Diverse Power Iteration Embeddings: Theory and Practice

Hao Huang, Shinjae Yoo, Dantong Yu, and Hong Qin

Abstract—Manifold learning, especially spectral embedding, is known as one of the most effective learning approaches on high dimensional data, but for real-world applications it raises a serious computational burden in constructing spectral embeddings for large datasets. To overcome this computational complexity, we propose a novel efficient embedding construction, Diverse Power Iteration Embedding (DPIE). DPIE shows almost the same effectiveness of spectral embeddings and yet is three order of magnitude faster than spectral embeddings computed from eigen-decomposition. Our DPIE is unique in that 1) it finds linearly independent embeddings and thus shows diverse aspects of dataset; 2) the proposed regularized DPIE is effective if we need many embeddings; 3) we show how to efficiently orthogonalize DPIE if one needs; and 4) Diverse Power Iteration Value (DPIV) provides the importance of each DPIE like an eigen value. Such various aspects of DPIE and DPIV ensure that our algorithm is easy to apply to various applications, and we also show the effectiveness and efficiency of DPIE on clustering, anomaly detection, and feature selection as our case studies.

Index Terms—Approximated spectral analysis, power iteration

1 Introduction

The family of Spectral Embedding algorithms, one of the most popular methods to calculate low dimensional embeddings, has been widely used in diverse application domains such as clustering [18], [32], [34], anomaly detection [15], [16], [17] and feature selection [5], [14], [20], [24], [44]. Spectral Embedding uses a spectral decomposition of the graph Laplacian [32]. The generated matrix can be considered as a discrete approximation of the low dimensional manifold embedded in the original high dimensional data space and such low dimensional embedding reveals the intrinsic relationship among data points and has showed superior effectiveness on machine learning and data mining.

However, the bottleneck in Spectral Embedding algorithm is its associated high complexity in both time $O(n^3)$ and space $O(n^2)$, which prevents it from practical utilization in many real-world applications. For instance, we cannot practically do spectral clustering analysis on popular RCV1 benchmark dataset [22] using a single machine due to its 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 [25], [27], [40]. Among them, Power Iteration Clustering (PIC) [27] is one of the most promising candidates due to its speed, small memory footprint and yet effectiveness in obtaining clustering results for datasets with small number of clusters. However, PIC cannot effectively handle large number of clusters, even with the improved PIC-k (k power iteration vectors with different random starts) method [26]. In addition, it is also an impediment to apply this type of power iteration embedding (PIE) 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, while it requires a far less amount of time and space which is similar to PIC-k. The proposed algorithm has strong theoretical foundation and shows excellent empirical performance. 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 was the main drawback in applying previous PIC methods on various applications.
(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 proposed how to further approximate eigenvalue decomposition by providing Diver Power Iteration Values (DPIV) as the weights for DPIE and efficient orthogonalization of DPIE. Both theoretical proofs and quantitative experiments demonstrate that given certain conditions, DPIVs are good approximation of eigenvalues. Furthermore, we

- *H. Huang is with the GE Global Research, San Ramon, CA 94583. E-mail: haohuanghw@gmail.com.*
- *S. Yoo and D. Yu are with the Brookhaven National Laboratory, Upton, NY 11973. E-mail: shinjae@gmail.com, dtyu@bnl.gov.*
- *H. Qin is with the Computer Science Department, Stony Brook University, Stony Brook, NY 11790. E-mail: qin@cs.stonybrook.edu.*

Manuscript received 27 June 2015; revised 16 Oct. 2015; accepted 1 Nov. 2015. Date of publication 9 Nov. 2015; date of current version 7 Sept. 2016. Recommended for acceptance by S. Yan.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TKDE.2015.2499184

introduced a general procedure to orthogonalize DPIE.

(5) We analyzed the effect of a regularization in choosing diverse PIE, and gave suggestions about how to choose regularization parameters in different situations.

(6) 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.

2 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 because it uses eigenspace to solve a relaxed form of a normalized graph partitioning problem [34]. 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.

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 $L_{sym} = I - D^{-1/2}W\ D^{-1/2}$;
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 $Y_i(j) = \psi_i(j)/(\sum_l \psi_i(l)^2)^{1/2}$;

Spectral embedding construction, as shown in Algorithm 1, starts with local information encoded in a weighted graph that is constructed from the input data with a certain similarity kernel, and selects embedding vectors from the global eigenvectors of the corresponding (normalized) affinity matrix.

Although it demonstrated its effectiveness in clustering [18], [32], [34], anomaly detection [15], [16], [17] and feature selection [5], [14], [20], [24], [44], 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 $O(n^3)$. Therefore for real world big data applications, a mechanism to approximate Algorithm 1 with less time and space complexity is imperative while retaining similar effectiveness.

3 Power Iteration Embeddings and Its Limitations

3.1 Power Iteration Embeddings

To address the complexity of classic spectral embedding construction, Lin and Cohen [27] proposed power iteration clustering, which finds a one dimensional data embedding using truncated power iteration on a Laplacian normalized affinity matrix, and then performs k-means on this one dimensional embedding. PIC is based on a simple iterative method called power iteration, which we will briefly introduce here.

According to [32], 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.

Given $\Psi = \{\psi_1, \psi_2, \ldots, \psi_n\}$, 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 [32]. From the spectral theorem, for the properly normalized affinity matrix W such as random walk normalization, there are eigenvalues as follows:

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

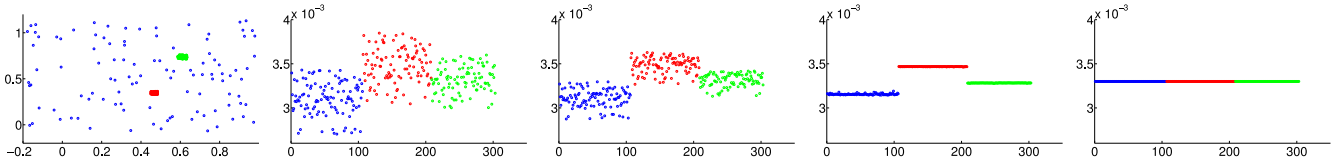
Note that power iteration embeddings assume that 1) there is at least a large enough eigen-gap between c and $c+1$ and 2) $\lambda_2 \sim \lambda_3 \sim \ldots \sim \lambda_c$, where $\sim$ indicates that the two values are close. Now let $v^{(0)} \in R^n$ be a randomly generated vector, since Ψ is a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

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

where a_i is the weight of ith eigenvector. Then, the power iteration will be:

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

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 [27], $W^t v^{(0)}$ would be a linear combination of the first c informative eigenvectors, while all the other eigenvectors are gone away due to the eigen-gap. In other words, 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 stops power iterations if ε is very small for early stopping. Fig. 1 shows the effect of different number of power iterations and $t = 20$ shows a pretty good clustering embedding. Power Iteration Embedding, the key step of PIC proposed by Lin and Cohen [27], is shown in Algorithm 2.



(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 [27] or Equation (3)) for 2D dataset in Fig. 1a with three clusters, of which each cluster is represented with a different color. In Figs. 1b, 1c, 1d and 1e, 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 (Fig. 1e when $t = 200$), the intermediate vectors (e.g., Fig. 1d 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.

