BNL-111902-2016-CP

Diverse Power Iteration Embeddings and Its Applications

Huang Hao, Shinjae Yoo, Dantong Yu, Hong Qin

Presented at the 2014 IEEE International Conference on Data Mining

Shenzhen, China
12/14/2016-12/17/2016

December 2016

Computational Science Initiative

Brookhaven National Laboratory

U.S. Department of Energy
[ASCR]

DISCLAIMER

BNL-111902-2016-CP

2014 IEEE International Conference on Data Mining

Diverse Power Iteration Embeddings and Its Applications

Hao Huang*, Shinjae Yoo†, Dantong Yu† and Hong Qin*
*Department of Computer Science, Stony Brook University
Email: haohuangcssbu@gmail.com, qin@cs.stonybrook.edu
†Computational Science Center, Brookhaven National Laboratory
Email: sjyoo@bnl.gov, dtyu@bnl.gov

***Abstract*—Spectral Embedding is one of the most effective dimension reduction algorithms in data mining. However, its computation complexity has to be mitigated in order to apply it for real-world large scale data analysis. Many researches have been focusing on developing approximate spectral embeddings which are more efficient, but meanwhile far less effective. This paper proposes Diverse Power Iteration Embeddings (DPIE), which not only retains the similar efficiency of power iteration methods but also produces a series of diverse and more effective embedding vectors. We test this novel method by applying it to various data mining applications (e.g. clustering, anomaly detection and feature selection) and evaluating their performance improvements. The experimental results show our proposed DPIE is more effective than popular spectral approximation methods, and obtains the similar quality of classic spectral embedding derived from eigen-decompositions. Moreover it is extremely fast on big data applications. For example in terms of clustering result, DPIE achieves as good as 95% of classic spectral clustering on the complex datasets but $4000+$ times faster in limited memory environment.**

I. Introduction

Spectral Embedding is one of the methods to calculate low dimensional embeddings. It was used in clustering [19] [10] at first but later applied to many other data mining applications such as anomaly detection [8] [9] and feature selection [1]. Spectral Embedding uses a spectral decomposition of the graph Laplacian[17]. The generated graph can be considered as a discrete approximation of the low dimensional manifold embedded in the original high dimensional data space. Minimizing a cost function based on the graph ensures neighboring data points that are close to each other on the manifold to be still mapped to neighboring ones in the low dimensional space, i.e. preserving local distances/neighborhood.

Although Spectral Embedding gained an increasing popularity in recent years, its associated high complexity in both time $O(n^3)$ and space $O(n^2)$ prevents it from practical utilization in many real-world applications. For instance, we cannot do spectral clustering directly on popular RCV1 benchmark dataset due to its large data size of nearly $200,000$ documents. Given a dataset with n data points, spectral methods create an $n \times n$ affinity matrix and apply eigen-decomposition on the subsequent Laplacian normalized matrix with the time complexity of $O(n^3)$ in general.

To overcome these limitations, several methods are proposed such as [13] [26] [11]. Among them, Power Iteration Clustering (PIC) [13] is one of the most promising candidates due to its speed, small memory requirements and yet effectiveness in obtaining clustering results for datasets with small number of clusters. However, PIC cannot handle well those datasets with a large number of clusters, even with the new PIC-k (with k power iteration vectors) method [12]. In addition, it is also an impediment to apply this type of power iteration embedding in many other data mining applications, such as feature selection and anomaly detection.

This paper proposes Diverse Power Iteration Embeddings (DPIE) which overcomes the limitations of PIC/PIC-k and applies it in a broad scope of spectral analysis. Moreover, it requires a far less amount of time and space, which is similar to PIC-k. Our contributions in DPIE are as follows:

(1) We proposed a novel power-iteration-based method that aims to find diverse and yet informative low dimensional embeddings, which is different from the single or similar embedding vectors from previous PIC methods.
(2) In theory, our proposed DPIE has the same or similar representational power of low dimensional projection with classic spectral embeddings, so that it can be applicable to various spectral analysis.
(3) Our proposed DPIE, compared with the existing spectral embedding approximations, achieves a similar or even lower time and space computational complexity, but a more desired quality.
(4) We systematically evaluated DPIE along with several closely-related algorithms on a number of important applications. The results confirmed that our new algorithm significantly outperformed other existing algorithms in terms of effectiveness and efficiency.

II. Spectral Embeddings Construction

Spectral embedding construction already gained its popularity in the last decade because of its ability to reveal embedded data structure. It has a strong connection with a graph cut, i.e., it uses eigenspace to solve a relaxed form of a normalized graph partitioning problem [19]. Its second desirable aspect is that it can capture the nonlinear structure of data with the help of nonlinear kernel, which is difficult for k-means or other linear clustering algorithms.

Spectral embedding construction as shown in Algorithm 1, starts with local information encoded in a weighted graph that is constructed from input data with a certain similarity kernel,

1550-4786/14 $31.00 © 2014 IEEE
DOI 10.1109/ICDM.2014.87

Algorithm 1: SpectralEmbeddingConstruction(X, c)

Input: $X \in R^{n\times m}$ where n is #instances and m is #features, and c is #low-dimensions.
Output: Spectral embeddings $Y \in R^{n\times c}$.

1 Construct the affinity matrix $W \in R^{n\times n}$ of X;
2 Compute the diagonal matrix $D \in R^{n\times n}$ where $D(i,i) = \sum_{j=1}^{n} W(i,j)$ and $D(i,j) = 0$ if $i \neq j$;
3 Construct a graph Laplacian L using $L_{nn} = D - W$, $L_{rw} = I - D^{-1}W$ or $L_{sym} = I - D^{-1/2}WD^{-1/2}$;
4 Extract the first c nontrivial eigenvectors Ψ of L, $\Psi = \{\psi_1, \psi_2, \ldots, \psi_c\}$;
5 Re-normalize the rows of $\Psi \in R^{n\times c}$ into $Y_i(j) = \psi_i(j)/(\sum_l \psi_i(l)^2)^{1/2}$;

and selects embedding vectors from the global eigenvectors of the corresponding (normalized) affinity matrix.

Although it demonstrated its effectiveness in clustering [19], feature selection [1], and anomaly detection [8], it is infeasible for large-scale data analysis due to its time and space complexities. The space requirement for constructing affinity matrix (Step 1) is $O(n^2)$, and the computing time for eigen-decomposition in Step 4 is $O(n^3)$. A mechanism is needed to approximate Algorithm 1 with less time and space requirements while retaining similar effectiveness.

III. Power Iteration Embeddings and Its Limitations

A. Power Iteration Embeddings

To address the complexity of classic spectral embedding construction, Lin et.al [13] proposed power iteration clustering (PIC), which finds a one dimensional data embedding using truncated power iteration on a Laplacian normalized affinity matrix. PIC is based on a simple iterative method called power iteration, which we will briefly introduce here.

According to [17], the c smallest eigenvectors of graph Laplacian L_{rw} happen to be the c largest eigenvectors of random walk normalized affinity matrix $W_{rw} = D^{-1}W$. For our notational convenience, we will use W for W_{rw} in the rest of our paper. Let $W \in R^{n\times n}$ and recall that if ψ is an eigenvector for W with eigenvalue λ, then $W\psi = \lambda\psi$. Therefore in general, there is $W^t\psi = \lambda^t\psi$ for any t. This observation is the very foundation of the power iteration method.

