective s nearest landmark points in Y by using the given similarity function δ.
3: Calculate the row and column sums of A and use them to normalize A to obtain $\widetilde{A}$ (as in (12)).
4: Find the largest $k-1$ singular values (excluding 1) and corresponding left and right singular vectors of $\widetilde{A}$.
5: Compute the diffusion coordinates matrix $\mathbb{V}^{(\alpha)}$ by (25) (with $p = k-1$).
6: Use the indicated clustering method to divide the input data set X into k clusters.

X and Y, respectively. A direct way of clustering the input data X is to focus on $\mathbb{V}_X^{(\alpha)}$, the α-step diffusion coordinates for X, and apply k-means to group them into k clusters. Alternatively, we can focus on $\mathbb{V}_Y^{(\alpha)}$, the α-step diffusion coordinates for the landmark set Y, and use k-means to group the landmark points into k subsets. Afterwards, we extend the landmark clustering to the input data through s nearest neighbors (sNN) classification, where s is the number of closest landmark points connected to each data point.

(2) α odd: In this case, we are still left with a bipartite graph. To cluster the input data X, we propose to run k-means with the full set of diffusion coordinates $\mathbb{V}^{(\alpha)}$ to divide $X \cup Y$ into k clusters, and later remove the landmark points from them.

We refer to the three above-mentioned methods for clustering the given data respectively as *direct clustering*, *landmark clustering*, and *co-clustering*.

We now present our scalable spectral clustering algorithm in Alg. 1.

Remark. We mention the work of Liu et al. (2013) in the setting of large graph data, which constructs a bipartite graph between the original graph nodes and "supernodes" generated through graph coarsening and then uses the plain co-clustering algorithm by Dhillon (2001) to partition the graph. Though the idea is somewhat similar, our method directly operates in the Euclidean data domain and extracts diffusion coordinates from the bipartite graph for clustering.

3.3 Run time analysis

The landmark selection step of Alg. 1 takes $O(ndm)$ time when k-means sampling is used, or $O(m)$ time when uniform sampling is used. The matrix A can be constructed in $O(nm(d+s))$ time since it takes $O(nmd)$ time to calculate all the pairwise distances between X and Y, and $O(nsm)$ time to find the s nearest landmarks in Y for each of the n data points in X. It then takes $O(ns)$ time to obtain $\widetilde{A}$, and $O(nsk)$ time to perform rank-k SVD of $\widetilde{A}$. The diffusion coordinates $\mathbb{V}^{(\alpha)}$ can be computed in $O((n+m)k)$ time. The final clustering step may take $O(nk^2)$, or $O(mk^2+ns)$, or $O((n+m)k^2)$ time, depending on the clustering method. Putting everything together, the total running time is $O(nm(d+s) + nk(s+k))$.

4 Experiments

In this section, we conduct extensive experiments to evaluate the practical performance of LBDM with $\alpha = 1, 2$. For the odd value $\alpha = 1$, for which the co-clustering method has to be used, we denote the corresponding implementation by LBDM$^{(1)}$. For the even value $\alpha = 2$, we use both the direct clustering and landmark clustering methods and denote them as LBDM$^{(2,X)}$ and LBDM$^{(2,Y)}$, respectively.

4.1 Experimental setup

We compare the different LBDM versions with the following algorithms: KASP (Yan et al., 2009), LSC (Cai and Chen, 2015), cSPEC (Wang et al., 2009), and the co-clustering algorithm by Dhillon (2001) (used similarly as LBDM$^{(1)}$), in the setting of Gaussian similarity. We also include the plain Ncut algorithm (Shi and Malik, 2000) in our study (as a baseline). We implement all the methods in MATLAB 2016b (except LSC[2]) and conduct all

[2]MATLAB code available at http://www.cad.zju.edu.cn/home/dengcai/Data/Clustering.html

the experiments on a compute server with 48GB of RAM and 2 CPUs with 12 total cores.

We choose six benchmark data sets - *usps*, *pendigits*, *letter*, *protein*, *shuttle*, *mnist* - from the LIBSVM website[3] (see Table 1 for their summary statistics). They are originally partitioned into training and test parts for classification purposes, but for each data set we have merged the two parts together for our unsupervised setting. Also, we provide the true number of clusters k to all algorithms to focus on the clustering task.[4]

Table 1: Data sets used in our experiments.