Algorithm 2. PowerIterationEmbedding(X)

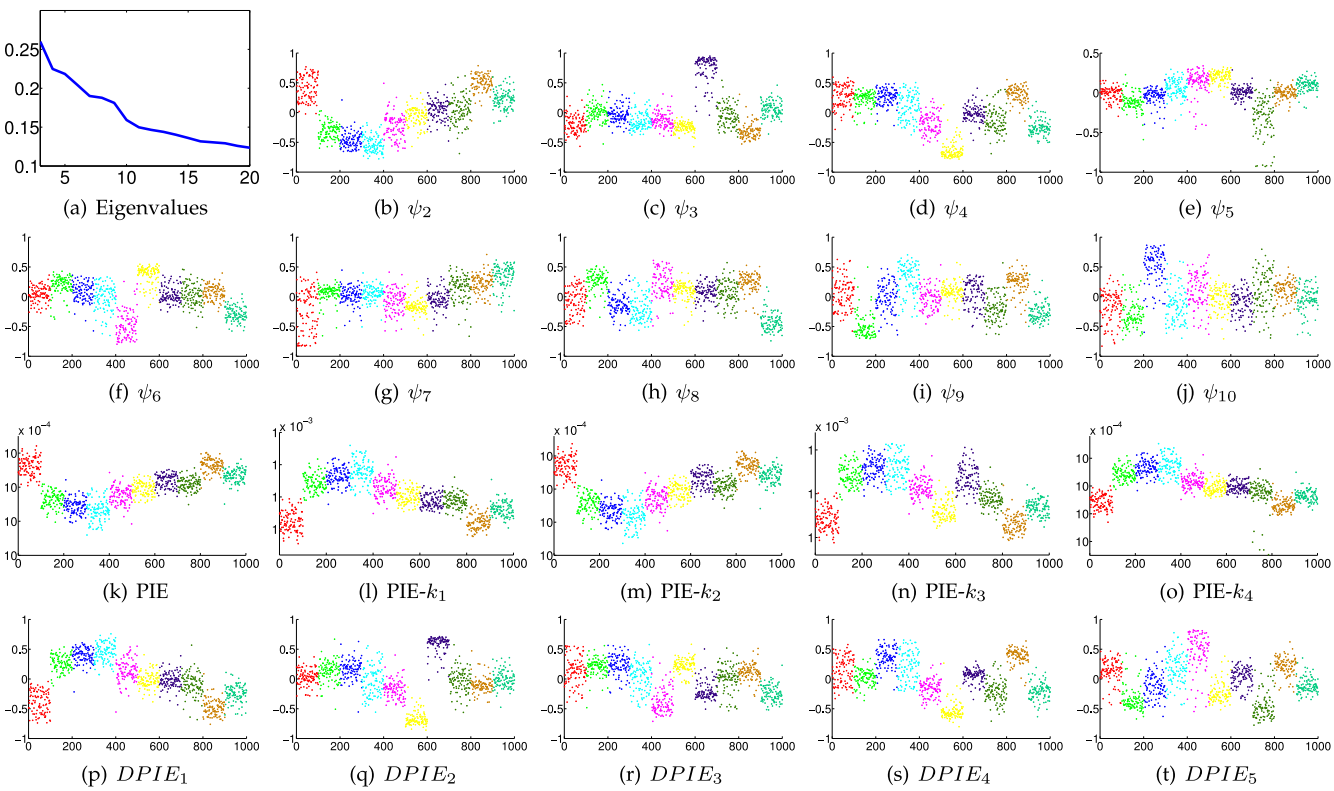
Input: $X \in R^{n \times m}$ where n is #instances and m is #features.
output: Power iteration embedding $v^t \in R^{n \times 1}$.

Construct the affinity matrix $W \in R^{n \times n}$ of X;
Perform positive random normalization $W \leftarrow D^{-1}W$;
Initialize $v^0 \in R^{n \times 1}$;
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 $\| \delta^t - \delta^{t+1} \|_{max} \simeq 0$;

3.2 The Limitations of PIE

Although it showed a pretty good embedding in Fig. 1, it is not good enough to handle a large number of clusters or various spectral applications. If the dataset has a relatively large number of clusters, it is quite difficult to discriminate clusters with only one 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 the first one) non-trivial eigenvectors dominate the whole vector. For instance, Fig. 2 showed ten selected clusters from 20Newsgroups (refer to Section 7) 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. On the contrary, the ten eigenvectors together reveal more information such as the blue cluster from ψ_3, the magenta cluster from ψ_6, etc.

One possible solution for making PIEs more diverse is by having more (different) starting vectors. 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



(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 20Newsgroups dataset (Section 7). Eigenvectors ψ (Figs. 2b, 2c, 2d, 2e, 2f, 2g, 2h, 2i, 2j) are sorted by eigenvalues in descending order (Fig. 2a). PIE (Fig. 2k) and PIE-k (Figs. 2l, 2m, 2n, 2o) are quite similar to ψ_2 in Fig. 2b. Relatively DPIEs (Figs. 2p, 2q, 2r, 2s, 2t) reveal more diverse yet informative signals than PIE and PIE-k.

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 eigenvectors ψ_i. So by increasing the number of initial vectors to generate multiple PIEs or PIE-k ($k = \lceil \log(c) \rceil$ according to [26]), the quality of the generated embedding vectors has potential to improve to a certain degree. For instance, the PIE-k of Fig. 2 share the similar general trends with the second eigenvector but it reveals slightly different distributions.

However, there is still a crucial and unsolved problem: there is no guarantee on the quality of starting vectors, and 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 $\lambda_1^t \gg \lambda_2^t \gg \cdots \gg \lambda_n^t$. 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 can also be verified in Section 7.

4 Diverse Power Iteration Embeddings

4.1 Low Embedding Extraction

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, $\mathbf{\Psi}' = \psi'_1, \psi'_2, \ldots, \psi'_n$. 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 $\mathbf{\Psi}_{1:k-1} = [\psi_1, \psi_2, \ldots, \psi_{k-1}]$ must be removed from ψ'_k, which is the major difference between our DPIE and PIE/PIE-k. The k-th DPIE can be represented as:

$$\psi'_k = b_k\lambda_k^t\psi_k + b_{k+1}\lambda_{k+1}^t\psi_{k+1} + \ldots + b_n\lambda_n^t\psi_n, \tag{4}$$

where b_i is the weight coefficient of the ith eigencomponent.

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 denote the ith starting random seed vector and $v_i^t = W^t v_i^0$, and the power iteration was stopped at tth iteration. We compute ψ'_k from the normalized linear fitting residue of the already-found $k-1$ DPIEs:

$$\psi'_k = \frac{v_i^t - \mathbf{\Psi}'_{1:k-1}f^*}{\| v_i^t - \mathbf{\Psi}'_{1:k-1}f^* \|_1}, \tag{5}$$

where $f^* \in R^{(k-1)\times 1}$ is the weight coefficient vector of those already-found DPIEs, and is derived by solving the following equation:

$$f^* = argmin_f \left(\|v_i^t - \mathbf{\Psi}'_{1:k-1}f\|_2^2 + \alpha \sum_j |f_j|^p \right), \tag{6}$$

where α is a penalty parameter that controls the degree of shrinkage of the coefficient estimates with p-norm (p usually is set as 1 or 2, and the choice of regularization type will be further analyzed in Section 4.3). 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 in PIE or PIE-k, we cannot discover good DPIEs. 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:

$$\varepsilon_i = i\lceil \log(c) \rceil \varepsilon / n, \tag{7}$$

where ε is a tuning parameter and we used 10^{-6} by default as in [27], [28].

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 ith random seed, and η is the normalized residual threshold.

output: Diverse Power iteration embeddings $\mathbf{\Psi}'$.