Suppose $\Psi = \{\psi_1, \psi_2, \ldots, \psi_n\}$, the set of unit eigenvectors of W, forms a basis in $R^{n\times n}$, and has corresponding real eigenvalues $\Lambda = \{\lambda_1, \lambda_2, \ldots, \lambda_n\}$. We assume that the first c eigenvectors carry informative signals and the rest eigenvectors are noise [17]. From the spectral theorem, for the properly normalized affinity matrix W such as random walk normalization, there are eigenvalues as follows:

$$1 = \lambda_1 > \lambda_2 > \ldots > \lambda_c \gg \lambda_{c+1} > \ldots > \lambda_n. \qquad (1)$$

Note that power iteration embeddings assume 1) there is at least a large enough eigen-gap between c and $c+1$ and 2) $\lambda_2 \sim \lambda_3 \sim \ldots \sim \lambda_c$. Now let $v^{(0)} \in R^n$ be a randomly generated vector, since Ψ is a basis of $R^{n\times n}$, we have:

$$v^{(0)} = a_1\psi_1 + a_2\psi_2 + \ldots + a_n\psi_n, \qquad (2)$$

where a_i is the weight of i-th eigenvector. Then, the power iteration will be:

$$\begin{aligned} v^t = W^t v^{(0)} &= a_1\lambda_1^t\psi_1 + a_2\lambda_2^t\psi_2 + \ldots + a_n\lambda_n^t\psi_n \\ &= a_1\psi_1 + \lambda_2^t\left(\sum_{i=2}^{n} a_i(\frac{\lambda_i}{\lambda_2})^t\psi_2\right). \end{aligned} \qquad (3)$$

The power iteration will finally converge to $a_1\psi_1$ which is useless because it is a constant vector. However, if the number of iteration t is cleverly set from being too large as shown in [13], $W^t v^{(0)}$ is a linear combination of the first c informative eigenvectors, while all the other eigenvectors are gone away due to the eigen-gap. In other word, the whole process should be controlled very well in order to remove the terms of $\psi_{c+1}\ldots\psi_n$ with diminishing rate $(\frac{\lambda_{c+1}}{\lambda_2})^t$, but still keep the rate of $(\frac{\lambda_c}{\lambda_2})^t$ big enough. Fortunately, if the power iteration reaches the eigen-gap, then the convergence rate will be relatively slow because the similar values from λ_2 to λ_c. PIC defines the velocity at t as $\delta^t = |v^t - v^{t-1}|$ and acceleration at t as $\varepsilon = ||\delta^t - \delta^{t-1}||_{max}$ as a measure of the convergence rate and stop power iterations if ε is very small to do early stopping. Figure 1 shows the effect of different number of power iterations and $t = 20$ shows a pretty good clustering embedding. Lin and Cohen [13] proposed the described procedure as Power Iteration Embedding (PIE) algorithm, also shown in Algorithm 2.

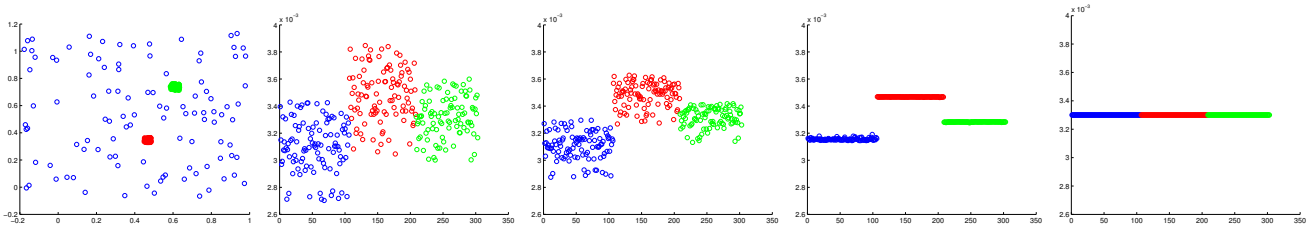
Algorithm 2: PowerIterationEmbedding(X)

Input: $X \in R^{n\times m}$ where n is #instances and m is #features.
Output: Power iteration embedding $v^t \in R^{n\times 1}$.

1 Construct the affinity matrix $W \in R^{n\times n}$ of X;
2 Perform positive random normalization $W \leftarrow D^{-1}W$;
3 Initialize $v^0 \in R^{n\times 1}$;
4 Repeat
5 $\quad v^{t+1} \leftarrow \frac{Wv^t}{\|Wv^t\|_1}$;
6 $\quad \delta^{t+1} \leftarrow |v^{t+1} - v^t|$;
7 $\quad t \leftarrow t + 1$;
8 until $\| \delta^t - \delta^{t+1} \|_{max} \simeq 0$;

B. The Limitations of PIE

Although it showed a pretty good embedding in Figure 1, it is not good enough to handle large c clusters or different spectral applications. If the dataset has a relatively large number of clusters, it is quite difficult to discriminate clusters with a single PIE. The obvious reason is that if c is sufficiently large, the number of required eigenvectors increases. But in PIE, the first few (or even one) nontrivial eigenvectors dominate the whole vector. For instance, Figure 2 showed ten selected clusters from 20Newsgroups (see Section VII-A) violates two PIE assumptions; the biggest eigen-gap is between λ_2 and λ_3 and the second biggest is between λ_3 and λ_4, which also violates similar eigenvalues before c eigenvectors. So, the PIE is quite similar to ψ_2, which is not good enough to distinguish the ten clusters. But the ten eigenvectors together reveal more



(a) 2D dataset (b) $t = 1$ (c) $t = 10$ (d) $t = 20$ (e) $t = 200$

Fig. 1. Single power iteration embedding (the embedding v_*^t provided by [13] or Equation 3) for 2D dataset in Figure 1(a) with three clusters, of which each cluster is represented with a different color. In Figure 1(b), 1(c), 1(d) and 1(e), the value of each component of v_*^t is plotted against its index. We can see that although v_*^t eventually converges to a uniform vector (Figure 1(e) when $t = 200$), the intermediate vectors (eg. v_*^t when $t = 20$) reveal the manifold embedding of the dataset. This example shows that PIE could be an efficient alternative to eigenvectors from traditional eigen-decomposition.

information such as the blue cluster from ψ_3, the pink cluster from ψ_6, etc.

Different random starting vectors v^0 may reveal different degrees of impact on top c eigenvectors due to different a_i in Equation 2. Suppose ψ_k ($k > 2$ and $\lambda_2 > \lambda_k$) is a very informative eigenvector and there happens to be $a_k \gg a_2$. By attentively controlling the number of iteration we may have $a_2\lambda_2^t \simeq a_k\lambda_k^t \gg a_{k+1}\lambda_{k+1}^t$, which means that v^t holds essential information from ψ_k without concealing by the first few ψ_i. So by increasing the number of initial vectors to generate multiple PIE or PIE-k ($k = \lceil log(c) \rceil$ according to [12]), the quality of the generated embedding vectors has potential to improve to a certain degree. For instance, the PIE-k of Figure 2 share the similar general trends with the second eigenvector but it reveals slightly different distributions.

But there is still a crucial and unsolved problem: the first few eigenvectors still overshadow the other less important but indispensable eigenvectors. Under this circumstance, these first few eigenvectors are still dominant in the result vector v^t. We can easily see this from Equation 3 as well : each v_k^t is still dominated by the first few $\psi_1, \psi_2, \ldots$ because of $\lambda_1^t \gg \lambda_2^t \gg \ldots \gg \lambda_n^t$. Therefore, for large c clustering problems or the other spectral applications such as spectral feature selection or anomaly detection, PIE and PIE-k are not practical, which we can also verify in Section VII.

IV. Diverse Power Iteration Embeddings

As analyzed in the last session, the fundamental problem in PIE/PIE-k is the essential influences by the first few eigenvectors in each converged embedding vector. To deal with this problem, we propose Diverse Power Iteration Embeddings (DPIE) $\Psi' = \psi'_1, \psi'_2, \ldots, \psi'_n$. We design DPIE to be a collection of informative and yet divergent embedding vectors where each ψ'_k reveals the corresponding eigenvector ψ_k more considerably than any other eigenvector. To achieve this goal, all the previous eigenvectors $\Psi_{1:k-1} = [\psi_1, \psi_2, \ldots, \psi_{k-1}]$ must be removed from ψ'_k, which is the major difference between our DPIE and PIE/PIE-k.

