Unsupervised Large Graph Embedding Based on Balanced and Hierarchical K-means

Feiping Nie, Wei Zhu, and Xuelong Li, *Fellow, IEEE*

Abstract—There are many successful spectral based unsupervised dimensionality reduction methods, including Laplacian Eigenmap (LE), Locality Preserving Projection (LPP), Spectral Regression (SR), *etc*. We find that LPP and SR are equivalent if the symmetric similarity matrix is doubly stochastic, Positive Semi-Definite (PSD) and with rank p, where p is the reduced dimension. Since solving SR is believed faster than solving LPP based on some related literature, the discovery promotes us to seek to construct such specific similarity matrix to speed up LPP solving procedures. We then propose an unsupervised linear method called Unsupervised Large Graph Embedding (ULGE). ULGE starts with a similar idea as LPP but adopts an efficient approach to construct anchor-based similarity matrix and then performs spectral analysis on it. Moreover, since conventional anchor generation strategies suffer kinds of problems, we propose an efficient and effective anchor generation strategy, called Balanced K-means based Hierarchical K-means (BHKH). The computational complexity of ULGE can reduce to $O(ndm)$, which is a significant improvement compared to conventional methods need $O(n^2d)$ at least, where n, d and m are the number of samples, dimensions, and anchors, respectively. Extensive experiments on several publicly available datasets demonstrate the efficiency and effectiveness of the proposed method.

Index Terms—Large Graph Embedding, Balanced K-means based Hierarchical K-means, Anchor Based Graph, Dimensionality Reduction

1 Introduction

With the rapid development of technologies, increasing amounts of data become more and more difficult to storage, process and transfer especially in information processing fields, e.g. pattern recognition, machine learning and data mining. As one of the most efficient and direct approaches to deal with large scale data, dimensionality reduction attracts many researchers' attentions, and lots of successful methods have been proposed. Recently, spectral based methods have been proven successful in many fields, and researchers have developed various spectral based approaches for classical unsupervised tasks, e.g. clustering [1], [2], unsupervised feature selection [3], [4], and also unsupervised dimensionality reduction [5], [6], [7]. The framework of spectral based methods are similar. They always first construct similarity graph, and then perform spectral analysis to find the low-dimensional manifold structure of the original data. However, the excessive amount of data bring lots of challenges, and conventional spectral-based methods cannot handle the large-scale data properly in real life application.

The most popular spectral-based unsupervised dimensionality reduction methods include Laplacian Eigenmap (LE) [5], Locality Preserving Projections (LPP) [7], Spectral Regression (SR) [6], etc. LPP and SR are actually two different types of linearization of LE. Nevertheless, we discover that if the symmetric similarity matrix is PSD, doubly stochastic and with rank p, LPP is equivalent to SR, where p is the reduced dimension. Since solving LPP by SR is believed faster than directly solving LPP in some cases [6], we thus seek methods to construct such similarity matrix which enables us to solve LPP more efficiently. Nonetheless, conventional spectral-based methods always construct almost full-rank similarity matrix by k-nearest neighbors, which makes both graph construction and spectral analysis time consuming, and the computational complexity is $O(n^2d)$ at least, which is surely unbearable for the data which contain hundreds of thousands samples. Recently, Graph neural network (GNN) has been widely used for handling structured data [8] [9] [10] [11]. Wang *et al.,* apply GNN to heterogeneous graph and propose node level and semantic level aggregation [9]. However, most of these methods could only be applied to small or medium scale graph due to the memory limitation. Zhang *et al.,* apply LSTM to aggregate heterogeneous information and node level information [10]. They propose to first use RWR (Random Walk with Restart) to sample a local graph, which, on the one hand, would require a predefined graph, and on the other hand, limits the global information aggregation. Our method is potentially helpful to enable GNN to handle large scale graph without local sampling, which would be one of our future work.

To alleviate the problem, inspired by recent progresses on scalable semi-supervised learning [12], large scale spectral clustering [13] [14] [15] [16], and large scale dimensionality reduction [17] [18], we propose an efficient method to perform large scale dimensionality reduction, named Unsupervised Large Graph Embedding (ULGE). ULGE adopts a quite efficient way to construct similarity graph by anchor-based strategy [12]. Moreover, the constructed graph can be proven to be symmetric, PSD, doubly stochastic and with rank p, and thus we can perform SR to efficiently obtain the solution of LPP with such similarity graph.

For anchor-based strategy, generally speaking, it first generates many anchors, and then uses these anchors to construct similarity matrix. The anchor generation strategy is obviously crucial for the final performance, and the two most popular methods used for anchor generation are ran-

Feiping Nie, Wei Zhu, and Xuelong Li are with School of Computer Science and Center for OPTical IMagery Analysis and Learning (OPTIMAL), Northwestern Polytechnical University, Xi'an 710072, Shaanxi, P. R. China (feipingnie@gmail.com; zwvews@gmail.com).

dom selection and k-means generation [19], [20]. Random selection is efficient while the performance is usually poor. K-means strategy achieves better performance but with undesirable time cost and things get worse when we need to generate thousands of anchors. As a general rule, the more samples we have, the more anchors we need to get good enough performance, which means that k-means strategy is not suitable for large scale data. We thus propose a very efficient approach to generate representative anchors for large scale datasets, called Balanced K-means based Hierarchical K-means (BKHK). BKHK adopts balanced binary tree structure and has a computational complexity of $O(nd\log(m)t)$, which is a great reduction compared with normal k-means $O(ndmt)$, where t is the number of iterations. BKHK algorithm can speed up the graph construction procedure a lot. For example, ULGE only needs about 30 seconds to get similarity graph on MNIST data set, which contains 70000 sample and 784 features. Combined with the efficient spectral analysis, although ULGE is based on LPP, it is extremely efficient especially for large scale problem.

Moreover, since orthogonal constraint has some desirable properties and often gets better performance [21], [22], we also give a version of ULGE with orthogonal constraint which is denoted as OULGE. It is worthwhile to note that OULGE adds the orthogonal constraint in an elegant way and the extra computation cost is negligible.

Three main contributions of this paper are listed as follows:

1) We discover that LPP is equivalent to SR under certain condition, which is of importance in guiding the design of efficient spectral based dimensionality reduction method.
2) We propose a spectral based unsupervised linear dimensionality reduction method, i.e. ULGE. ULGE adopts anchor-based graph and BKHK algorithm, which makes it efficient and effective to construct large graph. We also give the orthogonal version as OULGE.
3) Comprehensive experiments on several large scale data sets demonstrate the efficiency of ULGE and OULGE.

We first introduce some notations that are used throughout the paper. For matrix $M \in \mathbb{R}^{r\times t}$, the (i,j)-th entry of M is denoted by m_{ij}, the transpose of the i-th row of M is denoted by $m_i \in \mathbb{R}^{t\times 1}$. The trace of M is denoted by $Tr(M)$. The transpose of matrix M is denoted by M^T. The inverse of matrix M is denoted by M^{-1}. The Frobenius norm of M is denoted by $\|M\|_F$. $\mathbf{1}$ is the column vector of all ones. Other notations are summarized in Table 1

TABLE 1
Summary of notations

$X \in \mathbb{R}^{n\times d}$	Data matrix
$L \in \mathbb{R}^{n\times n}$	Laplacian matrix
$A \in \mathbb{R}^{n\times n}$	Similarity matrix
$Y \in \mathbb{R}^{n\times p}$	Low dimensional embedding
$D \in \mathbb{R}^{n\times n}$	Diagonal, degree matrix
$H = I - \frac{1}{n}\mathbf{1}\mathbf{1}^T$	Centering matrix
$W \in \mathbb{R}^{d\times p}$	Projection matrix
$F \in \mathbb{R}^{n\times n}$	Eigenvector Matrix of A
$\Lambda \in \mathbb{R}^{n\times d}$	Diagonal, eigenvalue matrix of A
$G \in \mathbb{R}^{n\times 2}$	Index matrix
$C \in \mathbb{R}^{d\times 2}$	Clustering center
I	Identity matrix
n	# samples
d	# dimensions
m	# anchors
p	# reduced dimensions

2 Background

In the past decades, many spectral based dimensionality reduction methods have been proposed. Most original attempts are non-linear methods, e.g. LE [5], LLE [23], and ISOMAP [24]. Soon after, linear methods, e.g. LPP [7] and SR [6], have been proposed to deal with out-of-sample problem. There are kinds of linear dimensionality reduction methods. To sum these linear methods up , Yan *et al.* and Nie *et al.* propose two different linear dimensionality reduction frameworks, called graph embedding [25] and Flexible Manifold Embedding (FME) [26], respectively.

Graph Embedding has been applied into many areas in recently years. For dimensionality reduction, various types of intrinsic graphs and penalty graphs are constructed first, and then graph embedding methods try to preserve the similarity or dissimilarity between samples at embedded space. Different choices of intrinsic graphs and penalty graphs lead to a wide range of graph embedding methods, e.g., LDA, LPP [7], LLE [23], MFA [25]. Kernel methods and tensor methods are also applied in graph embedding methods [27], [28]. Recently, some methods have been proposed to combine graph embedding with spare representation [29], Canonical correlation analysis [30] and non-negative matrix factorization [31]. Next, we briefly introduce some unsupervised dimensionality reduction methods which are most related to this paper.