1 Construct the affinity matrix of X;
2 Perform positive random walk normalization on the affinity matrix and denote as W;
3 Initialize $v^0 = [v_2^0 \,|\, v_3^0 \,|\, \ldots \,|\, v_E^0] \in R^{n\times E}$, $\mathbf{\Psi}' = \{\mathbf{1} \in R^{n\times 1}\}$;
4 For each v_i^0 ($i = 1, 2, \ldots, E$)
5 Repeat
6 $v^{t+1} \leftarrow \frac{Wv^t}{\|Wv^t\|_1}$;
7 $\delta^{t+1} \leftarrow |v^{t+1} - v^t|$;
8 $t \leftarrow t + 1$;
9 until ($\| \delta^t - \delta^{t+1} \|_{max} \leq \varepsilon_i$) or ($t \geq T$);
10 Solve equation $f^* = argmin_f(\|v_i^t - \mathbf{\Psi}'_{1:k-1}f\|_2^2 + \alpha\sum_j |f_j|^p)$;
11 $r_i^t \leftarrow v_i^t - \mathbf{\Psi}' f^*$;
12 If $\frac{\|r_i^t\|_1}{\|v_i^t\|_1} > \eta$
13 $\psi'_i \leftarrow \frac{r_i^t}{\|r_i^t\|_1}$;
14 Insert ψ'_i into $\mathbf{\Psi}'$;
15 If size of $\mathbf{\Psi}'$ equals to e
16 Break;
17 End;
18 End;
19 End;
20 Remove $\mathbf{1}$ from $\mathbf{\Psi}'$;

When ε is too small or the random seed is similar to one of what we have already used, it leads to the fact that v_i^t can be well represented by the existing DPIEs, and therefore the new DPIE contains no new signal. To avoid this case, we check the value of normalized residual (line 12 in Algorithm 3):

$$\vartheta = \frac{\| v_k^t - \mathbf{\Psi}'_{1:k-1}f^* \|_1}{\| v_k^t \|_1}. \tag{8}$$

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 eigen-gaps of specific dataset. We present the DPIE stability results in regards to ε and η in Experiment Section 7.4.

4.1.1 Connection to Diffusion Theory

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 (Equation (4)) is:

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

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

$$D_k^t(x,y) = |\psi'_k(x) - \psi'_k(y)| = \sum_{i=k}^{n} b_i \lambda_i^t |\psi_i(x) - \psi_i(y)|. \quad (9)$$

It is actually the same as the diffusion process [7], 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 $D_k^t(x,y)$ 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: $D_k^t(x,y)$ will be small if there are a large number of short paths connecting x and y, with large enough walking time t. In other words, there is a large transition probability from x to y [7]. In this sense, t plays the role of a scaling parameter to control the diffusion process. Therefore DPIE has a potential to be more stable to noise perturbation.

4.1.2 Algorithm Details

The whole procedure for DPIE is defined in Algorithm 3. Note that 1) we add one vector $\mathbf{1}$ 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_2^0 instead of v_1^0 due to the same reason. In the example shown in Fig. 2, we can see the final DPIEs are quite instructive yet different from each other, and it illustrates the outperformance of DPIEs compared with PIE/PIE-k. But like PIE/PIE-k, DPIE mainly relies on matrix-vector multiplications and enjoys the same speed-up and scalability, and it can be easily implemented as distributed matrix-vector multiplications (Section 5). 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.

4.1.3 Justification of DPIE

In the rest of this section, we provide a justification for DPIE that can obtain Ψ' (Equation (4)), of which each ψ'_k has dominant eigenvector ψ_k while removing the previous eigenvectors $\Psi_{1:k-1}$.

Claim 1: Assume that 1) t is sufficient large; and 2) there exists clear eigengap between every two successive eigenvalues. Given the first nontrivial DPIE ψ'_2 is found, the regression solver (Step 10 to 11 in Algorithm 3) can remove the eigenvectors $\Psi_{1:2}$ while constructing a DPIE, ψ'_3.

Justification: For notational convenience, we assume the constant eigencomponent (ψ_1) has been removed from r_2^T, ψ'_2 and v_3^t. Note that all the eigenvalues except λ_1 are less than 1 and non-negative as shown in Equation (1). We now justify that we can get ψ'_3 from v_3^t:

$$\begin{aligned} v_3^t &= a_2 \lambda_2^t \psi_2 + a_3 \lambda_3^t \psi_3 + \ldots + a_n \lambda_n^t \psi_n, \\ r_2^T &= b_2 \lambda_2^T \psi_2 + b_3 \lambda_3^T \psi_3 + \ldots + b_n \lambda_n^T \psi_n, \\ \psi'_2 &= r_2^T / \|r_2^T\|_1, \end{aligned} \quad (10)$$

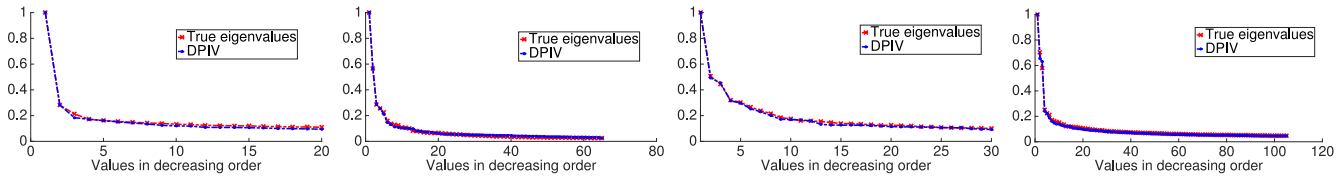
where $T = t + \Delta t$ with $\Delta t \geq 0$ (because we do earlier stopping by controlling ϵ_i when i increases) and r_i^t is the residual vector (Step 11 in Algorithm 3). Since 1) t is sufficiently large, 2) there exists clear eigen gap between any two successive eigenvalues, 3) $\lambda_j^t \ll 1$ and 4) Ψ forms orthonormal basis, we get $f^* = f_2 \sim a_2 \lambda_2^t \|r_2^T\|_1 / (b_2 \lambda_2^T)$ for $f^* = argmin_f \|v_3^t - \psi'_2 f\|_2^2 + \alpha \sum_j |f_j|^p$. In other words, the first nontrivial eigenvector ψ_2 is removed from the residue r_3^t:

$$\begin{aligned} r_3^t = v_3^t - \psi'_2 f_2 &= \sum_{j=2}^{n} \left(a_j \lambda_j^t - b_j \lambda_j^T \frac{a_2 \lambda_2^t}{b_2 \lambda_2^T} \right) \psi_j \\ &= \sum_{j=3}^{n} \left(a_j \lambda_j^t - b_j \lambda_j^T \frac{a_2 \lambda_2^t}{b_2 \lambda_2^T} \right) \psi_j \\ &= \sum_{j=3}^{n} \lambda_j^t \left(a_j - b_j \lambda_j^{\Delta t} \kappa_2 \right) \psi_j, \end{aligned} \quad (11)$$

where $\kappa_2 = \frac{a_2 \lambda_2^t}{b_2 \lambda_2^T}$. Unless $a_3 - b_3 \lambda_3^{\Delta t} \kappa_2$ is close to zero, ψ_3 becomes the dominant factor for ψ'_3 because of $\lambda_3^t \gg \lambda_j^t$ for large enough t and $j > 3$ and thus similarly the other eigencomponents can be removed from the coming DPIEs.

4.1.4 Discussion

In practice, DPIE procedure does not guarantee to find eigenvectors because 1) the eigengap is not always big enough between two successive eigenvalues, 2) t may be small (especially for finding large number of DPIEs), or 3) there is a randomization effect on a_i and b_i. However, our proposed DPIE procedure guarantees to find linearly independent PIEs, which are good enough as an approximated eigenvector solution for our proposed and other candidate applications.



(a) 20Newsgroups (b) Reuters21578 (c) RVC1 (30 largest cluster subset) (d) Sector-scale

Fig. 3. Comparison between DPIV and the true eigenvalues (sorted in decreasing order). DPIV can approximate the true eigenvalues very closely.