In our DPIE, the first nontrivial embedding vector ψ'_2 would be quite similar to PIE but the subsequent DPIEs will be different in the sense that we take out all the already-found DPIEs from the current one. Let v_i^0 denotes the i-th starting random seed vector and $v_i^t = W^t v_i^0$, and the power iteration was stopped at t-th iteration, we compute ψ'_k from the normalized linear fitting residue of the already-found $k-1$ DPIEs:

$$\psi'_k = \frac{v_i^t - \Psi'_{1:k-1} f^*}{\| v_i^t - \Psi'_{1:k-1} f^* \|_1}, \tag{4}$$

where $f^* \in R^{(k-1)\times 1}$ is the weight coefficient vector of those already-found DPIEs, and is derived from solving the linear equation $argmin_f = \|v_i^t - \Psi'_{1:k-1} f\|$. In other words, we treat the (unnormalized) ψ'_k as residue or regression error, which is obtained by subtracting the effects of the already-found DPIEs from v_i^t. After normalization ψ'_k becomes the next found DPIE.

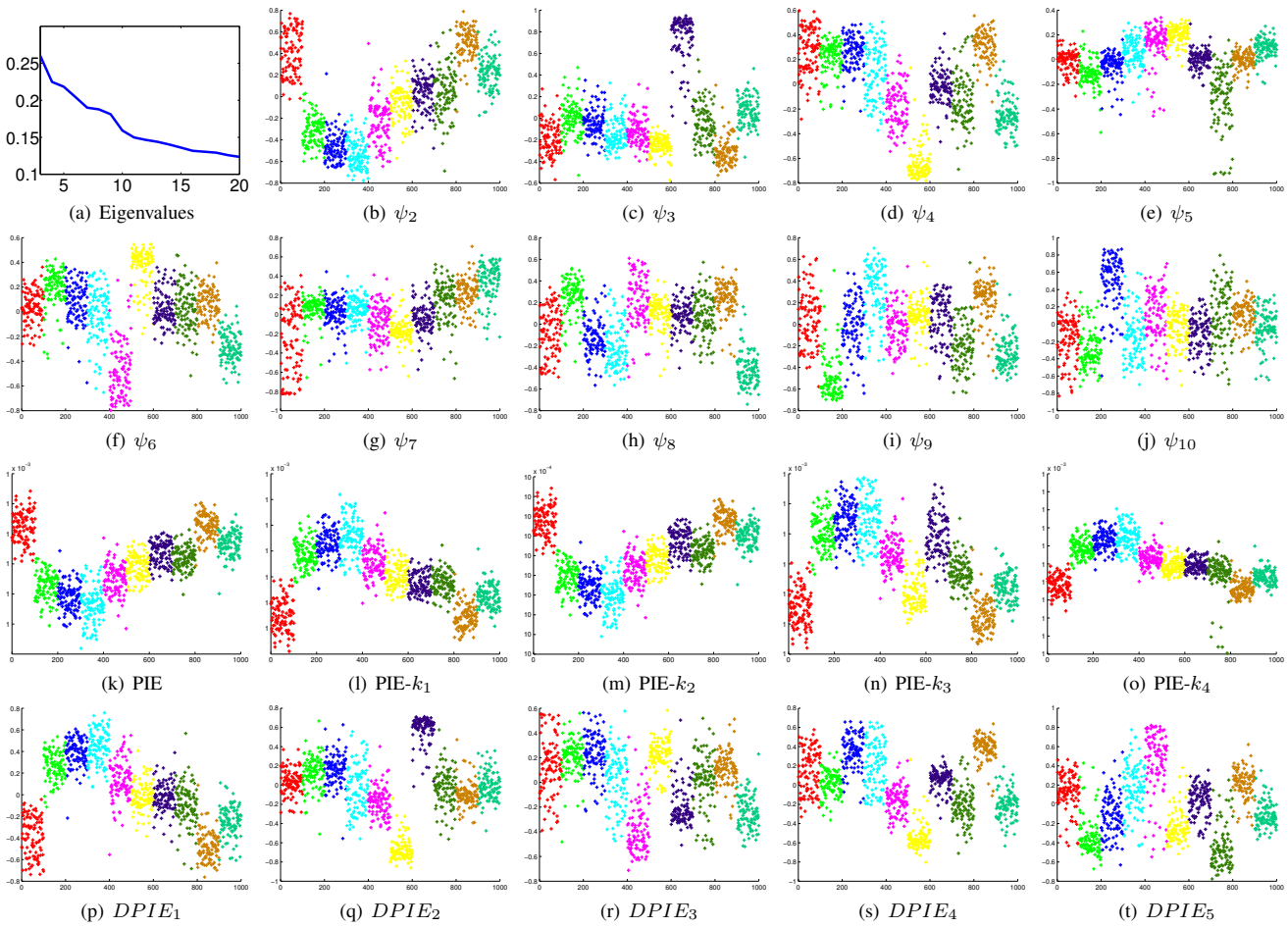
However, if we apply the same stopping criteria as that used

Algorithm 3: DPIE(X, e, E, T, ε_i, η)

Input: $X \in R^{n\times m}$ where n is #instances and m is #features, e is the maximum #DPIE, E is #random seed vectors ($E > e$), T is the maximum #iterations, ε_i defines the acceleration threshold for the i-th random seed, and η is the normalized residual threshold.

Output: Diverse Power iteration embeddings Ψ'.

Construct the affinity matrix of X;
Perform positive random walk normalization on the affinity matrix and denote as W ;
Initialize $v^0 = [v_2^0 \mid v_3^0 \mid \ldots \mid v_E^0] \in R^{n\times E}$, $\Psi' = \{\mathbf{1} \in R^{n\times 1}\}$;
For each v_i^0 $(i = 1, 2, \ldots, E)$
    Repeat
        $v^{t+1} \leftarrow \frac{Wv^t}{\|Wv^t\|_1}$;
        $\delta^{t+1} \leftarrow |v^{t+1} - v^t|$;
        $t \leftarrow t + 1$;
    until ($\| \delta^t - \delta^{t+1} \|_{max} \leq \varepsilon_i$) or ($t \geq T$) ;
    Solve equation $f^* = argmin_f = \|v_i^t - \Psi'_{1:k-1} f\|$;
    $r_i^t \leftarrow v_i^t - \Psi' f^*$;
    If $\frac{\|r_i^t\|_1}{\|v_i^t\|_1} > \eta$
        $\psi'_i \leftarrow \frac{r_i^t}{\|r_i^t\|_1}$;
        Insert ψ'_i into Ψ' ;
        If size of Ψ' equals to e
            Break ;
        End;
    End ;
End ;
Remove $\mathbf{1}$ from Ψ';



(a) Eigenvalues (b) ψ_2 (c) ψ_3 (d) ψ_4 (e) ψ_5

(f) ψ_6 (g) ψ_7 (h) ψ_8 (i) ψ_9 (j) ψ_{10}

(k) PIE (l) PIE-k_1 (m) PIE-k_2 (n) PIE-k_3 (o) PIE-k_4

(p) $DPIE_1$ (q) $DPIE_2$ (r) $DPIE_3$ (s) $DPIE_4$ (t) $DPIE_5$

Fig. 2. Different low dimensional embeddings of 20NG-10 dataset, which consists of 10 cluster subsets from 20Newgroups dataset (Section VII-A). Eigenvectors ψ (Figure 2(b) to 2(j)) are sorted by eigenvalues in descending order (Figure 2(a)). PIE (Figure 2(k)) and PIE-k (Figure 2(l) to 2(o)) are quite similar to ψ_2 in Figure 2(b). Relatively DPIEs (Figure 2(p) to 2(t)) reveal more diverse yet informative signals than PIE and PIE-k.

in PIE or PIE-k, we cannot discover good quality of DPIE. The primary reason is that PIE stopping criteria will suppress the rest of eigenvector signals except the first few because $(\lambda_k/\lambda_2)^t \ll 1$ if t is as large as the PIE stopping criteria. To avoid this problem, we need to increase the acceleration threshold ε of PIE as we find more DPIEs. So, our new stopping criteria for DPIE is as follows:

$$\varepsilon_i = i * \lceil log(c) \rceil * \varepsilon / n, \tag{5}$$

where ε is a tuning parameter and we used 10^{-6} by default as in [13] [14].