Data	n	d	k
usps	9,298	256	10
pendigits	10,992	16	10
letter	20,000	16	26
protein	24,387	357	3
shuttle	58,000	9	7
mnist	70,000	784	10

In order to have a fair comparison between the different algorithms, we use the same values for the shared parameters. In particular, we fix $m = 500$ (for all methods) and $s = 5$ (for LSC, Dhillon and LBDM with $\alpha = 1, 2$). Also, we feed all the algorithms with the same landmark set found by k-means (with only 10 iterations), which is initialized with the centroids obtained by preliminary k-means clustering on 10% of the data (with 100 iterations, 10 restarts). In the last step of each algorithm (where k-means is applied to cluster data in the respective embedding space), we use 100 iterations and 10 restarts.

We evaluate the different methods in terms of clustering accuracy and CPU time (averaged over 50 replications), with the former being calculated by first finding the best match between the output cluster labels and the ground truth and then computing the fraction of correctly assigned labels.

4.2 Results

We report the experimental results in Tables 2 (accuracy) and 3 (time).

The following observations on the clustering accuracy are at hand: (1) LBDM$^{(2,Y)}$ achieved the highest accuracy on three data sets (*usps*, *protein*, *mnist*), while LBDM$^{(2,X)}$ achieved the highest accuracy only on *letter*; (2) LSC and cSPEC each obtained the best accuracy once but each of them also performed very badly in at least one case; (3) The two co-clustering methods (Dhillon, LBDM with $\alpha = 1$) were very close and all performed reasonably well but never achieved the highest accuracy; (4) KASP never achieved the best accuracy either and performed very poorly in three cases (*pendigits*, *letter*, *mnist*). Overall, the LBDM family exhibited very stable performance (which demonstrates the power of diffusion coordinates), and they also outperformed the plain Ncut algorithm most of the time.

Regarding running time, LBDM$^{(2,Y)}$ and KASP are the two fastest methods because they both cluster the landmark points first and then extend the clustering to the input data through nearest neighbor classification. KASP is even faster because it applies spectral clustering directly to the landmark points in $\mathbb{R}^d$ (the corresponding weight matrix is only $m \times m$), but it is at the expense of accuracy. The cSPEC algorithm, on the other hand, is the slowest among the scalable methods (because it does not sparsify the matrix A), but it is still much faster than plain Ncut.

4.3 Parameter sensitivity study

We study in this section the effects of the parameters of LBDM (and relevant methods): m (number of landmark points), s (number of nearest landmark points), and α (diffusion time), using four data sets from Table 1: *usps*, *letter*, *protein*, and *mnist*.

In the first experiment, we focus on the parameter m by fixing $s = 5$ and varying m from 100 to 1000 with a step size of 100 in order to study its influence on all the scalable methods in Table 2. For each data set and each value of m, we apply k-means to sample m landmark points from the data set and provide the same landmark set to all the methods being compared to obtain their clustering accuracy and run time. This is then repeated 30 times and we report the average accuracy and time for each method in Fig. 4. In general, all methods except KASP and cSPEC improve their accuracy rates as more landmark points are used, with LBDM$^{(2,Y)}$ achieving the highest accuracy most of the time for three data sets (*usps*, *protein*, *mnist*). The CPU time of each algorithm seems to

[3] https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/

[4] When k is unknown, one may use methods like the gap statistic (Tibshirani et al., 2001) or eigenvectors rotation (Zelnik-Manor and Perona, 2004) to infer its value.

Table 2: Average clustering accuracy (%) of the various methods obtained on the data sets in Table 1. (Due to memory issue, we could not run the plain Ncut algorithm on the last two data sets.)

Dataset	Ncut	KASP	LSC	cSPEC	Dhillon	LBDM$^{(1)}$	_$^{(2,X)}$	_$^{(2,Y)}$
usps	66.21	67.25	66.86	66.89	68.21	67.80	68.10	69.45
pendigits	69.73	68.45	77.93	67.93	73.20	72.95	74.70	73.22
letter	24.93	26.19	31.51	24.98	32.06	32.13	32.21	31.28
protein	43.68	43.85	43.85	44.84	43.35	43.55	43.16	45.88
shuttle		74.52	39.71	82.78	74.24	74.26	74.38	74.49
mnist		57.99	70.28	54.50	72.15	72.43	72.37	73.29

Table 3: Average CPU time (in seconds) used by the various methods on the data sets in Table 1. The CPU time needed by the initial k-means to sample landmark points from each data set has been separately reported in the third column of the table (as it is common to all the methods).