4.2 Diverse Power Iteration Values

In this section, we propose a set of special scalar, called Diverse Power Iteration Values $\Lambda' = \{\lambda'_1, \lambda'_2, \ldots, \lambda'_n\}$ associated with each DPIE respectively. They play a similar role as eigenvalues in classic eigensystem. We define λ'_i through the following equation:

$$W\psi'_i = \lambda'_i \psi'_i, \tag{12}$$

so that the λ'_i can be calculated by

$$\lambda'_i = (W\psi'_i)\psi'^{-1}_i, \tag{13}$$

where ψ'^{-1}_i can be calculated by Moore-Penrose pseudoinverse.

We now prove that λ'_i can approximate the real eigenvalue λ_i given certain conditions.

Proposition 1. *Assume that t is sufficient large, clear eigengap exists between every two successive eigenvalues and ψ_i is removed from ψ'_{i+1}, λ'_i is an approximation of the true eigenvalue λ_i to certain degree.*

Proof. Start from the following equation:

$$\begin{aligned}\lambda'_i\psi'_i &= \lambda'_i(b_i\lambda_i^t\psi_i + b_{i+1}\lambda_{i+1}^t\psi_{i+1} + \ldots + b_n\lambda_n^t\psi_n) \\ &= \lambda'_i\lambda_i^t\left(b_i\psi_i + \sum_{j=i+1}^{n} b_j\left(\frac{\lambda_j}{\lambda_i}\right)^t\psi_j\right).\end{aligned} \tag{14}$$

On the other hand, we also have:

$$W\psi'_i = \lambda_i^{t+1}\left(b_i\psi_i + \textstyle\sum_{j=i+1}^{n} b_j\left(\frac{\lambda_j}{\lambda_i}\right)^{t+1}\psi_j\right). \tag{15}$$

If t is sufficient large and clear eigengap exists between λ_i and λ_{i+1}, we have

$$W\psi'_i = \lambda'_i\psi'_i \Rightarrow \lambda'_i\lambda_i^t \approx \lambda_i^{t+1} \Rightarrow \lambda'_i \approx \lambda_i. \tag{16}$$

□

Now we show some experimental result of how close DPIV can approximate the true eigenvalues. We plot DPIV and true eigenvalues of four text datasets, 20Newsgroups, Reuters21578, Sector-Scale and RCV1 (The description of these datasets can be found in Section 7. Here we only use a subset of RCV1 which contains the 30 largest clusters, since the eigendecomposition on the whole RCV1 dataset has out-of-memory issue). The results are shown in Fig. 3 where λ'_i (DPIV) and λ_i (eigenvalues) are sorted in decreasing order.

The comparisons here are to give performance evidences that our proposed DPIV can approximate the true eigenvalues very closely. However, since not all the compared baselines in the experimental Section 7 have eigenvalue approximations (and usually only the low embedding approximations are used in practise), we didn't include DPIV in the experiment settings in Section 7.

4.3 The Choice of Regression Types

One practical issue in solving DPIE is to address the regularization type in the following Equation (copy from Equation (6)):

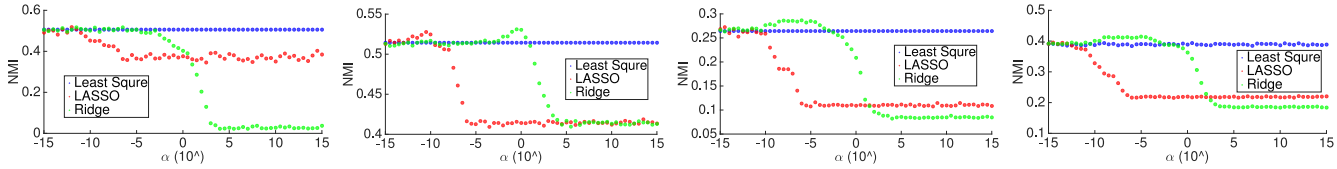
$$f^* = argmin_f = \|v_i^t - \Psi'_{1:k-1}f\| + \alpha\sum_j |f_j|^p.$$

We analyze three different regression types, namely the ordinary least squares regression ($\alpha = 0$), the lasso ($\alpha > 0, p = 1$) and ridge regression ($\alpha > 0, p = 2$).

For lasso and ridge regression, the parameter α can be treated as a penalizing cost or complexity parameter. As α increases, less and less effective terms or variables are likely to be excluded (lasso) or decreased (ridge). Therefore, the value of α should be chosen adaptively, in order to minimize an estimate of the expected prediction error. Lot of prior research have focused on deciding the best regularization parameter α but in practice, on supervised learning, cross validation is usually adopted to find the optimal α. However in unsupervised learning setting, there is no consensus method that provides a universally optimum choice.

In this paper, we use an unsupervised way to estimate a reasonable default value for α. Recent research such as [3] and [13] suggested that α should be set as $b\sqrt{n \times \log(k-1)}$, where n is the number of samples and $k-1$ is the number of existing DPIEs, and b is a scaling parameter. We evaluate the performance of lasso (in red) and ridge (in green) of Equation (6) on clustering task using α with the range of b in $\{-15, -14.5, -14, \ldots, 14, 14.5, 15\}$. The result is shown in Fig. 4 and we also include least square ($\alpha = 0$) of which performance is illustrated as a straight blue line.

From Fig. 4, we can observe the following: 1) Both lasso and ridge have very similar performance with least square when α is very small, which corresponds to the fact that when the regularization or penalty terms are close to zero, regression result is close to least square. 2) Lasso with larger α shows worse performance because lasso does variable selection, resulted in excluding already discovered DPIEs. So we may fail to discover new diverse PIE and it is really bad for the smaller number of DPIEs. 3) With the large number of required DPIEs, ridge regression slightly outperforms least square with certain value of α. When there are large number of DPIEs, there could be an overfitting issue and ridge regression suppresses such issue without excluding variables. In the Experimental Section 7, we will further



(a) 20Newsgroups ($\#clu. = 20$) (b) Reuters21578 ($\#clu. = 65$) (c) RVC1 ($\#clu. = 103$) (d) Sector-scale ($\#clu. = 105$)

Fig. 4. Comparison between lasso (in red) and ridge (in green) across different value of α. When the number of required DPIE (set as the number of clusters) in the dataset are small (such as 20Newsgroups and Reuters21578), Least Square (in blue) can give desirable performance. But with larger number of required DPIE (say $\#DPIE$ or $\#clu. > 80$, such as (RVC1 and Sector-scale)), ridge regression with 10^{-7} is a reasonable choice.

compare the performance between least square and ridge regression in the construction of DPIE.

4.4 Orthogonalizing DPIE

Note that the DPIE obtained in Algorithm 3 are not orthogonal. In this section, we introduce an efficient orthogonalization procedure proposed in [23], which can transform any decomposition of the form $G = \Psi'\Lambda'\Psi'^T$ to $G = \hat{\Psi}\hat{\Lambda}\hat{\Psi}^T$, such that $\hat{\Psi}$ is orthogonal (i.e., $\hat{\Psi}^T\hat{\Psi} = I$). The procedure is general in that it does not require Ψ' or Λ' to be positive semi-definite.

The orthogonalization procedure is shown in Algorithm 4. It can be easily seen that the time complexity of Algorithm 4 is $O(nk^2)$ and its space complexity is $O(nk)$. Thus, this orthogonalization procedure does not increase the complexity of the proposed Algorithm 3. Further complexity analysis of the whole DPIE algorithm is given in Section 5.3.

We now give the theoretical support for Algorithm 4.

Algorithm 4. DPIE-Orthogonalization(Ψ', Λ')

Input: $\Psi' \in R^{n\times k}$ where n is #instances and k is #DPIE, $\Lambda' \in R^{k\times k}$ is the diagonal matrix of DPIV.
output: Orthogonal DPIE $\hat{\Psi} \in R^{n\times k}$ and the corresponding DPIV $\hat{\Lambda} \in R^{k\times k}$.