When ε is too small; or the random seed is similar to one of what we have used; or v_i^t can be well represented by the existing DPIEs, DPIE cannot find any new PIE. In that case, we check the normalized residual (line 12 in Algorithm 3):

$$\vartheta = \frac{\parallel v_k^t - \Psi'_{1:k-1} f^* \parallel_1}{\parallel v_k^t \parallel_1}. \tag{6}$$

If ϑ is smaller than a certain threshold, we do not add such PIEs. In practice, we used $\lceil log(c) \rceil * \eta / n$ as our threshold and $\eta = 10^{-6}$ by default. For notational convenience, we denote the normalized residual threshold as η from now on.

In terms of stabilities, if ε is too large which means we do very early stopping, then we might not be able to find good eigenvector approximations because PIE is a mixture of interesting and noisy eigenvectors. Relatively, the small ε is not a big problem because the normalized residual threshold η can detect the duplicated information and it is just a little bit slower. However, if ε becomes too small then it will lead to over-convergent. In case of η, it is easy to tune because η has the direct meaning of how much new information is added through the new candidate PIE and it is not relevant to eigengaps of specific dataset. We present the DPIE stability results in regards to ε and η in Experiment Section VII-E.

On the other hand, the power of DPIE can be also interpreted by diffusion theorem. Note that $\Psi_{1:k-1}$ has been removed from ψ'_k, so the explicit formula of ψ'_k is:

$$\psi'_k = b_k \lambda_k^t \psi_k + b_{k+1} \lambda_{k+1}^t \psi_{k+1} + \ldots + b_n \lambda_n^t \psi_n, \tag{7}$$

where b_i is the weight coefficient. Considering the 1-norm distance between x and y on ψ'_k there is:

$$D_k^t(x,y) = |\psi'_k(x) - \psi'_k(y)| = \sum_{i=k}^{n} b_i \lambda_i^t |\psi_i(x) - \psi_i(y)|. \tag{8}$$

It is actually the same as the diffusion process [3], where $\psi'_k(x)$ is the diffusion coordinate of x after t steps/time diffusion process, with all the directions of ψ_i $(i \geq k)$ taken into account. So $D^t_k(x, y)$ is a family of 1-norm diffusion distances between x and y with Markov diffusion process in time t. It reflects the connectivity in the graph of the data: $D^t_k(x, y)$ will be small if there are a large number of short paths connecting x and y, and large enough walking time t. In other words, there is a large transition probability from x to y [3]. In this sense, t plays the role of a scaling parameter. Therefore DPIE has a potential to be more stable to the noise perturbation.

The whole procedure for DPIE is defined in Algorithm 3. Note that 1) we add one vector **1** from line 3 and take it out from the final results to simulate the first eigenvector ψ_1 which is a constant vector and it plays a role of intercept in line 10 in Algorithm 3, and 2) we start v^0 with v^0_2 instead of v^0_1 due to the same reason. We can see the final DPIEs are quite instructive yet different from each other in Figure 2. But like PIE/PIE-k, DPIE is mainly relying on matrix vector multiplication and enjoys the same speed-up and scalability, and it can be easily implemented as distributed matrix vector computation (Section V). Since the most time consuming part (from line 5 to line 9) does not depend on the other DPIE computations, we can further parallelize Algorithm 3.

In the rest of this section, we provide a simple proof of why DPIE can obtain Ψ' (Equation 7), of which each ψ'_k has dominant eigenvector ψ_k while removing the previous eigenvectors $\Psi_{1:k-1}$.

Proposition 1: Assume that t is sufficient large and clear eigengap exists between every two successive eigenvalues, the linear equation solver (Step 10 to 11 in Algorithm 3) can remove the eigencomponents $\Psi_{1:k-1}$ in order to construct DPIE.

Proof: Let us assume the first nontrivial DPIE ψ'_2 is found, and the constant eigencomponent (ψ_1) has been removed from ψ'_2 and v^t_3. We now prove we can get ψ'_3 from v^t_3:

$$\begin{aligned} v^t_3 &= a_2\lambda^t_2\psi_2 + a_3\lambda^t_3\psi_3 + \ldots + a_n\lambda^t_n\psi_n, \\ \psi'_2 &= b_2\lambda^T_2\psi_2 + b_3\lambda^T_3\psi_3 + \ldots + b_n\lambda^T_n\psi_n, \end{aligned} \tag{9}$$

where $T = t + \Delta t$ with $\Delta t \geq 1$ (since we use earlier stopping by controlling ϵ_i when i increases). We assume $argmin_f \|v^t_3 - \psi'_2 \times f\| = f_2$ and all $\lambda_j \leq t/(t + \Delta t)$ with $j \geq 1$, there is:

$$(\frac{1}{\lambda})^{\Delta t} > \frac{1}{\lambda} \geq \frac{t + \Delta t}{t}, \tag{10}$$

therefore:

$$\frac{\lambda^{t-1}}{\lambda^{T-1}} > \frac{T}{t} \Rightarrow \frac{d(\lambda^t - \lambda^T)}{d\lambda} = t\lambda^{t-1} - T\lambda^{T-1} > 0. \tag{11}$$

Since t is sufficiently large, the ratio between a_j and b_j can be ignored. Equation 11 means that $\lambda^t - \lambda^T$ becomes larger when λ is larger. Therefore to minimize the least square $\|v^t_3 - \psi'_2 \times f\|_2$, there should be $f^* = f_2 \sim \lambda^t_2/\lambda^T_2$, which means the first nontrivial eigenvector ψ_2 is removed from the residue:

$$v^t_3 - \psi'_2 \times f_2 = \sum_{j=2}^{n}(\lambda^t_j - \lambda^T_j\frac{\lambda^t_2}{\lambda^T_2})\psi_j = \sum_{j=3}^{n}(\lambda^t_j - \lambda^T_j\frac{\lambda^t_2}{\lambda^T_2})\psi_j, \tag{12}$$

in which ψ_3 is the dominant vector. For all $j \geq 3$, we assume $\lambda_2/\lambda_j \geq (t + \Delta t)/t$, there is:

$$(\frac{\lambda_2}{\lambda_j})^{\Delta t} > \frac{t + \Delta t}{t} \Rightarrow \frac{d(\lambda^t_j - \lambda^T_j\frac{\lambda^t_2}{\lambda^T_2})}{d\lambda_j} > 0. \tag{13}$$

which also leads to the removal of ψ_3 on the following ψ'. Similarly the other eigencomponents can be removed from the coming DPIEs. The above Proposition did not guarantee the eigenvectors if the eigengap is not big between every two successive eigenvalues. However, DPIE procedure guarantees to find diverse PIEs, which are good enough as an approximated eigenvector solution for our proposed applications.

V. Efficient Kernel Computation and Complexity Analysis

DPIE provides a scalable and effective alternative to spectral embedding construction, but it still requires the construction of normalized affinity matrix W (line 1 and 2 in Algorithm 3), which is a huge space cost. This section first describes how to avoid the overhead for storing the affinity matrix by using exact cosine similarity or an approximated Gaussian kernel, and then analyzes the time and space complexity of the whole algorithm.

A. Cosine Similarity

A popular similarity kernel for text dataset is the cosine angle between two vectors, which is defined as:

$$W_{(COS)}(i, j) = \frac{X(i) \cdot X(j)}{\| X(i) \|_2 \cdot \| X(j) \|_2}. \tag{14}$$

X is usually $tf - idf$ weighted sparse matrix and the two norm normalizations in the denominator term enable us to fairly compare documents with different length.