Given a data matrix $X = [x_1, \ldots, x_n]^T \in \mathbb{R}^{n\times d}$, where $x_i \in \mathbb{R}^d$ denotes the i-th sample, let $A \in \mathbb{R}^{n\times n}$ be the similarity matrix constructed by k-nearest neighbor approach, and a_{ij} is the similarity between x_i and x_j. We now seek to find the low dimensional embedding, i.e. $Y \in \mathbb{R}^{n\times p}$, where p is the reduced dimension, LE performs dimensionality reduction via solving following problem [5]:

$$\min_{Y} Tr((Y^T DY)^{-1} Y^T LY), \tag{1}$$

where $D \in \mathbb{R}^{n\times n}$ is a diagonal matrix and the i-th entry is defined as $\sum_{j=1}^{n} a_{ij}$, $L = D - A$ denotes Laplacian matrix [32]. LE aims to maintain the pairwise similarity at the reduced feature space. To keep things simple, Problem (1) is ratio trace representation [33]. Problem (1) can be tackled by generalized Rayleigh-Ritz theorem and the solution is formed by the p eigenvectors of $D^{-1}L$ corresponding to the p smallest eigenvalues. LE is a non-linear method, and it cannot directly deal with new coming data.

He and Niyogi propose LPP to deal with out-of-sample problem [7]. LPP adopts the projection function $W \in \mathbb{R}^{d\times p}$ by replacing Y with XW. However, as pointed by [34], [35], LPP does not have shift invariant property. To alleviate the problem, shift-invariant LPP can be performed as follows

[34]

$$\min_W Tr((W^T X^T HDHXW)^{-1} W^T X^T HLHXW), \quad (2)$$

where H is defined by

$$H = I - \frac{1}{n}\mathbf{1}\mathbf{1}^T. \quad (3)$$

Moreover, to avoid ill-posed problem, LPP can be reformulated as:

$$\min_W Tr((W^T(X^T HDHX + \alpha I)W)^{-1} W^T X^T HLHXW), \quad (4)$$

where α is the regularization parameter. LPP can be seen as the linearisation extension of LE, which is a successful method and computationally efficient for training data as well as new data. However, there are also some disadvantages besides the performance degradation with non-linearly distributed data. Cai *et al.* point out that LPP is time consuming when calculating the generalized eigenvalue problem of the dense matrices $X^T LX$ and $X^T DX$ [6].

Cai *et al.* then propose a different linearization extension method called SR [6]. SR first obtains the low dimensional embedding Y^* by solving problem (1), then gets the projection matrix via solving a regression problem:

$$\min_W \|HXW - Y^*\|_F^2. \quad (5)$$

Also, the regularized SR can be formulated as:

$$\min_W \|HXW - Y^*\|_F^2 + \alpha\|W\|_F^2. \quad (6)$$

Problem (6) can be efficiently solved by some well-studied algorithms, e.g., LSQR [36]. SR is believed more efficient since it only needs to calculate generalized eigenvalue problem of the sparse matrices L and D [6].

As pointed in FME [26], LPP and SR are two extreme cases of FME framework. However, we will show that if the symmetric similarity matrix is doubly stochastic, PSD with rank p, LPP is equivalent to SR.

3 Equivalence between SR and LPP

LPP performs dimensionality reduction via solving problem (2). Now let similarity matrix A be doubly stochastic which is defined as a square matrix of nonnegative real numbers and each of whose rows and columns sums to 1. Degree matrix D is thus equal to identity matrix, i.e. $D = I$. Then noting that H is an idempotent matrix, substituting $D = I$ and $L = D - A$ into LPP formulation, we get

$$\max_W Tr((W^T X^T HXW)^{-1} W^T X^T HAHXW) \quad (7)$$

Then, supposing that similarity matrix A is PSD with rank p, A thus can be decomposed by eigenvalue decomposition as:

$$A = F\Lambda F^T, \quad (8)$$

where $\Lambda \in \mathbb{R}^{n\times n} = diag(\lambda_1, \ldots, \lambda_n)$, and $\lambda_1 \geq \cdots \geq \lambda_p > 0 = \lambda_{p+1} = \cdots = \lambda_n$ are the eigenvalues of A, $F \in \mathbb{R}^{n\times n}$ is eigenvector matrix. It is easy to know that Eq. (8) can be rewritten as

$$A = F_p \Lambda_p F_p^T, \quad (9)$$

where $F_p \in \mathbb{R}^{n\times p}$ is the first p columns of F, and $\Lambda_p \in \mathbb{R}^{p\times p} = diag(\lambda_1, \ldots, \lambda_p)$. Then, substituting Eq. (9) into problem (7), we can formulate LPP as

$$\max_W Tr((W^T X^T HXW)^{-1} W^T X^T HF_p \Lambda_p F_p^T HXW). \quad (10)$$

On the other hand, given such similarity matrix A, the solution to LE, i.e. problem (1), is actually F_p. Therefore, the formulation of SR, i.e. problem (5), can be rewritten as

$$\min_W \|HXW - F_p\|_F^2. \quad (11)$$

Interestingly, although LPP and SR are two different spectral based dimensionality reduction methods, it can be proven that the solution space of problem (10) is equivalent to that of problem (11). In other words, LPP is equivalent to SR if symmetric similarity matrix A is PSD, doubly stochastic and with rank p.

Before presenting further proof, we first show an interesting observation of LPP as follows:

Lemma 1. *If $\hat{W}$ is the optimal solution to LPP, i.e. problem (2), $\hat{W}R$ is still an optimal solution, where $R \in \mathbb{R}^{p\times p}$ is an arbitrary invertible matrix.*

A proof of Lemma 1 is provided in the Appendix.

Then, as a simple corollary, we can get that LE, i.e. problem (1), also has same property, and we thus get following lemma for LE and SR as

Lemma 2. *Given F_p and $\hat{W}$, which are the optimal solutions to LE (i.e. problem (1)) and SR (i.e. problem (5)) respectively, F_pR and $\hat{W}R$ are also the optimal solution to LE and SR, respectively, where $R \in \mathbb{R}^{p\times p}$ is an arbitrary invertible matrix.*

A proof of Lemma 2 is provided in the Appendix.

Lemma 1 and Lemma 2 indicate that the optimal solution to LPP, SR and LE are actually the column space of $\hat{W}$, $\hat{W}$, and F_p respectively. We then give a simple lemma as follows:

Lemma 3. *The column space of $A^{-1}BB^T$ is exact same as that of $A^{-1}B$.*

A proof of Lemma 3 is provided in the Appendix.

Based on Lemma 3, we have a simple corollary as

Corollary 1. *Given that Λ is diagonal and positive definite, the column space of $A^{-1}B\Lambda B^T$ is exact same as that of $A^{-1}B$.*

A proof of Corollary 1 is provided in the Appendix.

Based on Lemma 1, Lemma 2 and Corollary 1, we come up with following theorem:

Theorem 1. *If the symmetric similarity matrix A is doubly stochastic, PSD and with rank p, LPP is equivalent to SR.*

Proof. As mentioned above, with such specific similarity matrix, problem (10) and problem (11) are actually LPP and SR, respectively. According to Lemma 1 and Lemma 2, the solutions to problem (10) and problem (11) are the column spaces of $(X^T HX)^{-1} X^T HF_p \Lambda_p F_p^T HX$ and $(X^T HX)^{-1} X^T HF_p$, respectively. Moreover, based on Corollary 1, we know that the column spaces of these two matrix are same, in other words, the solutions of LPP is same as that of SR, we then complete the proof. □

We further show that regularized LPP, i.e. problem (4), is equivalent to regularized SR, i.e. problem (6), by following theorem:

Theorem 2. *If similarity matrix A is symmetric, doubly stochastic, PSD and with rank p, regularized LPP, i.e. problem (4), is equivalent to regularized SR, i.e. problem (6).*

The proof is similar to Theorem 1, and we omit it here.

We next show some important significances of Theorem 1 as well as Theorem 2 as follows. For simplicity, we take Theorem 1 as an example:

1) Although LPP and SR are different methods, with certain similarity matrix, SR does make sense in the framework of LPP, and vice versa.
2) Theorem 1 gives a new approach to tackle LPP problem with certain similarity matrix, i.e. through solving SR problem.
3) Theorem 1 gives a new approach to tackle SR problem with certain similarity matrix, i.e. through solving LPP problem.

It goes without saying that the first point is important. For the second point, as pointed by Cai *et al.* [6], LPP needs to perform generalized eigenvalue decomposition on a dense matrix which is pretty time consuming, they then propose a more efficient method, i.e. SR. According to Theorem 1, instead of directly tackling LPP problem, we can now efficiently tackle specific SR problem to get the solution of LPP's. Moreover, as pointed by [37] [38], doubly stochastic similarity matrix always results in promising performance. Therefore, it is reasonably to construct the required similarity matrix in Theorem 1 to get both high efficiency and high performance.

4 Unsupervised Large Graph Embedding

In this section, we propose Unsupervised Large Graph Embedding (ULGE). We first show how to efficiently construct the anchor-based graph of large scale data, and then perform spectral analysis on the obtained graph.