1 $P \leftarrow \Psi'^T\Psi'$;
2 Perform eigen-decomposition: $P = V\Sigma V^T$;
3 $B \leftarrow \Sigma^{1/2}V^T\Lambda'V\Sigma^{1/2}$;
4 Perform eigen-decomposition: $B = V'\hat{\Lambda}V'^T$;
5 $\hat{\Psi} \leftarrow \Psi'V\Sigma^{-1/2}V'$;

Proposition 2 (Restated from [23]). *The equation* $\Psi'\Lambda'\Psi'^T = \hat{\Psi}\hat{\Lambda}\hat{\Psi}^T$ *holds.*

Proof. We have the following derivation:

$$\begin{aligned}\Psi'\Lambda'\Psi'^T &= (\Psi'V\Sigma^{-1/2})(\Sigma^{1/2}V^T\Lambda'V\Sigma^{1/2})(\Psi'V\Sigma^{-1/2})^T \\ &= (\Psi'V\Sigma^{-1/2})(V'\hat{\Lambda}V'^T)(\Psi'V\Sigma^{-1/2})^T \\ &= \hat{\Psi}\hat{\Lambda}\hat{\Psi}^T.\end{aligned} \tag{17}$$

□

Proposition 3 (Restated from [23]). *The output* $\hat{\Psi}$ *of Algorithm 4 are orthogonal.*

Proof. We have the following derivation:

$$\begin{aligned}\hat{\Psi}\hat{\Psi}^T &= V'^T\Sigma^{-1/2}V^T(\Psi'^T\Psi')V\Sigma^{-1/2}V' \\ &= V'^T\Sigma^{-1/2}V^T(V\Sigma V^T)V\Sigma^{-1/2}V' \\ &= V'^TV' = I.\end{aligned} \tag{18}$$

□

5 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.

5.1 Cosine Kernel

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

$$W_{(COS)}(i,j) = \frac{X(i)\cdot X(j)}{\parallel X(i) \parallel_2 \cdot \parallel X(j) \parallel_2}. \tag{19}$$

X is usually $tf\text{-}idf$ 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 [28] which is represented with a series of sparse matrix multiplications. As described in [28], 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:

$$\begin{aligned}A &= N(X(X^TN)), \\ D &= N(X(X^T(N\mathbf{1}))),\end{aligned} \tag{20}$$

where $\mathbf{1}$ is a constant vector of all 1's. The parentheses here and hereafter in this section emphasize the calculation order, thereby keeping low space complexity. To remove the diagonal on A, we use a modified equation $D = N(X(X^T(N\mathbf{1}))) - 1$. Therefore we can represent random walk power iteration as:

$$Wv^t = D^{-1}(N(X(X^T(Nv^t))) - v^t). \tag{21}$$

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 (21) 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:

$$Wv^t = D^{-1}(N(X(X^T(N(D^{-1}v^t)))) - D^{-1}v^t). \tag{22}$$

TABLE 1
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

5.2 Gaussian Kernel Approximation

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

$$W_{(GAU)}(i,j) = exp\left(\frac{- \parallel X(i) - X(j) \parallel^2}{2\sigma^2}\right), \tag{23}$$

where σ controls the width of neighborhood [32].

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 [35], [26] 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 (21) or (22) 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) [26]. According to the analysis in [35], as the number of samples d increases, the error of this random Fourier bases approximation goes to zero.

5.3 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') [37]. In practise, there is usually $mT > e\sqrt{\kappa}$ and $T \sim k$, therefore the time complexity of DPIE is almost $O(nmk)$.

6 Discussion and Connections to the Existing Methods

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.

6.1 Sampling Based Methods

Research like [6], [36], [41] first analyze a subset of original instances and later extend the analytic result to the whole dataset. Other research like [9] generate a sparse 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 [10], [23]. These methods, although reduce the computation cost, are quite sensitive to the sampling quality [41]. Therefore the embedding quality deteriorates with poor sampling (it will be shown later in Experiment Section 7). On the contrary, our proposed DPIE does not rely on any sampling strategy on the instance side.

6.2 Random-Projection Based Methods

Yan et al. proposed a general framework [43] 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. Khoa and Chawla [21] bypass the eigen-decomposition by using random projection and near-linear time solver. Gittens et al. [12] used randomized sketching to approximate the eigenvectors. Their qualities rely on the subspace embedding qualities 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 stable and desirable result (it will be shown later in Experiment Section 7). 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.

6.3 Frequent-Direction Based Methods

Recent research drew on the similarity between matrix sketching and the item frequency estimation problems, and proposed frequent-direction based methods [11], [25] with two major contributions: 1) because they are one-pass streaming algorithms, they can be implemented in space and time efficiently, and 2) they approximate the truncated Singular Value Decompositions (SVD) in batch setting. 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 [25] 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 [39]). Fig. 5 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 random walk process. Thereby, DPIE is more stable against perturbation or noisy features. We will further compare their performance in Experiment Section 7.

6.4 Power Iteration Based Methods

Power iteration clustering [27] 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

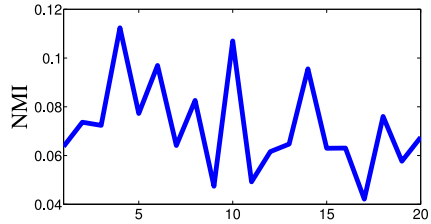


Fig. 5. MatrixSketching [25] clustering results (recorded in NMI) on 20NG-10 dataset, which is a subset of 20Newsgroups with 10 clusters. We ran the algorithm 20 times and every time we shuffled the input order randomly. Obviously the results are NOT stable against different input order, and a lot worse than our DPIE result (NMI = 0.4373).

in theory does not modify the original data distribution thus there is no lost information (with cosine kernel). 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 [26] with multiple output vectors, it does not guarantee to solve the signal-overlapping problem. Comparably, our proposed DPIE aim to diminish the effect of the previously found signals, thereby boosting up the effect of the later ones.

6.5 Deflation Based Methods

Recently deflation based method was proposed [40]. 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, in the process of deflation method, the orthogonality requires more iterations to extract certain eigenvectors with smaller eigengaps, therefore deflation-based methods take more time to converge. On the other hand, our DPIE also intends to eliminate the previously found embedding vectors from the next one. But Algorithm 3 does not require the embeddings to be orthogonal to each other: each embedding is a different linear combination of eigenvectors. DPIE (from Algorithm 3) has similar representation power as real eigenvectors but takes much less iterations than the deflation PIC, resulted in faster computational speed. However, please note that we can always use Algorithm 4 to orthogonalize DPIE. We will further compare the performance against [40] in Experiment Section 7.

6.6 Fast Clustering Methods on Graphs

For a different input setting (therefore not included in the experiment section), fast spectral clustering algorithms on graphs [4], [31], [42] have been proposed, where typically the input of algorithm is a graph of all the nodes (n^2 matrix). While these approaches work well on network datasets such as blogs, it is hard to maintain the whole input with limited memory when the number of samples is large. On the contrary, our proposed DPIE only requires $O(nm)$ complexity which is usually more space-efficient.

7 Experiments

The low rank embedding can be used on various data mining applications. We evaluate the quality of the generated embedding vectors through three different application areas: clustering, anomaly detection, and feature selection. Please note that since not all the baselines have eigenvalue approximation, we didn't include DPIV (Section 4.2) in the experiment settings.