We apply implicit manifold [14] which is represented with a series of sparse matrix multiplications. As described in [14], for the denominator term an additional diagonal matrix $N_{ii} = 1/\sqrt{X(i)X(i)^T}$ is computed and the affinity matrix A and degree matrix D can be calculated with:

$$\begin{aligned} A &= N * X * X^T * N, \\ D &= N * X * X^T * N * \mathbf{1}, \end{aligned} \tag{15}$$

where **1** is a constant vector of all 1's. To remove the diagonal on A, we use a modified equation $D = NXX^TN\mathbf{1} - 1$. Therefore we can represent random walk power iteration as:

$$Wv^t = D^{-1} * (N * (X * (X^T * (N * v^t))) - v^t). \tag{16}$$

Since v^t is a $n \times 1$ vector, and D and N are diagonal matrix which can be stored in a sparse format, Equation 16 is a lot more efficient to implement and at the same time keeps the same output as the conventional implementation. It is also worth to mention that in anomaly detection application we use bi-normalization instead of one-side random walk normalization to make the anomalies more salient:

$$Wv^t = D^{-1} * (N * (X * (X^T * (N * (D^{-1} * v^t)))) - D^{-1} * v^t). \tag{17}$$

TABLE I. NOTATIONS USED IN THE COMPLEXITY ANALYSIS.

	Notations	Meanings
1	n	the number of instances
2	m	the number of features
3	d	the number of samples
4	T	maximum power iterations in DPIE
5	e	maximum number of DPIEs
6	κ	condition number of data eigensystem

B. Gaussian Kernel Approximation

One of the most commonly used similarity measurements is the Gaussian kernel:

$$W_{(GAU)}(i,j) = exp(\frac{- \parallel X(i) - X(j) \parallel^2}{2\sigma^2}), \qquad (18)$$

where σ controls the width of neighborhood [17].

Gaussian kernel is a little bit more complicated than Cosine similarity since it is not a linear construction. In our implementation we approximate it in a space-efficient way by using random Fourier bases [20] [12] shown as follows:

1) Draw d i.i.d. samples $\varpi(1), \ldots, \varpi(d)$ from $p(\varpi \sim \frac{1}{\sigma^2}\mathcal{N}(0,1))$ where $p(*)$ is fast Fourier transform;
2) Draw d i,i,d. samples (offsets) $b(1), \ldots, b(d)$ from uniform distribution on $[0, 2\pi]$;
3) Compute R where $R(i,j) = \sqrt{2/d}[cos(\varpi(j)^T x(i) + b)]$;
4) Use Equation 16 or 17 by replacing X with R.

This approximation can be interpreted as a random projection with Gaussian basis. It projects each point onto a random direction and passes it through a sinusoidal function with σ as bandwidth, and then slides the function by a random amount (offset) [12]. According to the analysis in [20], as the number of samples d increases, the error of this random Fourier bases approximation goes to zero.

C. Analysis of Complexity

Space Complexity. Cosine similarity compresses every intermediate result in a vector form $O(n)$, while the Gaussian kernel approximation is based on sampling matrix of which size is $O(nd)$. Therefore, the space complexity is at most $O(nm)$, which is only as the size of original dataset X, which is much smaller than $O(n^2)$ in general.

Time Complexity. Since a matrix vector multiplication requires $O(nm)$, the process from line 5 to line 9 in Algorithm 3 takes $O(nmT)$, while the operation of solving linear systems takes $O(ne\sqrt{\kappa})$ when using conjugated gradient method ($\kappa = \lambda_1^*/\lambda_2^*$ is the condition number of Ψ' where λ_1^* and λ_2^* are the first and second eigenvalue of Ψ') [22]. Note that these time complexities are much smaller than $O(n^3)$.

VI. DISCUSSION

This section justifies the utility of our proposed DPIE by briefly discussing the theoretical distinctions and connections with a few existing methods, which also lays a solid foundation for DPIE's attractive properties for practical use.

Instance-sampling based Methods. Researches like [27] [2] [21] hold a subset of original instances and extend the clustering result to the whole dataset. Other researches like [4] generate a sparser version of matrix by sampling which can be stored more efficiently and multiplied faster. Alternatively the similarity matrix can also be sampled, which is known as the Nyström method [5]. These methods, although reduce the computation cost, are quite sensitive to the sampling quality [27]. Therefore the embedding quality deteriorates with poor sampling. On the contrary, our proposed DPIE does not rely on any sampling strategy.

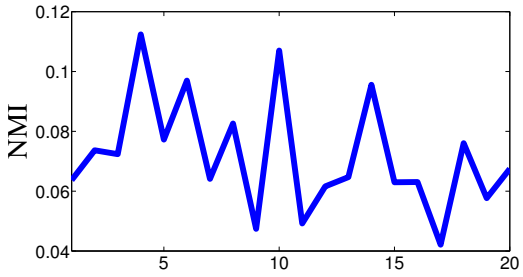


Fig. 3. MatrixSketching [11] clustering results (recorded in NMI) on 20NG-10 dataset, which is a subset of 20Newsgroups with 10 clusters. We ran the algorithm 20 times and every time we shuffled the input order randomly. Obviously the results are NOT stable against different input order, and a lot worse than our DPIE result (NMI = 0.4373).

Random-projection based Methods. Yan et.al. proposed a general framework [28] for fast approximate spectral clustering. It leverages random projection tree to produce a set of reduced representatives and uses them as centroids to cluster all the instances. Gittens et.al. [7] used randomized sketching to approximate the eigenvectors. Their qualities rely on the subspace embedding techniques which result from random projections. However the generated embeddings, because of the indeterministic process, could contain a lot of noisy signals and fail to provide desirable result. In spite of the fact that our DPIE also has random seed vectors as initial status, the seed vectors eventually converge to certain patterns of eigenvector combination during power iteration.

Frequent-direction based Methods. Recent researches drew on the similarity between matrix sketching and the item frequency estimation problems, and proposed frequent-direction based methods [11] [6] with two major contributions: 1) because it is one-pass streaming algorithm, it can be implemented in space and time efficiently, and 2) it approximates the truncated Singular Value Decompositions (SVD). These methods are claimed to be deterministic since they have no sampling or any randomized components. However, their quality is highly related to the input order. For instance, we evaluated the matrix sketching quality of [11] on 20NG-10 dataset 20 times and each time we randomly shuffled the order of input, and performed K-means clustering on the final sketched matrix (evaluated by NMI [24]). Figure 3 shows its poor results and the instability recorded in NMI across the 20 randomly shuffled experiments. On the other hand, our proposed DPIE is constructed with close connections with random walk process. Thereby, DPIE is more stable against perturbation or noisy features.

Power Iteration based Methods. Power iteration clustering [13] computes a linear combination of the important eigenvectors. It is extremely simple and elegant, and efficient in practice and this is why our work shares the same foundation.

Different from the sampling methods and random projection methods, PIC in theory does not modify the original data distribution thus there is no lost information. However the major drawback it suffers is that it tends to return only the first few (or even only one) eigenvectors, which are not enough to represent the datasets with multiple classes or patterns. Although an advanced version, PIE-k, has been proposed later in [12] with multiple output vectors, it does not solve the signal-overlapping problem. Recently deflation-based power iteration method was proposed [26]. It applies Schur complement deflation to remove the previously found pseudo-eigenvectors from the current matrix, so that it computes multiple orthogonal vectors without redundancy. However, strict orthogonality is also a "double-edged sword" since it requires more iterations to extract certain eigenvectors with smaller eigengaps, therefore deflation-based methods take more time to converge compared with PIE-based methods. On the other hand, our DPIE also intends to eliminate the previously found embedding vectors from the next one. But it does not require the embeddings to be orthogonal to each other: each embedding is a different linear combination of eigenvectors. DPIE has similar representation power as real eigenvectors but takes much less iterations than the deflation PIC, resulted in faster computational speed.

VII. Experiments

The low rank embeddings can be used on many data mining applications. We evaluate the quality of the generated embedding vectors through three different application areas: clustering, anomaly detection, and feature selection. For a fair comparison, we constrain each test within a single thread to measure the actual running time. But we want to emphasize that all the algorithms, especially our DPIE, can be implemented and run in a parallel environment.