4.1 Similarity Matrix Construction

Theorem 1 requires that similarity matrix A must be doubly stochastic, PSD and with rank p. However, conventional spectral based methods always adopt k-nearest neighbors to construct similarity matrix, which is not only time consuming but also impossible to obtain such similarity matrix. Moreover, as far as we know, it is difficult to construct doubly stochastic, PSD and rank-p similarity matrix simultaneously, and we thus propose a two-step approach. For detail, we first obtain doubly stochastic and PSD similarity matrix, and then construct rank-p matrix.

4.1.1 Anchor Generation with BKHK Algorithm

Following recent studies on scalable semi-supervised learning [12], we adopt an efficient strategy to construct doubly stochastic and PSD similarity matrix, i.e. anchor-based strategy. Anchor generation strategy is believed crucial for the final performance. Random selection strategy is extremely fast, but it cannot guarantee that the selected anchors are good enough to construct the similarity graph of the whole data, which means that the performance is usually poor in practice. By contrast, k-means strategy can generate representative anchors which surely makes it achieve better performance. Nevertheless, k-means has high computational complexity which makes simply k-means strategy almost impossible to be applied for large scale data. There are two common used methods to speed up the k-means procedure, i.e., performing down-sampling on the data or early stopping k-means iterations [17]. However both of them are likely to have performance degradation in practice. We propose Balanced K-means based Hierarchical K-means algorithm (BKHK) to tackle this problem. BKHK can generate representative anchors efficiently especially for large scale data. For detail, BKHK adopts balanced binary tree structure, and iteratively segments the data into two clusters with same number of samples.

We now show the detail of BKHK algorithm. Let us begin with binary k-means which can be formulated as follows:

$$\min_{G\in Ind,\mathbf{1}^TG=[\kappa,\iota]} \|X - GC^T\|_F^2 \tag{12}$$

where $C \in \mathbb{R}^{d\times 2}$ is the center of the cluster, $G \in \mathbb{R}^{n\times 2}$ is the index matrix, g_{i1} equals 1 if the i-th sample belongs to the first cluster, or g_{i2} equals 1 otherwise, and $\mathbf{1}$ is the column-vector of all ones. Moreover, κ and ι are the number of samples in these two clusters, and we clearly have $\kappa+\iota = n$. We can simply set $\kappa = \lfloor \frac{n}{2} \rfloor$ to make different clusters contain same amount of samples (If n is an odd number, we set $\kappa = \frac{n-1}{2}$). We then rewrite problem (12) as:

$$\min_{G\in Ind,\mathbf{1}^TG=[\kappa,\iota]} \sum_{i=1}^{n}\sum_{k=1}^{2} \|x_i - c_k\|_2^2 g_{ik} \tag{13}$$

where c_k is the k-th column of C. For convenience, we define matrix $E \in \mathbb{R}^{n\times 2}$ and the (i,j)-th entry of E is denoted as $e_{ik} = \|x_i - c_k\|_2^2$. Then, we rewrite problem (13) as

$$\min_{G\in Ind,\mathbf{1}^TE=[\kappa,\iota]} Tr(E^TG) \tag{14}$$

Let g denote the first column of G. Since G is an index matrix, the second column can be denoted as $(\mathbf{1} - g)$, and then problem (14) can be rewritten as

$$\min_{g\in\{0,1\},\mathbf{1}^Tg=\kappa} g^Te_1 + (\mathbf{1} - g)^Te_2 \tag{15}$$

where e_1 and e_2 are the first and second column of E, respectively. Then, we arrive at

$$\min_{g\in\{0,1\},\mathbf{1}^Tg=\kappa} g^T(e_1 - e_2) \tag{16}$$

The solution to problem (16) is intuitive. We assign $g_i = 1$ when the i-th element of $e_1 - e_2$ is the κ minimum of all its elements. Refer to Fig. 1 for an example of balanced k-means algorithm. We'd like to note that the proposed exact balanced k-means can only be conducted for binary cases.

Since the two clusters have same number of samples, we call this algorithm balanced k-means, and we hierarchically perform balanced k-means to perform the BKHK. The computational complexity of BKHK is $O(nd\log(m)t)$, and the algorithm is summarized in Algorithm 1. We'd also like to mention that hierarchy k-means has been studied by several

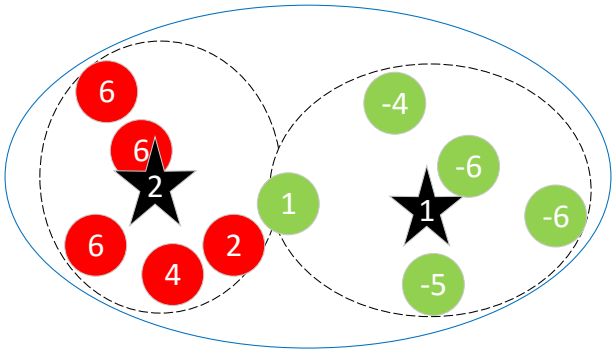


Fig. 1. Balanced k-means algorithm makes every cluster contain same number of samples. The value of $e_1 - e_2$ is indicated on every sample.

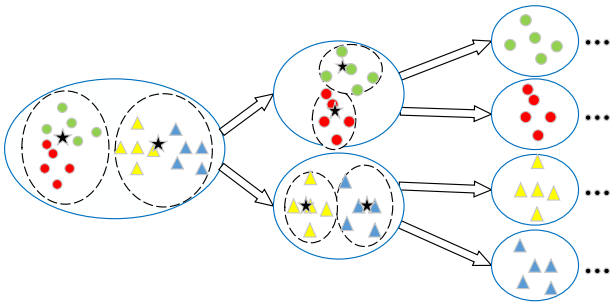


Fig. 2. BKHK algorithm has the binary tree structure, it performs balanced k-means at each node.

papers [39], but theoretically, hierarchy k-means is as fast as BKHK at best and as slow as k-means at worst. In other words, there is no theoretical guarantee for the efficiency of hierarchy k-means while BKHK is always fast. There are also several literatures about balanced k-means [40] [41]. They believe that the number of samples in each cluster should be similar, and they use kinds of regulation terms to achieve the goal. However, these methods are usually slower than k-means, and thus not suitable for large scale data.

Algorithm 1 Balanced K-means Based Hierarchical K-means (BKHK)

Input: Data matrix $X \in \mathbb{R}^{n\times d}$, number of anchors m

while not reach the termination condition **do**
  Initialize the cluster center matrix C.
  while not converge **do**
    Obtain the indicator vector g via solving problem (16).
    Obtain the indicator matrix $G = [g, 1-g]$.
    Calculate the cluster center for both of the two clusters.
  end while
  Hierarchically perform Algorithm 1 on the obtained two sub-clusters.
end while
Obtain anchor set U by calculating the center of all sub-clusters.

Output: Anchor set U.

There are two more points we would like to mention about BKHK. First, BKHK is a pretty efficient method especially for large scale data, and can be easily used to accelerate other graph based learning methods, e.g. hashing [42], clustering [43], semi-supervised learning [44], [45], dimensionality reduction [46], RBF networks [47], etc. Second, as shown previously, early stopping and down-sampling can speed up k-means a lot, and can also be adopted by BKHK for extremely large data. We also empirically study the running time between k-means and BKHK in Section 5.5.

4.1.2 Anchor-based Similarity Matrix

The similarity graph construction with the obtained anchors is a well-studied problem [12]. Let U denote the set of the generated anchors, and $U_{\langle i\rangle}$ denotes the set of k-nearest anchors for the i-th sample. Conventional methods usually use kernel based neighbor assignment strategy, e.g. Gaussian kernel $K_t(x_i, u_j) = exp(-\|x_i - u_j\|_2^2/2\sigma^2)$, but kernel based methods always bring extra parameters, e.g. bandwidth σ. Thus, we prefer to adopt a parameter-free yet effective neighbor assignment method. The neighbor assignment for the i-th sample can be seen as solving following problem [48]

$$\min_{z_i^T\mathbf{1}=1, z_i\geq 0} \sum_{j=1}^{m} h(x_i, u_j) z_{ij} + \gamma \sum_{j=1}^{m} z_{ij}^2, \tag{17}$$

where $Z \in \mathbb{R}^{n\times m}$ denotes the similarity between the i-th sample and the j-th anchor, $h(x_i, u_j)$ is the distance between the i-th sample and its j-th nearest anchor. To keep it simple, we define $h(x_i, u_j) = \|x_i - u_j\|_2^2$, which is the square of Euclidean distance. Follow [48], γ can be set as $\gamma = \frac{k}{2}h(i, k+1) - \frac{1}{2}\sum_{j=1}^{k} h(x_i, u_j)$. The solution to problem (17) is

$$z_{ij} = \begin{cases} \frac{h(x_i, u_{k+1}) - h(x_i, u_j)}{\sum_{j'=1}^{k} (h(x_i, u_{k+1}) - h(x_i, u_{j'}))}, & \text{if } j \leq k. \\ 0, & \text{otherwise.} \end{cases} \tag{18}$$

For detail derivation, see [48].

After we obtain the matrix Z, similarity matrix A then can be obtained by [12]:

$$A = Z\Delta^{-1}Z^T, \tag{19}$$

where $\Delta \in \mathbb{R}^{m\times m}$ is a diagonal matrix and the i-th entry is defined as $\sum_{j=1}^{n} z_{ji}$.