TABLE 2
Statistics of Datasets (Including Number of Instances, Features, and Clusters or Anomalies)

	Dataset	# ins.	# fea.	# clu.
1	20Newsgroups	18,846	26,214	20
2	Reuters21578	8,293	18,933	65
3	RCV1	193,844	47,236	103
4	Sector-Scale	9,619	55,197	105
5	USPS	9,298	256	10
6	MNIST	70,000	784	10
	Dataset	# ins.	# fea.	# ano.
7	20NG-10-11	4,991	26,214	100
8	Reuters21578AD	6,261	18,933	493
9	RCV1AD	7,803	29,992	200
10	magic04	19,020	10	6,688
11	satellite	6,435	36	2,036

7.1 Clustering

We perform K-means on the generated low-rank embeddings and evaluate the clustering result with NMI (Normalized Mutual Information [39]). Higher value of NMI means better embedding quality for clustering applications.

7.1.1 Datasets

We evaluate our algorithm on four text datasets : 20Newsgroups, Reuters21578, Sector-Scale and RCV1, and two image datasets USPS and MNIST, all summarized in Table 2. 20Newsgroups is a balanced dataset that covers 20 news topics. Reuters21578 is a collection of documents that appeared on Reuters newswire. Sector-scale dataset consists of company web pages classified in a hierarchy of industry sectors. RCV1 dataset contains an archive of manually categorized newswire stories made available by Reuters [22], and here we only include those documents with single topic category. Both of the USPS and MNIST datasets are 10 classes of handwritten digits. Reuters21578 and USPS are selected because they are unbalanced datasets with quite different size of clusters, while RCV1 and Sector-scale are chosen to show the potential effectiveness of ridge-regression over least square because of their high number of clusters.

For text datasets, cosine similarity (Section 5.1) is used. For USPS, MNIST, magic04 and satellite, Gaussian kernel (Section 5.2) is employed.

7.1.2 Baselines

We test two versions of our proposed algorithm: *DPIE-rr* (DPIE using ridge regression, or more specifically the version in [8]) and *DPIE-ls* (DPIE using simple least square regression, or more specifically the Conjugate Gradient Least Square (CGLS) [2]). For details about these two versions please refer to Section 4.3.

We select the following representative and diverse baselines for comprehensive comparison: 1) Spectral Embedding

TABLE 3
Clustering Results in NMI and Time Consuming

NMI	SE	PIE	PIE-k	FDSket	DFL	ColSpl	Rndm	DPIE-rr	DPIE-ls
20Newsgroups	0.5326	0.2519	0.3266	0.4877	0.4847	0.4541	0.4998	0.5025 (2)	**0.5061** (1)
Reuters21578	0.5048	0.2557	0.2718	**0.5322**	0.5014	0.4624	0.4519	0.5101 (3)	0.5143 (2)
RCV1	[38]*0.2875*	0.1022	0.1237	0.1521	0.1941	0.1837	**0.2708**	0.2701 (2)	0.2644 (3)
Sector-scale	0.3995	0.2211	0.2629	0.1275	0.3876	0.3273	0.3557	**0.4232** (1)	0.3945 (2)
USPS	0.6207	0.2026	0.2401	0.4667	**0.5871**	0.5167	0.5392	0.5623 (3)	0.5786 (2)
MNIST	0.4433	0.0022	0.0028	0.3522	0.3788	0.3342	0.3416	0.3922 (2)	**0.4032** (1)
Average	0.4778	0.1629	0.1930	0.3982	0.4292	0.3902	0.4207	0.4434 (2)	**0.4435** (1)
Time(s)	SE	PIE	PIE-k	FDSket	DFL	ColSpl	Rndm	DPIE-rr	DPIE-ls
20Newsgroups	5,653.0193	0.1461	5.0816	4131.7741	35.4688	6.0374	29.3142	5.4873	5.0834
Reuters21578	1,958.5777	0.0671	2.3548	830.7118	13.7681	2.0011	9.1132	1.8332	1.6388
RCV1	——	5.1961	110.5477	108,998.2234	923.6324	159.3213	562.3231	139.3193	127.6903
Sector-scale	5,832.7020	0.1673	5.7768	4,213.1350	37.3254	6.5263	30.1232	6.6233	5.8651
USPS	1,665.3840	0.0675	1.9807	395.9329	7.2451	0.8987	5.9983	0.7111	0.6584
MNIST	201,581.2017	4.0707	38.8645	46,072.8311	196.3723	50.1781	152.3238	48.8324	43.6582
Average	——	1.6191	27.4344	27,440.4300	202.3020	37.4938	131.5326	33.8011	30.7657

*For each dataset, the bold-faced number indicates the best approximation method (**excluding SE (NJW) which is the approximation target**), and the numbers in the parentheses indicate the ranks of our two versions of DPIE. Average is the average NMI and Time of each method across all the datasets respectively.* *We couldn't run SE on RCV1 dataset due to out-of-memory issue, but instead cite its NMI score from [38] for reference.

(**SE**) which is from the conventional spectral clustering NJW [34] (refer to Algorithm 1 with L_{sym}); it uses true eigen-decomposition which is time and space consuming but provides the best spectral embedding quality in theory; 2) Power Iteration Embedding [27] and 3) **PIE-k** [26] which are two classic power iteration based methods (refer to Sections 3.1 and 3.2); 4) Frequent-direction Sketching (*FDSket*) [25] which is a recently proposed deterministic matrix sketching method; 5) Deflation PIC (*DFL*) [40] that applies Schur complement deflation to extract non-overlapping embeddings/signals; 6) Column Sampling Approximation (*ColSpl*) [23] based on Nyström sampling method and 7) Randomized Sketching (*Rndm*) [12] that approximates the eigenvectors using random projections.

In clustering experiments, once we get the embeddings we perform a l_2-norm normalization along instance side that is similar to the last step of NJW (Algorithm 1), and a WCSS (minimizing within-cluster sum of squares, with 100 inner loops and 100 outer loops) K-means to obtain the cluster assignments.

7.1.3 Parameters

For clustering (and feature selection as well), suppose the number of clusters c to be a priori known as in previous works [34], [5], we set $k = c$ and use the first c embeddings of SE, FDSket, DFL, ColSpl and Rndm. PIE generates only one vector while PIE-k set k to be $\lceil \log(c) \rceil$ [26]. We set the maximum number of DPIEs to be $e = 6\lceil \log(c) \rceil$ out of $E = max(30\lceil \log(c) \rceil, 2c)$ random seeds. When we use Gaussian kernel to build the actual similarity matrix (e.g., in SE), 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. When using Gaussian kernel approximation (Section 5.2) we set the number of samples $d = 2{,}000$ and $\sigma = 2{,}000$. The maximum number of power iteration T is fixed to be $1{,}000$. 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 [27] and [26]. 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 7.4 we test DPIE stability with different ε and η.

It is also worth to mention the following: 1) As the other methods, we use normalized affinity matrix as the input in matrix sketching method (FDSket) to provide manifold insight; 2) In Nyström method (ColSpl), the number of sampled columns is fixed at $1{,}000$ as in [23], and these columns are selected by uniform sampling without replacement; and 3) For each method with sampling steps or random seeds, we run 50 times and report the average performance.

7.1.4 Result Analysis

The clustering results are summarized in Table 3. We reported the time used for the affinity matrix and embeddings constructions but we excluded the final K-means steps. For SE (NJW), we also excluded the time for the affinity matrix construction.