Dataset	Ncut	(k-means)	KASP	LSC	cSPEC	Dhillon	LBDM$^{(1)}$	_$^{(2,X)}$	_$^{(2,Y)}$
usps	131.78	7.46 +	0.61	4.44	7.89	4.45	4.39	4.17	1.95
pendigits	246.08	3.13 +	0.55	3.08	5.26	3.14	2.91	3.08	1.65
letter	1180.70	5.30 +	0.77	12.24	25.07	13.51	14.96	12.87	2.78
protein	2024.54	27.04 +	0.41	3.55	7.54	3.93	4.04	3.93	4.40
shuttle		23.89 +	1.23	8.49	61.68	12.35	15.09	12.15	5.88
mnist		299.74 +	0.63	25.07	39.26	27.17	25.69	25.83	16.67

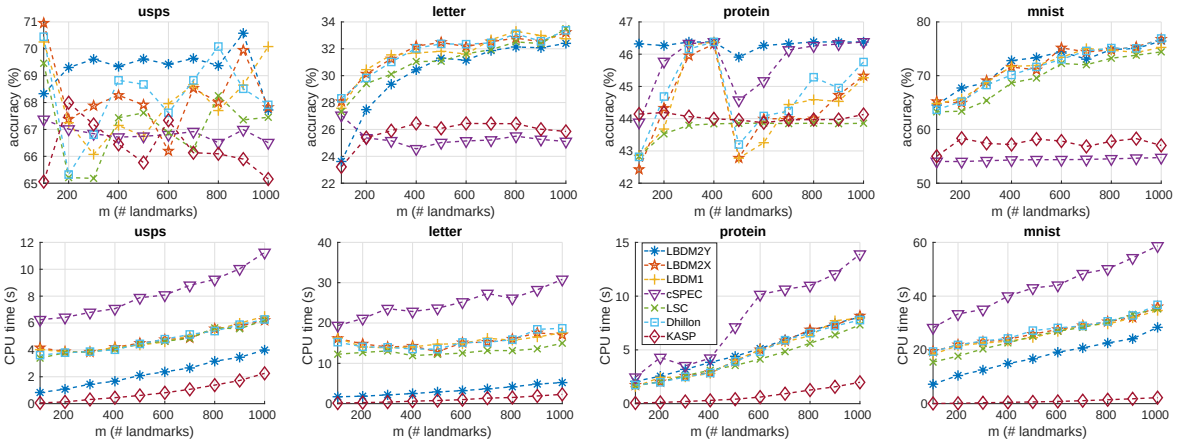

Figure 4: Effects of the parameter m (with $s = 5$ fixed all the time). Top row: clustering accuracy; bottom row: CPU time. In all plots the color and symbol of each method is fixed, so only one legend box is displayed.

depend linearly on m, with KASP always being the fastest two method.

In our second experiment about the parameter s (which is only needed by LBDM, LSC and Dhillon), we use the same setup as in the first experiment, except to fix $m = 500$ while varying s from 2 to 10 continuously. We plot the average clustering accuracy and run time of the different methods against the parameter s in Fig. 5. We see that increasing the value of s tends to decrease the clustering accuracy of each algorithm (with LBDM$^{(2,Y)}$ being the best in three cases), while increasing their run time linearly (but very little for LBDM$^{(2,Y)}$).

Lastly, we study the α parameter of LBDM by varying it from 1 to 40 continuously (with $m = 500$ and $s = 5$ fixed). Recall that for odd values of α, we have to use the co-clustering method LBDM$^{(\alpha)}$, while for each even value of α, we can use either the direct clustering method LBDM$^{(\alpha,X)}$ or the landmark clustering method LBDM$^{(\alpha,Y)}$. Their average accuracy (over 30 replications) for each value of α is displayed in Fig. 6. We can see that increasing the time scale α may further improve the clustering accuracy for all three methods on some data sets, demonstrating the power of diffusion maps.