4.1.3 Low-rank Approximation for Anchor-based Similarity Matrix

The similarity matrix A obtained by solving problem (19) is symmetric, PSD, and doubly stochastic [12]. However, the rank of A is always larger than p in most cases, we thus seek a rank-p approximation of A as

$$\min_{\tilde{A}} \|\tilde{A} - A\|_F, \quad s.t. \quad rank(\tilde{A}) = p, \tag{20}$$

where $\tilde{A} \in \mathbb{R}^{n\times n}$. The optimal solution $\tilde{A}^*$ to problem (20) is $\tilde{A}^* = F_p \Lambda_p F_p^T$ according to Eckart-Young-Mirsky theorem [49]. The approximated similarity matrix $\tilde{A}^*$ is obviously PSD, symmetric and with rank p, and we now show that it is still doubly stochastic as follows:

Lemma 4. *Given matrix $A \in \mathbb{R}^{n\times n}$ is symmetric, PSD and doubly stochastic, the rank-p approximation $\tilde{A}_p$ is still doubly stochastic, where $1 \leq p \leq n$.*

A proof of Lemma 4 is provided in the Appendix.

The optimal solution to problem (20), i.e. $\tilde{A}^*$, can be directly obtained by the eigenvalue decomposition on A, however, note that Eq. (19) can be rewritten as

$$A = BB^T, \tag{21}$$

where $B = Z\Delta^{-\frac{1}{2}} \in \mathbb{R}^{n\times m}$. We then can perform Singular Value Decomposition (SVD) on B instead of performing eigenvalue decomposition on A to get the solution, and the time complexity is $O(nmk)$, where k is the number of number of nearest neighbors. At last, we obtain the specific similarity matrix which is required in Theorem 1.

4.2 Spectral Analysis with Anchor-based Graph

We build two versions of ULGE, the difference between them is just whether adopting orthogonal constraint or not.

4.2.1 ULGE

We first show the simpler method, called ULGE, i.e. without the orthogonal constraint. ULGE starts with similar idea as LPP, but to avoid ill-posed problem, ULGE adopts regularization term and can be formulated as:

$$\max_W Tr((W^T(X^THX + \alpha I)W)^{-1}W^TX^TH\tilde{A}^*HXW). \tag{22}$$

Besides, the corresponding regularized SR can be formulated as

$$\min_W \|HXW - F_p\|_F^2 + \alpha\|W\|_F^2, \tag{23}$$

where α is the ridge regression parameter. Clearly, the approximated $\hat{A}^*$ meets the condition of Theorem 2, and problem (22) then can be solved by a simple regression problem, i.e. problem (23).

4.2.2 OULGE

OULGE is similar to ULGE but with orthogonal constraint, and can be formulated as

$$\max_{W^TW=I} Tr((W^T(X^THX + \lambda I)W)^{-1}W^TX^TH\tilde{A}^*HXW) \tag{24}$$

Problem (24) can be tackled by following lemma:

Lemma 5. *The solution to OULGE, i.e. problem (24), can be divided as two steps: first, obtain $\hat{W}$ by solving problem (23), and then orthogonalize the obtained $\hat{W}$.*

Proof of Lemma 5. According to Theorem 2, if $\hat{W}$ is the solution to problem (23), then it is also the solution to problem (22). Then, from Lemma 1, we can simply get that $\hat{W}R$ is still the optimal solution to problem (22), where R is an arbitrary invertible matrix. Therefore, the orthogonalized $\hat{W}$ is still the optimal solution to problem (22), and naturally the optimal solution to problem (24). □

The orthogonalized $\hat{W} \in \mathbb{R}^{d\times p}$ can be obtained by performing SVD decomposition on $\hat{W}$. Considering that $d \ll n$ and $p \ll n$ for large scale data, the extra cost is negligible. We summarize the detail algorithm of ULGE and OULGE in Algorithm 2.

Algorithm 2 Unsupervised Large Graph Embedding

Input: Data matrix $X \in \mathbb{R}^{n\times d}$, projection dimension p, number of anchors m, regularization parameter α

1: Generate m anchors by BKHK algorithm.
2: Obtain the anchor-based similarity matrix A by solving problem (19).
3: Obtain F_p by performing SVD on matrix B, where B is defined as $Z\Delta^{-\frac{1}{2}}$.
4: Compute the projection matrix W by solving Eq. (23).
5: For OULGE, orthogonalize the obtained W. For ULGE, skip this step.

Output: Projection matrix $W \in \mathbb{R}^{d\times p}$.

4.3 Computational Complexity Analysis

Given a data matrix $X \in \mathbb{R}^{n\times d}$, the computational complexity of ULGE is demonstrated as following.

1) We need $O(nd\log(m)t)$ to obtain m anchors by BKHK algorithms, where t is the iterative number of balanced k-means.
2) We need $O(ndm)$ to construct anchor-based graph.
3) Considering that B is sparse, we need $O(nmk)$ to obtain F_p by performing SVD on matrix B, where k is the number of nearest neighbors.
4) We need $O(ndp)$ to obtain projection matrix W by solving problem (23).
5) For OULGE, we need extra $O(p^2d)$ to orthogonalize the obtained W.

Considering that $m \ll n$ and p is usually a small constant, the computational complexity of both ULGE and OULGE can be regarded as $O(ndm)$. Compared to conventional spectral based methods which need $O(n^2d)$ at least, ULGE and OULGE have great computational advantages especially on large scale data. Moreover, we also give theoretical computational analysis on directly solving LPP, solving LPP via SR and CSR in Table 2.

5 Experiments

In this section, we experimentally demonstrate the efficiency and effectiveness of the proposed methods ULGE and OULGE on 7 benchmark large scale data sets, and then show several useful analyses of the proposed ULGE and OULGE.

5.1 Data Sets

We conduct experiments on 7 different public available data sets downloaded from the LibSVM data sets page[1], UCI machine learning repository[2], and Deng Cai's page[3]. The data sets include handwritten digit (i.e. MNIST and USPS), types of moving vehicle (i.e. SensIT), object image (i.e. COIL100), forest cover type (i.e. Covtype), molecular biology (i.e. Protein), and connect-4 game (i.e. Connect-4). The detail of all the data sets are summarized in Table 3.

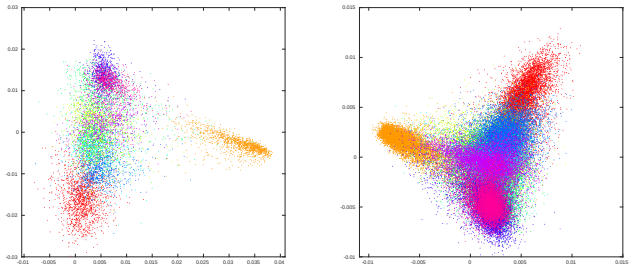
1. https://www.csie.ntu.edu.tw/∼cjlin/libsvmtools/datasets/
2. http://archive.ics.uci.edu/ml/
3. http://www.cad.zju.edu.cn/home/dengcai/Data/data.html

TABLE 2
Computational Complexity Comparisons of Different Spectral Analysis Methods

Method	Inverse	multiplication	Eigenvalue decomposition
Solving LPP by SR(ours)	$O(d^3)$	$O(nd^2 + ndp)$	$O(nmk)$
LPP	$O(d^3)$	$O(2nd^2 + ndk)$	$O(d^3)$
CSR	$O(m^3)$	$O(nm^2 + nmp)$	$O(nmk)$

TABLE 3
Data Set Description

Data Set	Sample	Feature	Class
COIL100	7200	1024	100
USPS	9298	256	10
Protein	24387	357	3
Connect-4	67557	126	3
MNIST	70000	784	10
SensIT	98528	100	3
Covtype	581012	54	7



(a) USPS (b) MNIST

Fig. 3. Visualisation of ULGE results with reduced dimension 2.

5.2 Comparison Methods

To experimentally validate the advantages of ULGE and OULGE, we compare it with several conventional spectral based unsupervised dimensionality reduction methods.

1) **Baseline**: K-means with all features.
2) **LE**: Laplacian Eigenmap [5].
3) **LPP**: Locality Preserving Projection [7].
4) **SR**: Spectral Regression [6].
5) **CSR**: Compressed Spectral Regression [17].
6) **HKULGE**: Unsupervised Large Graph Embedding with Hierarchical Kmeans.
7) **ULGE**: Unsupervised Large Graph Embedding with Balanced Kmeans-based Hierarchical Kmeans.
8) **OULGE**: Orthogonal Unsupervised Large Graph Embedding with Balanced Kmeans-based Hierarchical Kmeans.

CSR is a typical anchor based nonlinear method which uses k-means to generate anchors and performs a modified SR to obtain the final result. We use same Gaussian kernel for LE, LPP and SR, and the reduced dimension is set as the number of classes of the data set. We use 5-nearest neighbors to construct graph for all the methods. The regularization parameter α is default set to 0.01. For parameter m, i.e. the number of anchors, we empirically set $m = 1024$ for both CSR and the proposed methods. To achieve best performance, we set the iteration as 100 for k-means used in CSR and balanced k-means used in the proposed methods. Note that although fewer iterations can speed up CSR [17] and the proposed ULGE a lot, but will also lead to poorer performance in some cases.