- **Clustering.** We perform K-means on the generated low-rank embeddings and evaluate the clustering result with NMI (Normalized Mutual Information [24]).
- **Anomaly Detection.** We approximately compute Heat Kernel Signature (HKS) [25] [8] score using the generated low-rank embeddings and evaluate the score with AUC (Area under Receiver Operating Characteristics Curve [18]) which is commonly used to evaluate anomaly detectors and is cut-off independent [16].
- **Feature Selection.** We apply Multi-Cluster Feature Selection (MCFS) [1] with the low-rank embeddings as input to extract feature subset. Although it would be the best to evaluate results based on ground truth of feature importance, it is difficult to find such ground truth. Therefore we evaluate with NMI by applying K-means clustering on the selected feature space.

A. Datasets, Baselines and Parameters.

Datasets. All datasets used in the experiments are summarized in Table II. To demonstrate the quality of the generated embedding on **clustering**, we evaluate our algorithm on three text datasets : 20Newsgroups, Reuters21578 and RCV1, and two image datasets USPS and MNIST. Both of the USPS and MNIST datasets are 10 classes of handwritten digits. Reuters21578 and USPS are unbalanced datasets with quite different size of clusters. For **feature selection** evaluation, we focus on two datasets: 20Newsgroups and Reuters21578. In case of **anomaly detection**, we choose three text datasets and two scientific datasets. 20NG-10-11 is a subset of 20Newsgroups, which consists of all the samples from 6 computer-related clusters (from "comp.graphics" to "comp.windows.x" and treated as regular samples) and 100 randomly-selected samples from "talk.religion.misc" (anomalous samples). Reuters21578AD is a subset of Reuters21578 which is composed of the first two largest categories as regular documents and the smallest 45 categories as anomalous documents. RCV1AD is a subset of RCV1 which is made up of four categories "C15", "ECAT", "GCAT", and "MCAT" and we selected 200 "C15" category documents as anomalies and the rest of three categories as regular documents. Satellite consists of the multi-spectral values of pixels in 3×3 neighborhoods in a satellite image which has unbalanced classification associated with each neighborhood central pixel. Magic04 is a binary classification dataset from the UCI repository which was generated to simulate registration of high energy gamma particles.

TABLE II. Statistics of datasets (including number of instances, features, clusters or anomalies).

	Dataset	# ins.	# fea.	# clu.
1	20Newsgroups	18846	26214	20
2	Reuters21578	8293	18933	65
3	RCV1	193844	47236	103
4	USPS	9298	256	10
5	MNIST	70000	784	10
	Dataset	# ins.	# fea.	# ano.
6	20NG-10-11	4991	26214	100
7	Reuters21578AD	6261	18933	493
8	RCV1AD	7803	29992	200
9	magic04	19020	10	6688
10	satellite	6435	36	2036

For text datasets, cosine similarity (Section V-A) is a reasonable choice. For USPS, MNIST, magic04 and satellite, Gaussian kernel (Section V-B) is used. To adopt an adaptive width of neighborhood σ instead of a fixed value, we assign σ to be the average Euclidean distance of each instance to its second nearest neighbor.

Baselines. For clustering we choose five baselines: NJW (one of the conventional spectral clustering, or Spectral Embedding (SE) when we mention in feature selection) [19], Power Iteration Embedding (PIE) [13], PIE-k [12], Matrix Sketching (MatSket) [11] and DeflationPIC [26]. Once we get the embeddings, we performed a 2-norm normalization along instance side and a WCSS (minimizing within-cluster sum of squares, with 100 inner loops and 100 outer loops) K-means to obtain the cluster assignments.

The anomaly detection experiment is inspired by HKS [8] which is a measure of $X(i)$'s anomalousness using $H_t(i) = \sum_p [e^{\lambda_p t}(\psi_p(i))^2]$ (λ and ψ are derived from positive random walk Laplacian). We name HKS with true eigenvectors as HKS-SE. However, since eigenvalues are not explicitly extracted by PIE, PIE-k, MatSket, DeflationPIC and our proposed DPIE, we use the approximated equation $H'(i) = \sum_p [(v_p(i))^2]$ where v_p is the p-th embedding vector, and call them HKS-PIE, HKS-PIEK, HKS-MatSket, HKS-DFL and HKS-DPIE respectively. To have a more comprehensive comparison, we also include IForest [16] which is a very efficient and effective anomaly detection method. IForest detects data-anomalies with

binary trees, using the property that anomalies are more susceptible to isolation.

The feature selection experiment is integrated with MCFS [1] which measures the importance of each feature along each generated embedding that corresponds to the contribution of each cluster by minimizing $\{min_{s_p}(\| v_p - X s_p \|^2 + \beta \mid s_p \mid)\}$ where s_p is a m-dimensional vector and β controls the s_p's approximation speed to zero. For the j-th feature, MCFS defines the feature importance as $max_p|s_{p,j}|$ where $s_{p,j}$ is the j-th element of vector s_p. We evaluate the output feature subsets by WCSS K-means clustering.

Parameters. Firstly, the number of generated embeddings plays an essential role on the embedding quality. It should be large enough to cover all the signals but small enough to stay away from noise. For clustering and feature selection, we use the first c embeddings from NJW, MatSket and DeflationPIC. PIE generates only one vector while PIE-k set $k = \lceil log(c) \rceil$ [12]. We set the maximum number of DPIEs to be $e = \lceil log(c) \rceil * 6$ out of $E = max(\lceil log(c) \rceil * 30, 2c)$ random seeds. In anomaly detection experiment, for HKS-SE we use all the eigenvectors with eigenvalue-weighted, as the original definition in [25] and [8]. HKS-PIE use only one embedding. For a fair comparison, we compute H' with (the first) 5 output embeddings for HKS-PIEK, HKS-MatSket HKS-DFL and our HKS-DPIE. It is also worth to mention the followings: 1) As the other methods, we use the same normalized affinity matrix as the input in Matrix Sketching to provide manifold insight; 2) For text dataset on IForest, we use l_2-norm normalized X as input to make sure that the result is not sensitive to the document length; and 3) For MCFS in feature selection, we perform 2-norm normalization along sample side of X to evaluate uniform feature scales.

The heat diffusion time variable t in HKS-SE is set to be 1 in order to avoid over-diffusion [8]. In IForest, to conduct a safe and fair comparison, we set the sub-sampling size $\rho = 4000$ and the number of trees $nt = 100$ because these parameters are the authors' recommendation [15].

When we use Gaussian kernel approximation (Section V-B) we set the number of samples $d = 2000$ and $\sigma = 2000$. The maximum number of power iteration T is fixed to be 1000. Acceleration convergence rate in PIE and PIE-k is set to be $\varepsilon = 10^{-5}/n$ where n is the number of samples, as described in [13] and [12]. In our proposed DPIE, we set $\varepsilon_i = i * \lceil log(c) \rceil * \varepsilon/n$ with $\varepsilon = 10^{-6}$, and normalized residual threshold as $\lceil log(c) \rceil * \eta/n$ with $\eta = 10^{-6}$ by default. In Section VII-E we test DPIE stability with different ε and η.

Finally, for each method with sampling steps or random seeds, we run 50 times and report the average performance.

B. Clustering Result Analysis

The clustering results are summarized in Table III. We reported the time used for the affinity matrix and embeddings constructions but we excluded the final K-means steps. For NJW, we also excluded the affinity matrix construction time.

Generally speaking, NJW has the best average performance in NMI since it has full knowledge of the real eigenvectors, but at the same time requires the most expensive cost in time. Compared with PIE, PIE-k is 15 times slower on average since it requires more input and output, but PIE-k improves 20% on average NMI since it has the potential to contain different aspects of signal resulting from different starting vectors. However, it only gets 40% of NJW in NMI. By truncated SVD on normalized affinity matrix, MatSket can deterministically extract the low rank approximation. So it covers additional signals in a more effective way than PIE-k (more than two times better in NMI). But at the same time MatSket is also 1000+ times slower than PIE-k since it requires lots of SVD calculations. DeflationPIC, on the other hand, computes multiple orthogonal pseudo-eigenvectors using deflation technique, so that it could approximate the original eigenvectors to certain degree. It shows improved performance in USPS and MNIST compared with MatSket. But since it requires more matrix computations in the deflation equation, it is noticeably much slower than PIE-k. Our DPIE, although not always the best among all the (approximate) methods, achieves more than 95% performance of NJW in NMI, and at the same time only requires quite a short running time which is close to PIE-k. Especially, DPIE only takes about 2 minutes to process RCV1 dataset but more than 35% better than the second best approximation method with 7 times faster speed.