Generally speaking, SE 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 and space. Compared with PIE, PIE-k is 17 times slower on average since it requires more initial vectors to generate more PIE, but PIE-k improves about 18 percent on NMI because of its potential to contain different aspects of signal resulting from different starting vectors. However, it only gets 44 percent of SE in NMI. By truncated SVD on normalized affinity matrix, FDSket can deterministically extract the low rank approximation. So it covers additional signals in a more effective way than PIE-k (more than 70 percent better in NMI). But at the same time FDSket is also $1,000+$ times slower than PIE-k since it requires lots of SVD calculations. DFL 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

TABLE 4
Anomaly Detection Results in AUC and Time Consuming

AUC	SE	PIE	PIE-k	FDSket	DFL	IForest	ColSpl	Rndm	DPIE-rr	DPIE-ls
20NG-10-11	0.9022	0.3294	0.4858	0.6331	0.2318	0.6176	0.6013	0.6172	0.8832	**0.8844** (1)
Reuters21578AD	0.8145	0.3034	0.5131	0.4824	0.7863	0.6048	0.5904	0.7351	0.9236	**0.9271** (1)
RCV1AD	0.5504	0.4403	0.5049	0.4619	**0.5925**	0.4879	0.4762	0.5197	0.5565	0.5547 (2)
magic04	0.7184	0.5757	0.5757	0.5799	0.4205	**0.7506**	0.5732	0.5757	0.7183	0.7179 (3)
satellite	0.7065	0.3378	0.3378	0.5062	0.5416	0.7173	0.4873	0.6013	0.7154	**0.7193** (1)
Average	0.7384	0.3973	0.4835	0.5327	0.5145	0.6356	0.5457	0.6098	0.7594	**0.7607** (1)
Time(s)	SE	PIE	PIE-k	FDSket	DFL	IForest	ColSpl	Rndm	DPIE-rr	DPIE-ls
20NG-10-11	876.9247	0.0297	0.8683	181.7283	5.7138	7.6199	1.5279	4.9213	0.8685	0.8193
Reuters21578AD	4,141.9718	0.0528	1.1995	170.0181	7.3392	8.2016	2.5782	5.7284	1.1246	1.0608
RCV1AD	4,199.1405	0.0476	1.3253	475.9983	10.6519	5.5944	3.0021	8.3717	1.2023	1.1128
magic04	14,732.0387	0.1252	0.3402	3,241.6766	20.3112	53.8751	3.5443	17.6214	2.4211	2.2759
satellite	779.7334	0.0145	0.1121	152.7320	8.9713	49.3959	1.6767	7.1368	0.6343	0.5889
Average	4,945.9618	0.0540	0.7691	844.4307	10.5975	24.9374	2.4658	8.75592	1.2502	1.1715

*For each dataset, the bold-faced number indicates the best approximation method (**excluding SE which is the approximation target**), and the numbers in the parentheses indicate the ranks of our two versions of DPIE. Average is the average AUC and time of each method across all the datasets respectively.*

MNIST compared with FDSket. But since it requires more matrix computations in the deflation equation, it is noticeably much slower than PIE-k.

Our two versions of DPIE, although not always the best, stay on the top three among all the (approximate) methods on any dataset. Specifically, DPIE-ls achieves more than 95 percent performance of SE in NMI, and at the same time only requires quite a short running time which is close to PIE-k. And DPIE-rr outperforms DPIE-ls about $3\% \sim 7\%$ when the number of cluster in the dataset is high (e.g., RCV1, Sector-scale).

Due to out-of-memory problem, the SE experiment on RCV1 could not be finished since it requires full affinity matrix construction. However, using the space-efficient ways introduced in Section 5 it is not a problem for the other listed methods, especially our proposed DPIE. Notably, DPIE only takes about 2 minutes to process RCV1 dataset but more than 35 percent better than the second best approximation method with considerable faster speed.

7.2 Anomaly Detection

In a straightforward assumption, the node degree tells how isolated (anomalous) one sample is with respect to the surrounding neighborhood [1]. Therefore we compute degree matrix on the generated low-rank embeddings and evaluate the diagonal score in AUC (Area under Receiver Operating Characteristics Curve [33]), which is cut-off independent and commonly used to evaluate anomaly detectors [30]. Higher value of AUC means better embedding quality for anomaly detection applications.

7.2.1 Datasets

We choose three text datasets and two scientific datasets (summarized in Table 2). 20NG-10-11 is a subset of 20Newsgroups, which consists of all the samples from six 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 documents from "C15" category as anomalies and the rest of three categories as regular documents. Magic04 is a binary classification dataset from the UCI repository which was generated to simulate registration of high energy gamma particles. 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.

7.2.2 Baselines

Besides all the baselines used in clustering experiments, to have a more comprehensive comparison we also include IForest [30], which is a well-known anomaly detection algorithm for its efficiency and reasonable effectiveness at the same time. IForest detects data-anomalies with binary trees, using the property that anomalies are more susceptible to isolation.

7.2.3 Parameters

All the parameter setting are the same as in clustering. But it is worth to mention that: 1) We compute the degree matrix with (the first) five output embeddings for all the baselines except for PIE which again has only one embedding output; 2) In IForest, to conduct a safe and fair comparison, we set the sub-sampling size $\rho = 4{,}000$ and the number of trees $nt = 100$ because these parameters are the authors' recommendation [29]. For text dataset in IForest experiments, we use l_2-norm normalized X as input to make sure that the result is not sensitive to the document length.

7.2.4 Result Analysis

Table 4 shows the anomaly detection results. Similar to the clustering comparisons, PIE-k performes better than PIE (more than 20 percent improvement), with the reason that PIE-k is possible to provide more informative signals. DFL and FDSket can capture supplementary yet important

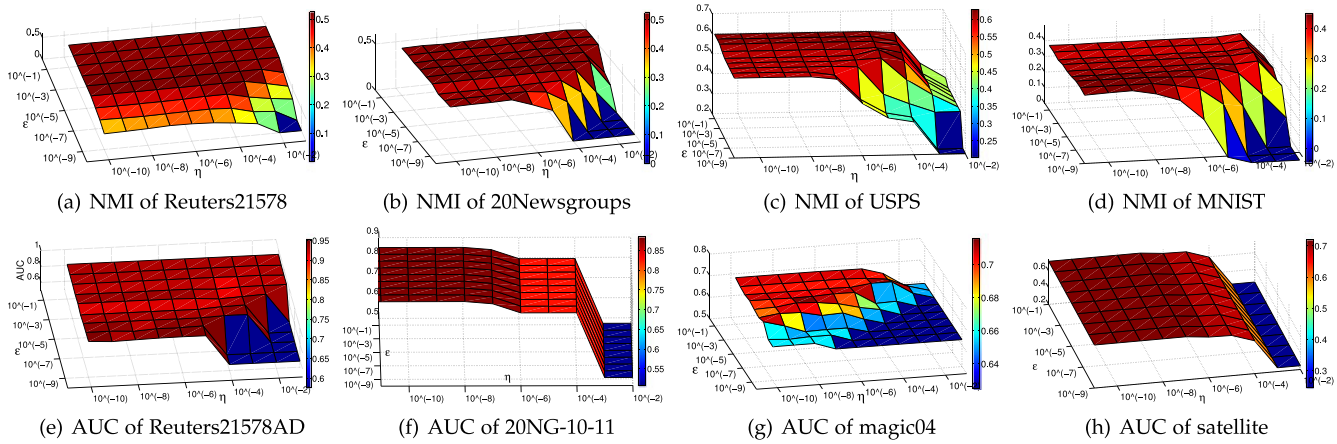
TABLE 5
Feature Selection Results in NMI