5.3 Evaluation Metric

We run all methods 10 times and perform 10 times k-means on each reduced subspace to give the promise comparisons. We evaluate the k-means clustering results by ACCuracy (ACC), which is calculated as follows:

$$ACC = \frac{\sum_{i=1}^{n} \delta(t_i, map(l_i))}{n}, \tag{25}$$

where t_i is the ground truth, and l_i is the clustering label. $\delta(x, y)$ is the delta function, which equals to 1 when $x = y$, equals to 0 otherwise, and $map(x)$ denotes the best map.

We also record the average running time. All the codes used in the experiments are implemented in MATLAB R2015b. We run the algorithms on a Windows 10 machine with 3.20 GHz i5-3470 CPU, 32 GB main memory.

5.4 Clustering Results

The results of running time of all methods are reported in Table 4, and the performance is reported in Table 5. Since the differences of running time between OULGE and ULGE is negligible, we demonstrate these two methods together in Table 4. There are several interesting points which are listed as following:

Comparing the performance between spectral based methods and baseline which uses the original data to perform k-means, we recognize that dimensionality reduction not only reduces the computation cost but also refines the quality of the data. However, conventional spectral based methods are not good at dealing with large scale data set, e.g. conventional spectral based methods need more than 24 minutes for SensIT which contains 98528 samples and 4 hours for Covtype which contains 581012 samples. The main reason is that it is quite time consuming to construct the similarity graph for large data by k-nearest neighbors method.

Considering the running time which is demonstrated in Table 4, the proposed method achieves significant improvements for large scale data sets. For example, ULGE and OULGE only need 22.4 seconds for SensIT which contains 98528 samples, and get about 66.5 and 37.8 times faster than LPP and CSR respectively. Moreover, for Covtype which contains 581012 samples, ULGE and OULGE only need 126.4 seconds while conventional spectral based methods need to run almost 4 hours. Noting that although CSR also adopts anchor-based graph, it uses k-means algorithm to

TABLE 4
Running time on 7 data sets. (The best method is highlighted in bold.) (s)

Data Set	LE	LPP	SR	CSR	HKULGE(# anchors)	ULGE
COIL100	15.5	3.5	16.7	3.8	3.3(1044)	**3.0**
USPS	36.4	32.7	38.7	34.6	4.9(1047)	**4.6**
Protein	73.3	64.0	82.8	315.9	23.9(990)	**21.8**
Connect-4	398.7	322.9	400.1	458.5	18.9(1010)	**18.9**
MNIST	242.6	260.9	249.5	116.6	34.7(1012)	**32.3**
SensIT	1418.1	1399.7	1434.1	720.9	24.8(1051)	**22.4**
Covtype	14985.2	13828.1	15302.6	4340.3	134.1(1027)	**126.4**

TABLE 5
Clustering results (Mean%±STD) on 7 large scale data sets. (Top 2 rank methods are highlighted in bold.)

Data Set	Baseline	LE	LPP	SR	CSR	HK-ULGE	ULGE	OULGE
COIL100	37.8 ± 2.3	50.0 ± 1.8	41.6 ± 2.4	51.7 ± 1.2	51.8 ± 1.5	$\mathbf{56.5 \pm 0.7}$	$\mathbf{55.1 \pm 0.8}$	51.7 ± 1.1
USPS	66.9 ± 2.6	$\mathbf{68.9 \pm 6.4}$	67.1 ± 2.6	$\mathbf{68.1 \pm 4.9}$	64.7 ± 3.2	64.2 ± 1.2	64.5 ± 0.9	67.4 ± 1.7
Protein	42.7 ± 1.8	43.9 ± 0.1	44.0 ± 0.0	44.0 ± 0.0	44.1 ± 1.3	$\mathbf{44.3} \pm 0.7$	$\mathbf{44.1 \pm 0.5}$	44.1 ± 0.7
Connect-4	37.4 ± 0.4	44.3 ± 4.7	43.9 ± 5.3	$\mathbf{48.9 \pm 2.3}$	45.2 ± 4.4	43.0 ± 3.1	$\mathbf{45.8 \pm 2.0}$	45.0 ± 2.1
MNIST	55.7 ± 3.5	$\mathbf{68.4 \pm 6.2}$	51.3 ± 3.4	57.9 ± 4.5	$\mathbf{67.6 \pm 6.3}$	58.9 ± 2.3	59.3 ± 1.6	54.5 ± 4.2
SensIT	$\mathbf{69.0 \pm 4.8}$	61.7 ± 2.3	$\mathbf{69.3 \pm 2.5}$	65.2 ± 0.0	65.5 ± 1.7	68.3 ± 3.5	68.1 ± 2.5	67.6 ± 0.9
Covtype	25.1 ± 0.0	$\mathbf{39.3 \pm 2.1}$	25.2 ± 0.1	$\mathbf{33.3 \pm 1.4}$	28.7 ± 0.1	26.6 ± 0.7	26.5 ± 0.2	25.4 ± 1.5

generate anchors which even dramatically slows down the speed for some data sets, e.g. Protein and Connect-4. SR often runs little slower than LPP in practice. The reason is that although LPP needs to calculate the generalized eigenvalue problem of the dense matrices, i.e.$X^TLX \in \mathbb{R}^{d\times d}$ and $X^TDX \in \mathbb{R}^{d\times d}$, and SR only needs to tackle the sparse matrices, i.e. $L \in \mathbb{R}^{n\times n}$ and $D \in \mathbb{R}^{n\times n}$; but considering that $d \ll n$ in the used data sets, SR is naturally slower. However, as mentioned, ULGE and OULGE only need to perform SVD on $Z\Delta^{\frac{1}{2}} \in \mathbb{R}^{n\times m}$, which is not only sparse but also with low dimensions, thus eventually makes the spectral analysis of the proposed methods more efficient in most cases. Benefiting from both the efficient graph construction and spectral analysis, ULGE and OULGE become more and more superior as the growth of data size. Focus on Table 6 which demonstrates the detail time cost of the proposed method, since we adopt anchor-based graph as well as BKHK algorithm, the graph construction is extremely fast, and the spectral analysis is also faster than directly tackles LPP.

Considering the performance which is demonstrated in Table 5, the proposed methods achieve competitive performance for almost all the data sets. The reasons for the improvements should be attributed to the effectiveness of BKHK algorithm and anchor-based graph construction strategy. Comparing the performance of OULGE with that of ULGE, these two methods get relatively similar result in most cases, but the performance of ULGE seems to be more better on average.

Besides, although we fix the number of anchors in experiments, i.e. $m = 1024$, ULGE and OULGE still achieve competitive results even for giant scale data sets, e.g. SensIT and Covtype. This directly validates the effectiveness of BKHK algorithm. Nevertheless, we still suggest increasing the number of anchors to get better results in real life application. Some detail results on the variance of running time with different number of anchors are shown in Table 7. Although the time consuming of CSR seems to have a big advantage in our experimental settings, i.e., m=1024, noting that the running time of CSR grows cubically as a function of the number of anchors, thus it is not very suitable for large scale data when we need a large amount of anchors to generate good enough anchor-based graph. Moreover, although the number of anchors has a negligible influence on LPP, in fact, one the one the hand, we do not need too many anchors in most cases as shown in Figure 4, and on the other hand, the computational complexity of LPP is pretty high with small number of anchors and is also more sensitive to the number of dimensions. To sum up, we believe that ULGE is the safest choice especially for large scale high dimensional data sets.

At last, we visualization of the results of ULGE of USPS and MNIST in Fig. (3)

TABLE 6
Details of running time on 7 data sets, the first three columns record the results of anchor generation methods, while the last two columns show the efficiency of solving LPPvia SR. (s)

Data Set	K-means	HK	BKHK	Solve LPP	LPP by SR
COIL100	3.4	1.7	1.3	1.6	**0.8**
USPS	32.6	2.0	1.7	1.4	**0.3**
Protein	308.5	8.9	7.2	12.5	**6.6**
Connect-4	451.1	6.3	6.4	3.2	**3.2**
MNIST	103.5	22.3	20.1	2.6	**0.7**
SensIT	721.5	8.2	6.8	3.2	**1.6**
Covtype	4290.6	38.8	31.1	7.3	**4.5**

5.5 Comparison of Anchor Generation methods on Running Time

In this section, we experimentally study the running time between normal K-means, HK (Hierarchy Kmeans) and BKHK. The experiments are performed on MNIST data set. We adopt different number of samples and anchors (i.e. cluster number), and separately study the affects of these parameters on running time. Table 8 shows that the running time varies with different number of anchors and samples. The number of anchors

TABLE 7
Running time comparisons of our method, LPP, and CSR with different number of anchors.(s)

Methods	SensIT(# anchors)					Mnist(# anchors)				
	1024	2048	4096	8192	16384	1024	2048	4096	8192	16384
Solving LPP via SR(ours)	1.6	1.7	1.8	2.1	2.2	0.7	0.8	0.9	1.0	4.3
LPP	3.2	3.3	3.3	3.3	3.8	2.6	2.6	2.6	2.7	3.0
CSR	0.2	0.4	0.9	3.9	24.5	0.2	0.3	0.8	4.0	23.8

TABLE 8
Running time with different number of anchors and samples of K-means, HK, BKHK.(s)

# Anchors	10000 Samples			30000 Samples			50000 Samples			70000 Samples		
	K-means	HK	BKHK	K-means	HK	BKHK	K-means	HK	BKHK	K-means	HK	BKHK
256	3.2	1.9	1.8	20.6	7.6	6.9	38.5	14.2	12.5	66.3	18.7	18.2
512	4.3	2.2	2.0	30.6	7.8	7.1	43.7	14.9	13.2	101.5	19.9	18.8
1024	4.7	2.4	2.3	37.0	8.3	7.8	75.2	15.6	14.4	119.9	21.3	19.1
2048	5.3	2.8	2.7	43.5	9.0	8.8	100.8	16.5	14.9	184.9	23.0	20.4
4096	8.1	3.5	3.2	73.8	9.9	9.0	122.0	18.1	16.1	214.3	23.4	21.4

varies from$\{256, 512, 1024, 2048, 4096\}$ while that of samples varies from $\{10000, 30000, 50000, 70000\}$. As we see, the running time of BKHK and HK is much faster than normal k-means, especially with large number of anchors. Although BKHK is not so faster than HK, BKHK has more theoretical guarantees. Moreover, it is hard to control the number of generated anchors for HK, and one may need to try many times to get a desirable result in practice.