Due to out-of-memory problem, the NJW experiment on RCV1 could not be finished since it requires full affinity matrix construction. However, using the space-efficient ways introduced in Section V it is not a problem for the other listed methods, especially our proposed DPIE.

C. Anomaly Detection

Table IV shows the anomaly detection results. Similar to the clustering comparisons, HKS-PIEK performed better than HKS-PIE (21% improvement), with the reason that PIE-k is possible to provide more informative signals. HKS-DFL and HKS-MatSket can capture supplementary yet important eigenvectors, which leads to a 6% and 10% boost up respectively compared with HKS-PIEK, but still much worse than HKS-SE (less than 73%). IForest is efficient in that it detects the anomalies by recording the short expected path lengths, so that it has 200% faster running time than HKS-SE and still acquires 86+% performance of HKS-SE. However, our proposed HKS-DPIE is 4220 times faster than HKS-SE and yet reach the best average performance.

D. Feature Selection

We tested all the embedding construction methods using MCFS [1] with {50, 200, 800, 1200, 1800} selected features, and reported the result in Table V. Similar to clustering experiments, DeflationPIC and MatSket perform better than PIE-k and PIE. But DPIE extracts more representative features, which are even with better quality than those derived from original spectral embeddings (SE). This can be explained by the fact that DPIE formulates all the informative signals within diffusion space, which is a more compact and profound way than discrete eigenvectors.

E. Stability Experiments

We conduct experiments with different acceleration threshold ε and normalized residual threshold η to study the parameter tuning sensitivities of DPIE. The results are illustrated

TABLE III. CLUSTERING RESULTS IN NMI AND TIME CONSUMING. FOR EACH DATASET, THE BOLD-FACED NUMBER INDICATES THE BEST APPROXIMATION METHOD (**EXCEPT NJW**), AND THE NUMBERS IN THE PARENTHESES INDICATE THE RANKS OF OUR DPIE. AVERAGE IS THE AVERAGE NMI AND TIME OF EACH METHOD ACROSS ALL THE DATASETS RESPECTIVELY. **We couldn't run NJW on RCV1 dataset due to out-of-memory error, but instead cite its NJW score from [23] for reference.*

NMI	NJW	PIE	PIE-k	MatSket	DeflationPIC	DPIE
20Newsgroups	0.5326	0.2519	0.3266	0.4877	0.4847	**0.5061** (1)
Reuters21578	0.5048	0.2557	0.2718	**0.5322**	0.5014	0.5143 (2)
RCV1	[23]*0.2875*	0.1022	0.1237	0.1521	0.1941	**0.2644** (1)
USPS	0.6207	0.2026	0.2401	0.4667	**0.5871**	0.5786 (2)
MNIST	0.4433	0.0022	0.0028	0.3523	0.3788	**0.4032** (1)
Average	0.4778	0.1629	0.1930	0.3982	0.4292	**0.4533** (1)
Time(s)	NJW	PIE	PIE-k	MatSket	DeflationPIC	DPIE
20Newsgroups	5653.0193	0.1461	5.0816	4131.7741	35.4688	5.0834
Reuters21578	1958.5777	0.0671	2.3548	830.7118	13.7681	1.6388
RCV1	——	5.1961	110.5477	108998.2234	923.6324	127.6903
USPS	1665.3840	0.0675	1.9807	395.9329	7.2451	0.6584
MNIST	201581.2017	4.0707	38.8645	46072.8311	196.3723	43.6582
Average	——	1.9095	31.7659	32085.8947	235.2973	35.7458

TABLE IV. ANOMALY DETECTION RESULTS IN AUC AND TIME CONSUMING. FOR EACH DATASET, THE BOLD-FACED NUMBER INDICATES THE BEST APPROXIMATION METHOD (**EXCEPT HKS-SE**), AND THE NUMBERS IN THE PARENTHESES INDICATE THE RANKS OF OUR HKS-DPIE. AVERAGE IS THE AVERAGE AUC AND TIME OF EACH METHOD ACROSS ALL THE DATASETS RESPECTIVELY.

AUC	HKS-SE	HKS-PIE	HKS-PIEK	HKS-MatSket	HKS-DFL	IForest	HKS-DPIE
20NG-10-11	0.9042	0.3294	0.4858	0.6331	0.2318	0.6176	**0.8844** (1)
Reuters21578AD	0.7845	0.3034	0.5131	0.4824	0.7863	0.6048	**0.9271** (1)
RCV1AD	0.5428	0.4403	0.5049	0.4619	**0.5925**	0.4879	0.5547 (2)
magic04	0.7286	0.5757	0.5757	0.5799	0.4205	**0.7506**	0.7179 (3)
satellite	0.7078	0.3378	0.3378	0.5062	0.5416	0.7173	**0.7193** (1)
Average	0.7336	0.3973	0.4835	0.5327	0.5145	0.6356	**0.7607** (1)
Time(s)	HKS-SE	HKS-PIE	HKS-PIEK	HKS-MatSket	HKS-DFL	IForest	HKS-DPIE
20NG-10-11	876.9247	0.0297	0.8683	181.7283	5.7138	7.6199	0.8193
Reuters21578AD	4141.9718	0.0528	1.1995	170.0181	7.3392	8.2016	1.0608
RCV1AD	4199.1405	0.0476	1.3253	475.9983	10.6519	5.5944	1.1128
magic04	14732.0387	0.1252	0.3402	3241.6766	20.3112	53.8751	2.2759
satellite	779.7334	0.0145	0.1121	152.7320	8.9713	49.3959	0.5889
Average	4945.9618	0.0540	0.7691	844.4307	10.5975	24.9374	1.1715

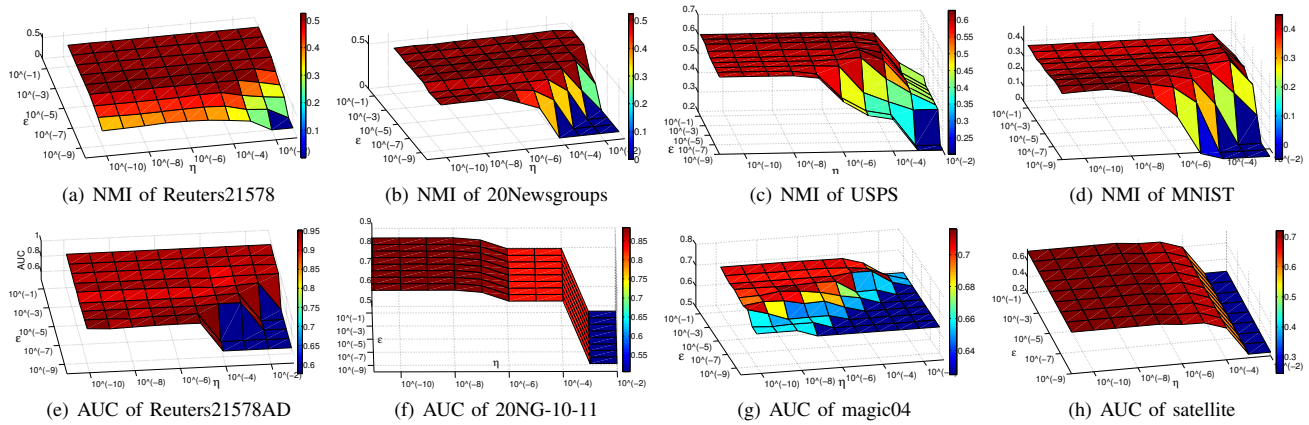
TABLE V. FEATURE SELECTION RESULTS IN NMI. FOR EACH DATASET, THE BOLD-FACED NUMBER INDICATES THE BEST APPROXIMATED METHOD, AND THE NUMBERS IN THE PARENTHESES INDICATE THE RANKS OF OUR DPIE. AVERAGE IS THE AVERAGE NMI OF EACH METHOD. DUE TO SPACE LIMITATION AND THE CLOSE CONNECTIONS BETWEEN CLUSTERING AND FEATURE SELECTION TECHNIQUE WE USED IN THIS PAPER WE DO NOT LIST THE TIME CONSUMING HERE.