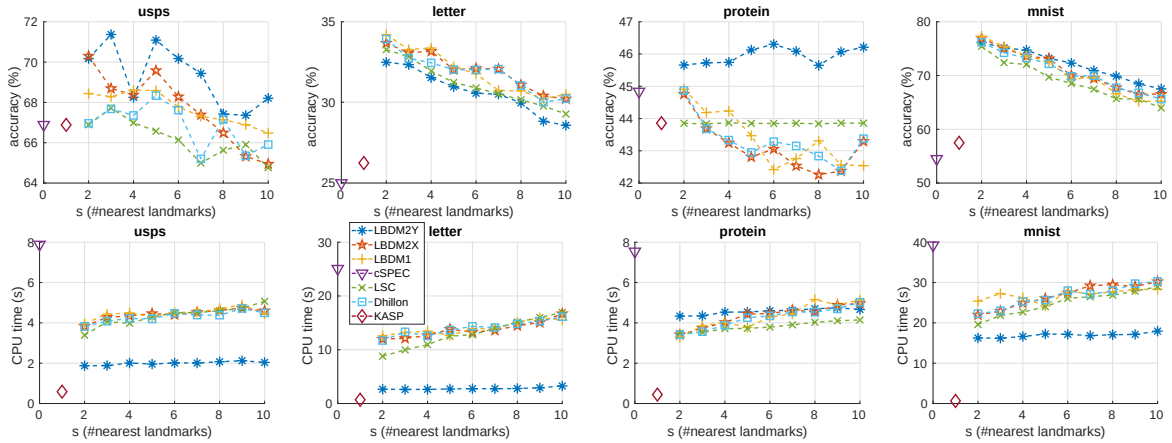

Figure 5: Effects of the parameter s (with $m = 500$ fixed all the time). Top row: clustering accuracy; bottom row: CPU time. Since KASP fixes $s = 1$ and cSPEC requires no sparsification, we have respectively plotted their accuracy rates at $s = 1$ and 0 in each plot.

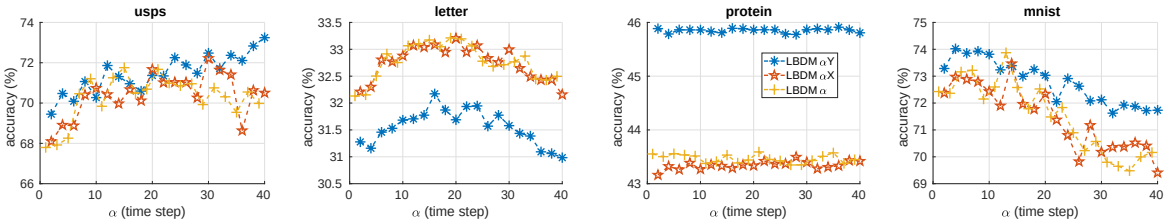

Figure 6: Effects of the parameter α on LBDM (with $m = 500$ and $s = 5$ fixed all the time).

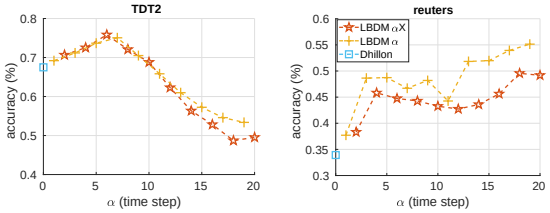

Figure 7: Clustering accuracy of LBDM (with $1 \le \alpha \le 20$) and Dhillon's method (shown at $\alpha = 0$) on two text data sets.

4.4 LBDM with bipartite graphs between documents and terms

In this section we conduct an extra experiment to show that we can easily adapt LBDM (Alg. 1) for the original bipartite graph model by Dhillon (2001), which consists of documents and terms, by simply treating the terms as the "landmarks" and using the document-term frequency matrix A as the affinity matrix between the two components of the bipartite graph. We then carry out the remaining steps of Alg. 1, using either the direct clustering method (for even α) or the co-clustering method (for odd α), and still denote them as LBDM$^{(\alpha,X)}$ and LBDM$^{(\alpha)}$.

We compare these two methods for $1 \le \alpha \le 20$ with Dhillon's co-clustering algorithm using two news data sets, *TDT2* and *Reuters21578*.[5] Because of the much varied cluster sizes, we focus on the top 30 categories in each data set. The clustering accuracy of the three methods on both data sets is reported in Fig. 7. It is clear that the use of diffusion coordinates on the bipartite graph (for small α) considerably improves the documents clustering accuracy.