5.6 Parameter Sensitivity

In this section, we experimentally study the impact of parameters for the proposed methods. To be specific, the parameters of ULGE and OULGE are m (i.e. number of anchors), α (i.e. regularization parameter), and k (i.e. number of nearest anchors). For simplicity, we only conduct the experiments on 4 public available data sets, i.e. USPS, Protein, Connect-4, and SensIT. Besides, we also experimentally study the impact of n (i.e. number of samples) on the running time, the experiments are conducted on 4 data sets, i.e. Connect-4, MNIST, SensIT, and Covtype, with different size of samples.

We first show the effectiveness of the number of samples. To eliminate the impact of other factors which surely exist in Table 4, we conduct the experiments on different subsets of one data set to show it more clearly, and 4 data sets are adopted in the experiments, i.e. Connect-4, MNIST, SensIT and Covtype. As for Connect-4, SensIT and Covtype, since the data are imbalance and contain pretty fewer classes, we simply use first several tens of thousands of samples to make up the subsets which leaves us no choice but only give the running time without performance. And for MNIST, the subsets are formed by different number of classes, and there are about 7000 samples per class. The results of all data sets are demonstrate in Table 9, Table 10, Table 11 and Table 12, respectively. We can easily get that the running time of ULGE and OULGE increases linearly with respect to the number of samples, and for conventional spectral based methods, i.e. LE, LPP and SR, the running time increases quadratically. It is worthwhile to note that although CSR theoretically has linear computational complexity with respect to the number of samples and is certainly efficient than conventional methods for large scale data, the high computational complexity of k-means makes it still hard to deal with large scale data properly. For example, CSR is even much slower than conventional methods on Connect-4 since the k-means algorithm needs over 500 seconds to converge for all the 67557 samples.

As we vary the number of selected anchors, the performance variances of OULGE and ULGE are demonstrate in Figure 4. In general, Figure 4 shows that the more anchors we selected, the better performance we will get. However, if we use over large number of anchors, the performances stay stable. Figure 5 shows the running time variances of ULGE and OULGE as we change the number of selected anchors. We can easily conclude that the running time almost linearly increases with the number of selected anchors. Figure 6 shows the performance variance of OULGE and ULGE as we change the value of regularization parameter α. The results demonstrate that the performance curves vary among different data sets. Therefore, it is difficult to get the best setting and we simply fix it as 0.01. And also, the proposed OULGE and ULGE are both robust to α to some extent, and achieve comparable performance even in the worst case. As we vary the number of nearest anchors, the performance variances of OULGE and ULGE are demonstrated in Figure 7. Overall, both of OULGE and ULGE are robust, and it is a proper choice to set the number of nearest anchors as 5.

6 Conclusions

In this paper, we propose an interesting theorem about the equivalence between LPP and SR, which are believed as two different linear dimensionality reduction methods. Through analysis, if the rank p similarity matrix is PSD, symmetric and doubly stochastic, LPP is equivalent to SR, where p is the reduced dimension. Inspired by this theorem, we then propose efficient dimensionality reduction methods, called ULGE and OULGE. The proposed methods start with similar idea as LPP and adopt anchor-based strategy to construct similarity matrix. To alleviate the high computation cost of anchor generation, we propose Balanced K-means based Hierarchical K-means (BKHK) algorithm. BKHK generates

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. Citation information: DOI 10.1109/TKDE.2020.3000226, IEEE Transactions on Knowledge and Data Engineering

TABLE 9
Running time on subsets of Connect-4.(s)

# Sample	LE	LPP	SR	CSR	HKULGE	ULGE
10000	9.5	9.3	9.8	16.1	**2.4**	**2.5**
20000	34.0	32.8	34.3	60.2	5.5	**5.4**
30000	74.1	73.2	75.3	127.8	9.1	**8.8**
40000	129.5	128.4	130.8	260.3	**10.9**	**11.1**
50000	201.8	198.3	203.2	337.9	**13.2**	**13.4**
60000	289.5	284.1	291.0	387.6	15.8	**15.8**
67557	360.8	354.3	365.6	537.2	18.9	**18.9**

TABLE 10
Running time on subsets of MNIST.(s)

# Sample	LE	LPP	SR	CSR	HKULGE	ULGE
14000	8.7	8.9	10.0	11.6	4.7	**4.5**
21000	18.6	19.2	21.9	18.2	7.5	**6.9**
28000	34.2	38.7	36.1	32.3	11.5	**10.3**
35000	54.3	61.9	56.6	42.7	14.2	**13.4**
42000	78.9	82.7	84.5	51.3	16.8	**15.1**
49000	110.7	112.7	117.4	65.3	22.7	**21.2**
56000	146.3	154.4	169.9	82.0	26.1	**24.3**
63000	203.3	195.9	206.6	105.2	31.5	**29.3**
70000	242.6	260.9	249.5	116.6	34.6	**32.3**

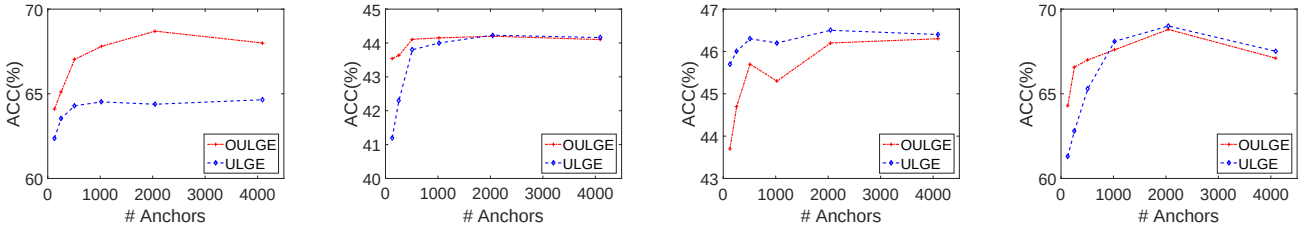
TABLE 11
Running time on subsets of SensIT.(s)

Sample	LE	LPP	SR	CSR	HKULGE	ULGE
10000	15.0	14.3	14.4	15.9	2.6	**2.4**
20000	51.6	51.7	51.6	38.9	4.8	**4.2**
30000	116.7	115.0	117.8	93.0	7.2	**6.1**
40000	203.1	199.9	204.9	148.3	8.9	**8.0**
50000	329.6	317.1	335.3	248.4	12.0	**10.5**
60000	491.0	471.9	495.1	324.0	13.8	**12.3**
70000	687.2	654.4	688.7	403.9	16.1	**14.9**
80000	890.8	859.0	887.8	488.7	17.8	**16.3**
90000	1171.6	1130.2	1180.5	558.9	21.6	**19.6**
98528	1418.8	1399.5	1434.7	680.9	23.5	**21.2**

TABLE 12
Running time on subsets of Covtype.(s)

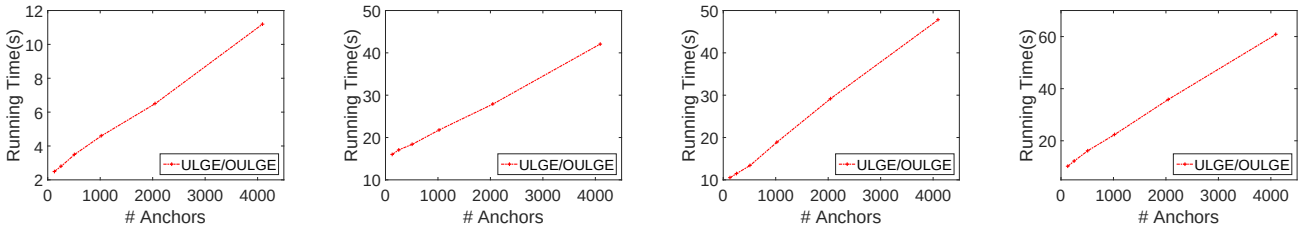
# Sample	LE	LPP	SR	CSR	HKULGE	ULGE
50000	127.2	98.5	126.1	185.0	8.1	**7.5**
100000	493.0	413.8	490.7	549.3	15.4	**14.5**
150000	1134.5	958.5	1101.6	893.9	24.2	**23.0**
200000	1997.2	1748.5	1983.5	1250.1	34.4	**32.0**
250000	3456.6	2871.2	3172.3	1709.7	42.7	**40.5**
300000	4651.9	4201.1	4638.5	2117.5	56.1	**53.3**
350000	6574.3	6302.2	6614.3	2510.4	69.5	**67.7**
400000	–*	–*	–*	2908.2	78.7	**75.8**
450000	–*	–*	–*	3301.7	98.4	**94.5**
500000	–*	–*	–*	3700.0	106.8	**102.3**
550000	–*	–*	–*	4095.9	121.4	**116.1**
581012	–*	–*	–*	4340.3	134.1	**126.4**

* More than 2 hours.