20Newsgroups	MCFS-SE	MCFS-PIE	MCFS-PIEK	MCFS-MatSket	MCFS-DFL	MCFS-DPIE
50	0.2971	0.1691	0.1590	0.2691	0.2552	**0.3446** (1)
200	0.3361	0.3089	0.3181	0.3603	0.3274	**0.3834** (1)
800	0.4118	0.3899	0.4115	0.4061	0.4256	**0.4372** (1)
1200	0.4256	0.4696	0.4498	0.4692	0.4335	**0.4819** (1)
1800	0.4865	0.4671	0.4587	0.4340	0.4748	**0.4993** (1)
Reuters21578	MCFS-SE	MCFS-PIE	MCFS-PIEK	MCFS-MatSket	MCFS-DFL	MCFS-DPIE
50	0.3957	0.3959	0.3889	**0.4399**	0.3973	0.4366 (2)
200	0.4607	0.4539	0.4598	0.4745	0.4677	**0.4814** (1)
800	0.5125	0.5021	**0.5183**	0.5113	0.4993	0.5176 (2)
1200	0.5125	0.4783	0.4882	0.4971	0.5122	**0.5297** (1)
1800	0.5081	0.5104	0.5078	0.4980	0.5200	**0.5308** (1)
Average	0.4347	0.4145	0.4160	0.4360	0.4313	**0.4646** (1)

in Figure 4. It indicates that DPIE has a stable range of performance on clustering with large enough ε and small enough η. The reason is that for clustering we need more number of embeddings which cover enough informative eigenvectors. Consequently the iteration should have early stopping controlled by increasing ε to prevent the iteration procedure to remove the less strong eigencomponents, and lowering η to include more diverse DPIEs. Similarly, for anomaly detection DPIE performs stably with large ε and small η. If the anomalies only take a small percentage of total instances, more PIEs are required to separate anomalies from the normal ones. By assigning large enough ε and small enough η, we ensure to obtain enough PIEs while removing the negative influence from the later (noisy) ones.

VIII. CONCLUSION

We proposed a power-iteration-based low dimensional embeddings to cope with the time and space complexities of traditional spectral analysis. Our proposed Diverse Power Iteration Embedding (DPIE), inspired by the power iteration embedding (PIE [13]), can eliminate duplicated information due to a few dominant eigenvectors, which makes it achieve outstanding performance compared with PIE and other related methods [12]. DPIE can be used for not only clustering but also various spectral analysis including feature selection and anomaly detection. Extensive experiments and evaluations on the three spectral analysis applications have demonstrated that our proposed DPIE is the most effective in improving the clustering, anomaly detection, and feature selection methods



(a) NMI of Reuters21578 (b) NMI of 20Newsgroups (c) NMI of USPS (d) NMI of MNIST

(e) AUC of Reuters21578AD (f) AUC of 20NG-10-11 (g) AUC of magic04 (h) AUC of satellite

Fig. 4. Stability experiment with different acceleration threshold ε and normalized residual threshold η.

in the comparison with state-of-the-art baseline approximation algorithms. Meanwhile, DPIE remains efficient in terms of time and space complexity, i.e. being as efficient as PIE-k and much faster than MatrixSketching [11] and DeflationPIC [26].

IX. Acknowledgements

We gratefully thank all the anonymous reviewers for constructive suggestions toward paper improvement. This research is supported in part by National Science Foundation of USA (No. IIS-0949467, IIS-1047715, and IIS-1049448), and National Natural Science Foundation of China (No. 61190120, 61190121, and 61190125). It is also supported by United States Departmentof Energy, Grant No. DE-SC0003361, funded through the American Recovery and Reinvestment Act of 2009, and BSA/DOE Prime Contract (DE-AC02-98CH10886) to Brookhaven National Laboratory.

References

[1] D. Cai, C. Zhang, and X. He. Unsupervised feature selection for multi-cluster data. *SIGKDD*, 2010.

[2] X. Chen and D. Cai. Large scale spectral clustering with landmark-based representation. *AAAI*, 2011.

[3] R. R. Coifman and S. Lafon. Diffusion maps. *Applied and Computational Harmonic Analysis*, 2006.

[4] P. Drineas and A. Zouzias. A note on element-wise matrix sparsification via a matrix-valued bernstein inequality. *Information Processing Letters*, 2011.

[5] C. Fowlkes, S. Belongie, F. Chung, and J. Malik. Spectral grouping using the nyström method. *Pattern Analysis and Machine Intelligence*, 2004.

[6] M. Ghashami and J. M. Phillips. Relative errors for deterministic low-rank matrix approximations. *ACM-SIAM Symposium on Discrete Algorithms*, 2013.

[7] A. Gittens, P. Kambadur, and C. Boutsidis. Approximate spectral clustering via randomized sketching. *Ebay/IBM Research Technical Report*, 2013.

[8] H. Huang, H. Qin, S. Yoo, and D. Yu. Local anomaly descriptor: a robust unsupervised algorithm for anomaly detection based on diffusion space. *CIKM*, pages 405–414, 2012.

[9] H. Huang, H. Qin, S. Yoo, and D. Yu. A new anomaly detection algorithm based on quantum mechanics. *ICDM*, 2012.

[10] H. Huang, S. Yoo, H. Qin, and D. Yu. A robust clustering algorithm based on aggregated heat kernel mapping. *IEEE ICDM*, pages 270–279, 2011.

[11] E. Liberty. Simple and deterministic matrix sketching. *ACM KDD*, 2013.

[12] F. Lin. Scalable methods for graph-based unsupervised and semi-supervised learning. *Doctoral dissertation, Carnegie Mellon University*, 2012.

[13] F. Lin and W. W. Cohen. Power iteration clustering. *ICML*, pages 655–662, 2010.

[14] F. Lin and W. W. Cohen. A very fast method for clustering big text datasets. *ECAI*, 2010.

[15] F. T. Liu and K. M. Ting. Can isolation-based anomaly detectors handle arbitrary multi-modal patterns in data? *Technical Report*, 2010.

[16] F. T. Liu, K. M. Ting, and Z. H. Zhou. Isolation forest. *IEEE ICDM*, pages 413–422, 2008.

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

[18] C. Marzban. A comment on the roc curve and the area under it as performance measures. *Technical Report*, 2004.

[19] A. Ng, M. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. *Advances in Neural Information Processing Systems*, 2002.

[20] A. Rahimi and B. Recht. Random features for large-scale kernel machines. *NIPS*, 2007.

[21] O. Shamir and N. Tishby. Spectral clustering on a budget. *International Conference on Artificial Intelligence and Statistics*, pages 661–669, 2011.

[22] Jonathan Richard Shewchuk. An introduction to the conjugate gradient method without the agonizing pain, 1994.

[23] Y. Song, W. Chen, H. Bai, C. Jin, and E. Y. Chang. Parallel spectral clustering. *Machine Learning and Knowledge Discovery in Databases*, 2008.

[24] A. Strehl and J. Ghosh. Cluster ensembles - a knowledge reuse framework for combining multiple partitions. *J. Mach. Learn. Res.*, pages 583–617, 2003.

[25] J. Sun, M. Ovsjanikov, and L. Guibas. A concise and provably informative multi-scale signature based on heat diffusion. *SGP*, 2009.

[26] N. D. Thang, Y. K. Lee, and S. Lee. Deflation-based power iteration clustering. *Applied intelligence*, 2013.

[27] L. Wang, C. Leckie, K. Ramamohanarao, and J. Bezdek. Approximate spectral clustering. *Advances in Knowledge Discovery and Data Mining*, 2009.

[28] D. Yan, L. Huang, and M. I. Jordan. Fast approximate spectral clustering. *ACM KDD*, 2009.