5 Conclusions

We presented a landmark-based scalable spectral clustering approach by a novel combination of diffusion maps and bipartite graphs. Our experiments showed that the proposed algorithm achieved very stable and competitive accuracy while running fast. We conclude that LBDM can be used as a very promising new alternative to current large-scale spectral clustering methods.

Acknowledgments

We thank the anonymous reviewers for helpful feedback. G. Chen was supported by a Simons Foundation Collaboration Grant for Mathematicians while conducting this research.

[5] Available at http://www.cad.zju.edu.cn/home/dengcai/Data/TextData.html

References

D. Cai and X. Chen. 2015. Large scale spectral clustering via landmark-based sparse representation. *IEEE Transactions on Cybernetics*, 45(8):1669–1680.

G. Chen. 2018. Scalable spectral clustering with cosine similarity. In *Proceedings of the 24th International Conference on Pattern Recognition (ICPR)*, Beijing, China. To appear.

X. Chen and D. Cai. 2011. Large scale spectral clustering with landmark-based representation. In *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence*.

A. Choromanska, T. Jebara, H. Kim, M. Mohan, and C. Monteleoni. 2013. *Fast Spectral Clustering via the Nyström Method*, volume 8139 of *Algorithmic Learning Theory. ALT 2013. Lecture Notes in Computer Science*. Springer, Berlin, Heidelberg.

F. R. K. Chung. 1996. *Spectral graph theory*, volume 92 of *CBMS Regional Conference Series in Mathematics*. AMS.

R. Coifman and S. Lafon. 2006. Diffusion maps. *Applied and Computational Harmonic Analysis*, 21(1):5–30.

I. Dhillon. 2001. Co-clustering documents and words using bipartite spectral graph partitioning. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '01, pages 269–274, New York, NY, USA.

I. Dhillon, Y. Guan, and B. Kulis. 2004. Kernel k-means: Spectral clustering and normalized cuts. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 551–556.

C. Fowlkes, S. Belongie, F. Chung, and J. Malik. 2004. Spectral grouping using the nystr´om method. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 26(2):214–225.

J. Liu, C. Wang, M. Danilevsky, and J. Han. 2013. Large-scale spectral clustering on graphs. In *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence*, pages 1486–1492. AAAI Press.

U. von Luxburg. 2007. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416.

M. Meila and J. Shi. 2001. A random walks view of spectral segmentation. In *Proceedings of the Eighth International Workshop on Artificial Intelligence and Statistics (AISTATS), Key West, Florida, USA*.

Y. Moazzen and K. Tasdemir. 2016. Sampling based approximate spectral clustering ensemble for partitioning data sets. In *Proceedings of the 23rd International Conference on Pattern Recognition (ICPR), Cancun, Mexico*. IEEE.

A. Ng, M. Jordan, and Y. Weiss. 2001. On spectral clustering: Analysis and an algorithm. In *Advances in Neural Information Processing Systems 14*, pages 849–856.

T. Sakai and A. Imiya. 2009. *Fast Spectral Clustering with Random Projection and Sampling*, volume 5632 of *Machine Learning and Data Mining in Pattern Recognition. MLDM 2009. Lecture Notes in Computer Science*. Springer, Berlin, Heidelberg.

J. Shi and J. Malik. 2000. Normalized cuts and image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(8):888–905.

K. Tasdemir. 2012. Vector quantization based approximate spectral clustering of large datasets. *Pattern Recognition*, 45(8):3034–3044.

R. Tibshirani, G. Walther, and T. Hastie. 2001. Estimating the number of clusters in a data set via the gap statistic. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 63(2):411–423.

L. Wang, C. Leckie, R. Kotagiri, and J. Bezdek. 2011. Approximate pairwise clustering for large data sets via sampling plus extension. *Pattern Recognition*, 44:222–235.

L. Wang, C. Leckie, K. Ramamohanarao, and J. Bezdek. 2009. *Approximate Spectral Clustering*, volume 5476 of *Advances in Knowledge Discovery and Data Mining. PAKDD 2009, Lecture Notes in Computer Science*. Springer, Berlin, Heidelberg.

D. Yan, L. Huang, and M. Jordan. 2009. Fast approximate spectral clustering. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 907–916.

L. Zelnik-Manor and P. Perona. 2004. Self-tuning spectral clustering. In *Advances in Neural Information Processing Systems 17*, pages 1601–1608.