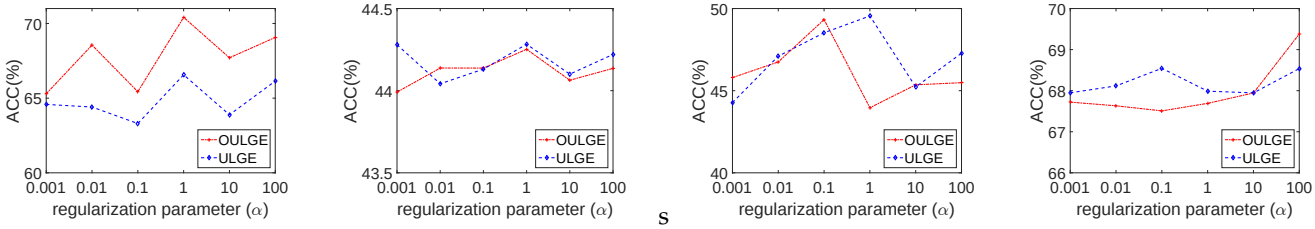
(a) USPS (b) Protein (c) Connect-4 (d) SensIT

Fig. 4. The performance of ULGE vs. # anchors.



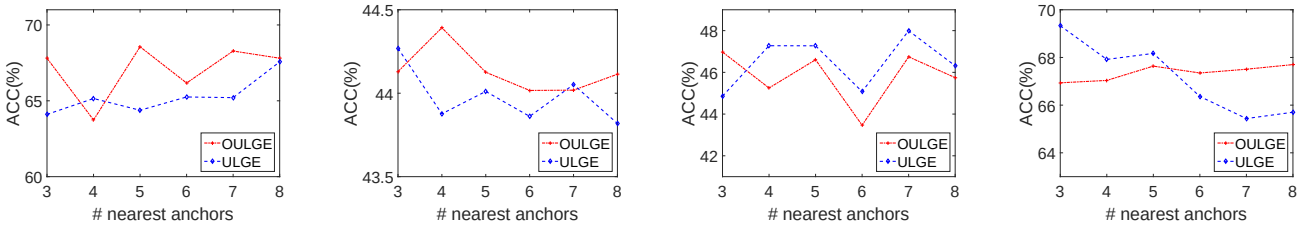
(a) USPS (b) Protein (c) Connect (d) Sensit

Fig. 5. Running time of ULGE vs. # anchors.



(a) USPS (b) protein (c) Connect-4 (d) SensIT

Fig. 6. The performance of ULGE vs. regularization parameter (α).



(a) USPS (b) protein (c) Connect-4 (d) SensIT

Fig. 7. The performance of ULGE vs. # nearest anchors.

representative anchors with a computational complexity of $O(nd\log(m)t)$. Then, the overall computational complexity of ULGE/OULGE is $O(ndm)$, which is a significant improvement compared with conventional methods. Extensive experiments conducted on 7 large scale data sets demonstrate the efficiency and effectiveness of the proposed methods.

APPENDIX A

Proof of Lemma 1. Directly substitute $\hat{W}R$ into problem (2) as

$$\begin{aligned}&Tr(((\hat{W}R)^TX^THDHX\hat{W}R)^{-1}(\hat{W}R)^TX^THLHX\hat{W}R)\\=&Tr(R^{-1}(\hat{W}^TX^THDHX\hat{W})^{-1}\hat{W}^TX^THLHX\hat{W}R)\\=&Tr((\hat{W}^TX^THDHX\hat{W})^{-1}\hat{W}^TX^THLHX\hat{W})\end{aligned} \tag{26}$$

which means $\hat{W}R$ also makes problem (2) achieve optimal value. Thus, we complete the proof. □

Proof of Lemma 2. One can easily check that if F_p is the optimal solution to LE, F_pR is still the optimal solution, the proof is similar to Lemma 1. We then just need to validate that $\hat{W}R$ is the optimal solution to following problem:

$$\min_W \|HXW - F_pR\|_F^2. \tag{27}$$

To see this, note that $\hat{W}$ is the optimal solution to problem (5), we get the derivative of problem (5) as $X^TH(HX\hat{W} - F_p) = 0$, we then write the derivative of problem (27) and substituting $W = \hat{W}R$ as

$$X^TH(HX\hat{W}R - F_pR) = X^TH(HX\hat{W} - F_p)R = 0, \tag{28}$$

therefore, $\hat{W}R$ is the optimal solution to problem (27), which completes the proof. □

Proof of Lemma 3. On the one hand, it is easy to know that

$$\mathrm{Span}(A^{-1}BB^T) \subseteq \mathrm{Span}(A^{-1}B), \tag{29}$$

where $\mathrm{Span}(M)$ denotes the space spanned by the column of the matrix M.

On the other hand, we get

$$\mathrm{rank}(A^{-1}BB^T) = \mathrm{rank}(BB^T) = \mathrm{rank}(B) = \mathrm{rank}(A^{-1}B), \tag{30}$$

where $\mathrm{rank}(M)$ denotes the rank of matrix M.

Combining Eq. (29) and Eq. (30), we know that these two spaces are exact same, thus we complete the proof. □

Proof of Corollary 3. We first rewrite the first matrix as

$$A^{-1}B\Lambda B^T = A^{-1}B\Lambda^{\frac{1}{2}}(B\Lambda^{\frac{1}{2}})^T \tag{31}$$

Thus, based on Lemma 3, we have

$$\mathrm{Rank}\,(A^{-1}B\Lambda^{\frac{1}{2}}(B\Lambda^{\frac{1}{2}})^T) = \mathrm{Rank}\,(A^{-1}B\Lambda^{\frac{1}{2}}) \tag{32}$$

Note that Λ is positive definite, it is easy to know that

$$\mathrm{Rank}\,(A^{-1}B\Lambda^{\frac{1}{2}}) = \mathrm{Rank}\,(A^{-1}B) \tag{33}$$

Combining Eq. (32) and Eq. (33), we complete the proof. □

Proof of Lemma 4. According to Gerschgorin's disk theorem, since A is PSD and doubly stochastic, the maximum eigenvalue of A is 1, and the corresponding eigenvector is $\frac{1}{\sqrt{n}}\mathbf{1}$. Then, we perform eigenvalue decomposition of A as

$$A = \lambda_1 f_1 f_1^T + \lambda_2 f_2 f_2^T + \ldots + \lambda_n f_n f_n^T, \tag{34}$$

where $0 \leq \lambda_i \leq 1$ is the i-th largest eigenvalue and f_i is the corresponding eigenvector. According to Eckart-Young-Mirsky theorem [49], rank-p approximation of A is constructed as

$$\tilde{A}_p = \lambda_1 f_1 f_1^T + \lambda_2 f_2 f_2^T + \ldots + \lambda_p f_p f_p^T \tag{35}$$

Note that A is symmetric, according to the orthogonality of the eigenvector, we get

$$\begin{aligned}\tilde{A}_p\mathbf{1} &= \lambda_1 f_1 f_1^T\mathbf{1} + \lambda_2 f_2 f_2^T\mathbf{1} + \ldots + \lambda_p f_p f_p^T\mathbf{1}\\&= \frac{1}{n}\mathbf{1}\mathbf{1}^T\mathbf{1} + \mathbf{0} + \ldots + \mathbf{0}\\&= \mathbf{1}\end{aligned} \tag{36}$$

Similarly, we can get $\mathbf{1}^T\tilde{A}_p = \mathbf{1}^T$. That is, $\tilde{A}_p$ is obviously still doubly stochastic, which we complete the proof. □

REFERENCES

[1] N. Cristianini, J. Shawe-Taylor, and J. S. Kandola, "Spectral kernel methods for clustering," in *Advances in Neural Information Processing Systems 14*, 2001, pp. 649–655.

[2] F. Nie, X. Wang, and H. Huang, "Clustering and projected clustering with adaptive neighbors," in *The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2014, pp. 977–986.

[3] X. He, D. Cai, and P. Niyogi, "Laplacian score for feature selection," in *Advances in Neural Information Processing Systems 18*, 2005, pp. 507–514.

[4] L. Du and Y. Shen, "Unsupervised feature selection with adaptive structure learning," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Sydney, NSW, Australia, August 10-13, 2015*, 2015, pp. 209–218.

[5] M. Belkin and P. Niyogi, "Laplacian eigenmaps and spectral techniques for embedding and clustering," in *Advances in Neural Information Processing Systems 14*, 2001, pp. 585–591.

[6] D. Cai, X. He, and J. Han, "Spectral regression: a unified subspace learning framework for content-based image retrieval," in *Proceedings of the 15th International Conference on Multimedia*, 2007, pp. 403–412.

[7] X. He and P. Niyogi, "Locality preserving projections," in *Advances in Neural Information Processing Systems 16*, 2003, pp. 153–160.

[8] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[9] X. Wang, H. Ji, C. Shi, B. Wang, Y. Ye, P. Cui, and P. S. Yu, "Heterogeneous graph attention network," in *The World Wide Web Conference*, 2019, pp. 2022–2032.

[10] C. Zhang, D. Song, C. Huang, A. Swami, and N. V. Chawla, "Heterogeneous graph neural network," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019, pp. 793–803.

[11] S. Wang, Z. Chen, D. Li, Z. Li, L.-A. Tang, J. Ni, J. Rhee, H. Chen, and P. S. Yu, "Attentional heterogeneous graph neural network: Application to program reidentification," in *Proceedings of the 2019 SIAM International Conference on Data Mining*. SIAM, 2019, pp. 693–701.

[12] W. Liu, J. He, and S. Chang, "Large graph construction for scalable semi-supervised learning," in *Proceedings of the 27th International Conference on Machine Learning*, 2010, pp. 679–686.

[13] D. Cai and X. Chen, "Large scale spectral clustering via landmark-based sparse representation," *IEEE T. Cybernetics*, vol. 45, no. 8, pp. 1669–1680, 2015.

[14] Y. Li, F. Nie, H. Huang, and J. Huang, "Large-scale multi-view spectral clustering via bipartite graph," in *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015, pp. 2750–2756.

[15] R. Zhang and Z. Lu, "Large scale sparse clustering," in *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*, 2016, pp. 2336–2342.

[16] W. Chen, Y. Song, H. Bai, C. Lin, and E. Y. Chang, "Parallel spectral clustering in distributed systems," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 3, pp. 568–586, 2011.

[17] D. Cai, "Compressed spectral regression for efficient nonlinear dimensionality reduction," in *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015, pp. 3359–3365.

[18] F. Nie, W. Zhu, and X. Li, "Unsupervised large graph embedding," in *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[19] K. Zhang and J. T. Kwok, "Clustered nyström method for large scale manifold learning and dimension reduction," *IEEE Trans. Neural Networks*, vol. 21, no. 10, pp. 1576–1587, 2010.

[20] S. Kumar, M. Mohri, and A. Talwalkar, "Sampling methods for the nyström method," *Journal of Machine Learning Research*, vol. 13, pp. 981–1006, 2012.

[21] E. Kokiopoulou and Y. Saad, "Orthogonal neighborhood preserving projections: A projection-based dimensionality reduction technique," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 12, pp. 2143–2156, 2007.

[22] F. Nie, S. Xiang, Y. Jia, and C. Zhang, "Semi-supervised orthogonal discriminant analysis via label propagation," *Pattern Recognition*, vol. 42, no. 11, pp. 2615–2627, 2009.

[23] R. S. T and S. L. K, "Nonlinear dimensionality reduction by locally linear embedding," *Science*, vol. 290, no. 5, pp. 2323–2326, 2000.

[24] J. B. Tenenbaum, "Mapping a manifold of perceptual observations," in *Advances in Neural Information Processing Systems 10*, 1997, pp. 682–688.

[25] S. Yan, D. Xu, B. Zhang, H. Zhang, Q. Yang, and S. Lin, "Graph embedding and extensions: A general framework for dimensionality reduction," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 1, pp. 40–51, 2007.

[26] F. Nie, D. Xu, I. W. Tsang, and C. Zhang, "Flexible manifold embedding: A framework for semi-supervised and unsupervised dimension reduction," *IEEE Transactions on Image Processing*, vol. 19, no. 7, pp. 1921–1932, 2010.

[27] Y. E. Traboulsi, F. Dornaika, and A. Assoum, *Kernel flexible manifold embedding for pattern classification*. Elsevier Science Publishers B. V., 2015.

[28] W. Hu, J. Gao, J. Xing, C. Zhang, and S. Maybank, "Semi-supervised tensor-based graph embedding learning and its application to visual discriminant tracking," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 1, pp. 172–188, 2017.

[29] X. Shi, Z. Guo, Z. Lai, Y. Yang, Z. Bao, and D. Zhang, "A framework of joint graph embedding and sparse regression for dimensionality reduction," *IEEE Transactions on Image Processing A Publication of the IEEE Signal Processing Society*, vol. 24, no. 4, pp. 1341–55, 2015.

[30] X. B. Shen, Q. S. Sun, and Y. H. Yuan, "A unified multiset canonical correlation analysis framework based on graph embedding for multiple feature extraction," *Neurocomputing*, vol. 148, no. 148, pp. 397–408, 2015.

[31] J. Cui, J. Wen, Z. Li, and B. Li, "Discriminant non-negative graph embedding for face recognition," *Neurocomputing*, vol. 149, pp. 1451–1460, 2015.

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

[33] H. Wang, S. Yan, D. Xu, X. Tang, and T. S. Huang, "Trace ratio vs. ratio trace for dimensionality reduction," in *2007 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2007.

[34] F. Nie, X. Cai, and H. Huang, "Flexible shift-invariant locality and globality preserving projections," in *Machine Learning and Knowledge Discovery in Databases - European Conference*, 2014, pp. 485–500.

[35] F. Nie, S. Xiang, Y. Song, and C. Zhang, "Orthogonal locality minimizing globality maximizing projections for feature extraction," *Optical Engineering*, vol. 48, no. 1, pp. 017 202–017 202, 2009.

[36] C. C. Paige and M. A. Saunders, "LSQR: an algorithm for sparse linear equations and sparse least squares," *ACM TransD-BLP:conf/ijcai/Cai15a. Math. Softw.*, vol. 8, no. 1, pp. 43–71, 1982.

[37] R. Zass and A. Shashua, "Doubly stochastic normalization for spectral clustering," in *Advances in Neural Information Processing Systems*, 2006, pp. 1569–1576.

[38] X. Wang, F. Nie, and H. Huang, "Structured doubly stochastic matrix for graph based clustering: Structured doubly stochastic matrix," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016, pp. 1245–1254.

[39] H. Gifford, "Hierarchical k-means for unsupervised learning."

[40] X. Chang, F. Nie, Z. Ma, and Y. Yang, "Balanced k-means and min-cut clustering," *CoRR*, vol. abs/1411.6235, 2014.

[41] J. He, *Large Scale Nearest Neighbor Search–Theories, Algorithms, and Applications*. Columbia University, 2014.

[42] X. Li, D. Hu, and F. Nie, "Large graph hashing with spectral rotation," in *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[43] A. Y. Ng, M. I. Jordan, and Y. Weiss, "On spectral clustering: Analysis and an algorithm," in *Advances in Neural Information Processing Systems 14*, 2001, pp. 849–856.

[44] D. Zhou, J. Weston, A. Gretton, O. Bousquet, and B. Schölkopf, "Ranking on data manifolds," in *Advances in Neural Information Processing Systems 16*, 2003, pp. 169–176.

[45] X. Zhu, "Semi-supervised learning literature survey," *Computer Science*, vol. 37, no. 1, pp. 63–77, 2008.

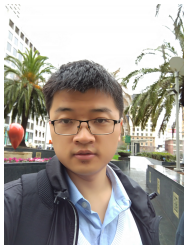
[46] F. Nie, D. Xu, X. Li, and S. Xiang, "Semisupervised dimensionality reduction and classification through virtual label regression," *IEEE Trans. Systems, Man, and Cybernetics, Part B*, vol. 41, no. 3, pp. 675–685, 2011.

[47] F. Schwenker, H. A. Kestler, and G. Palm, "Three learning phases for radial-basis-function networks," *Neural Networks*, vol. 14, no. 4-5, pp. 439–458, 2001.

[48] F. Nie, X. Wang, M. I. Jordan, and H. Huang, "The constrained laplacian rank algorithm for graph-based clustering," in *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA.*, 2016, pp. 1969–1976.

[49] C. Eckart and G. Young, "The approximation of one matrix by another of lower rank," *Psychometrika In Psychometrika*, vol. 1, no. 3, pp. 211–218, 1936.

Feiping Nie received the Ph.D. degree in Computer Science from Tsinghua University, China in 2009, and currently is full professor in Northwestern Polytechnical University, China. His research interests are machine learning and its applications, such as pattern recognition, data mining, computer vision, image processing and information retrieval. He has published more than 100 papers in the following journals and conferences: TPAMI, IJCV, TIP, TNNLS, TKDE, ICML, NIPS, KDD, IJCAI, AAAI, ICCV, CVPR, ACM MM. His papers have been cited more than 17000 times and the H-index is 71. He is now serving as Associate Editor or PC member for several prestigious journals and conferences in the related fields.



Wei Zhu received the masters degree with the School of Computer Science and Center for OPTical IMagery Analysis and Learning (OPTIMAL), Northwestern Polytechnical University, Xian, P. R. China. His research interests include feature extraction and unsupervised learning.

Xuelong Li (M'02 - M'07 - F'12) is a Full Professor with the Center for OPTical IMagery Analysis and Learning (OPTIMAL), State Key Laboratory of Transient Optics and Photonics, Xian Institute of Optics and Precision Mechanics, Chinese Academy of Sciences, Xian, Shaanxi, P. R. China.