20Newsgroups	SE	PIE	PIEK	FDSket	DFL	ColSpl	Rndm	DPIE-rr	DPIE-ls
50	0.2971	0.1691	0.1590	0.2691	0.2552	0.2013	0.2311	0.3402 (2)	**0.3446** (1)
200	0.3361	0.3089	0.3181	0.3603	0.3274	0.2934	0.3352	0.3799 (2)	**0.3834** (1)
800	0.4118	0.3899	0.4115	0.4061	0.4256	0.3962	0.4011	0.4342 (2)	**0.4372** (1)
1200	0.4256	0.4696	0.4498	0.4692	0.4335	0.4513	0.4621	0.4803 (2)	**0.4819** (1)
1800	0.4865	0.4671	0.4587	0.4340	0.4748	0.4602	0.4683	0.4944 (2)	**0.4993** (1)
Average	0.3914	0.3609	0.3594	0.3877	0.3833	0.3605	0.3796	0.4258 (2)	**0.4293** (1)
Reuters21578	SE	PIE	PIEK	FDSket	DFL	ColSpl	Rndm	DPIE-rr	DPIE-ls
50	0.3957	0.3959	0.3889	**0.4399**	0.3973	0.3821	0.3913	0.4368 (2)	0.4366 (3)
200	0.4607	0.4539	0.4598	0.4745	0.4677	0.4599	0.4614	0.4767 (2)	**0.4814** (1)
800	0.5125	0.5021	**0.5183**	0.5113	0.4993	0.5097	0.5133	0.5203 (2)	0.5176 (3)
1200	0.5125	0.4783	0.4882	0.4971	0.5122	0.4923	0.5029	0.5245 (2)	**0.5297** (1)
1800	0.5081	0.5104	0.5078	0.4980	0.5200	0.5069	0.5136	0.5284 (2)	**0.5308** (1)
Average	0.4779	0.4681	0.4726	0.4842	0.4793	0.4702	0.4765	0.4973 (2)	**0.4992** (1)
RCV1	SE	PIE	PIEK	FDSket	DFL	ColSpl	Rndm	DPIE-rr	DPIE-ls
50	——	0.1266	0.1422	0.1531	0.1632	0.1512	0.1613	**0.1652** (1)	0.1621 (3)
200	——	0.1285	0.1436	0.1734	0.1867	0.1622	**0.2402**	0.2377 (2)	0.2232 (3)
800	——	0.1305	0.1523	0.1845	0.2029	0.1864	0.2683	**0.2709** (1)	0.2595 (3)
1200	——	0.1312	0.1567	0.1934	0.2239	0.2017	0.2815	**0.2853** (1)	0.2702 (3)
1800	——	0.1511	0.1732	0.2011	0.2188	0.2256	0.2817	**0.2861** (1)	0.2735 (3)
Average	——	0.1336	0.1536	0.1811	0.1991	0.1854	0.2466	**0.2490** (1)	0.2377 (3)
Sector-scale	SE	PIE	PIEK	FDSket	DFL	ColSpl	Rndm	DPIE-rr	DPIE-ls
50	0.3211	0.1284	0.1325	0.0877	0.3156	0.2721	0.3002	**0.3198** (1)	0.3013 (3)
200	0.3556	0.1624	0.1783	0.0945	0.3693	0.3134	0.3438	**0.3821** (1)	0.3613 (3)
800	0.3793	0.2181	0.2479	0.1161	0.3743	0.3216	0.3510	**0.4048** (1)	0.3852 (2)
1200	0.3868	0.2187	0.2572	0.1254	0.3852	0.3278	0.3522	**0.4286** (1)	0.4099 (2)
1800	0.4023	0.2205	0.2634	0.1283	0.3857	0.3309	0.3545	**0.4325** (1)	0.4122 (2)
Average	0.3690	0.1896	0.2159	0.1104	0.3660	0.3132	0.3403	**0.3936** (1)	0.3740 (2)
USPS	SE	PIE	PIEK	FDSket	DFL	ColSpl	Rndm	DPIE-rr	DPIE-ls
25	0.3843	0.1323	0.1331	0.3192	0.3612	0.3215	0.3564	0.3809 (2)	**0.3813** (1)
50	0.4792	0.1517	0.1614	0.4177	0.4513	0.4314	0.4416	**0.4823** (1)	0.4817 (2)
100	0.5853	0.2013	0.2213	0.4597	0.5624	0.4768	0.4882	**0.5734** (1)	0.5722 (2)
150	0.5862	0.2015	0.2395	0.4672	0.5711	0.4902	0.4973	**0.5851** (1)	0.5838 (2)
200	0.6192	0.2024	0.2413	0.4794	**0.6018**	0.5237	0.5399	0.6011 (2)	0.6001 (3)
Average	0.5308	0.1778	0.1993	0.4286	0.5096	0.4487	0.4647	**0.5246** (1)	0.5238 (2)
MNIST	SE	PIE	PIEK	FDSket	DFL	ColSpl	Rndm	DPIE-rr	DPIE-ls
25	0.3375	0.0011	0.0011	0.2511	0.3099	0.2764	0.2869	0.3306 (2)	**0.3322** (1)
50	0.3535	0.0015	0.0015	0.2974	0.3516	0.2811	0.3115	**0.3572** (1)	0.3563 (2)
100	0.4385	0.0013	0.0013	0.3166	0.3634	0.3197	0.3213	0.4356 (2)	**0.4378** (1)
150	0.4421	0.0018	0.0018	0.3293	0.3717	0.3231	0.3296	**0.4323** (1)	0.4312 (2)
200	0.4513	0.0019	0.0023	0.3465	0.3803	0.3311	0.3467	0.4404 (2)	**0.4499** (1)
Average	0.4046	0.0015	0.0016	0.3082	0.3554	0.3063	0.3192	0.3992 (2)	**0.4015** (1)
Total Average	——	0.2219	0.2337	0.3167	0.3821	0.3474	0.3711	**0.4149** (1)	0.4109 (2)

For each dataset, the bold-faced number indicates the best approximated method, and the numbers in the parentheses indicate the ranks of our two versions of DPIE. Average is the average NMI of each method on each dataset, while Total Average is the global average across all datasets. 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. *We couldn't run SE on RCV1 dataset due to out-of-memory issue.

eigenvectors, which leads to a 6 and 10 percent boost up respectively compared with PIE-k, but still much worse than SE (only about 70 percent). IForest is efficient in that it detects the anomalies by recording the short expected path lengths, so that it has 200 percent faster running time than SE and still acquires more than 85 percent performance of SE. However, our proposed DPIEs are $4{,}000+$ times faster than SE and yet reach the best average performance.

It is also very interesting to see that DPIE have better effectiveness than SE in this test. We do not have a very clear answer for this but the following answer is what we conjecture. As we claimed, DPIE can be interpreted using diffusion theorem and thus closely related instances are better connected, which is especially good for anomaly detection. However, such diffusion distances do not necessarily improve clustering effectiveness always because we need to find k clusters compared to outlier



(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. 6. Stability experiment with different acceleration threshold ε and normalized residual threshold η.

detection where the focus is to find instances not belong to majority.

7.3 Feature Selection

We exploit Multi-Cluster Feature Selection (MCFS) framework [5] (details will be explained later in this section) with the low-rank embeddings as the regression target to extract feature subset. Although it would be best to evaluate 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. Higher value of NMI means better feature subset quality, and therefore better embedding quality for feature selection.

The *datasets*, *baselines* and *parameters* used in this experiment are the same as those in clustering section. The experiments here are integrated within the MCFS framework [5] which measures the importance of each feature along each generated embedding that corresponds to each cluster by minimizing $\{min_{s_p}(\| v_p - Xs_p \|^2 + \beta |s_p|)\}$, where s_p is a m-dimensional vector and β controls the s_p's approximation speed to zero. For the jth feature, MCFS defines the feature importance as $max_p|s_{p,j}|$ where $s_{p,j}$ is the jth element of vector s_p. And we select the important feature subset based on the decreasing order of feature importance, and evaluate the quality of the feature subsets by WCSS K-means clustering. It is also worth to mention that in the MCFS framework, we perform l_2-norm normalization along sample side of X to evaluate uniform feature scales.

7.3.1 Result Analysis

We tested all the embedding construction methods within MCFS framework [5] with $\{50, 200, 800, 1{,}200, 1{,}800\}$ selected features on text datasets, and $\{25, 50, 100, 150, 200\}$ selected features on image datasets (due to their less number of features). The full result is reported in Table 5. Similar to clustering experiments, DFL and Rndm perform better than PIE-k and PIE, while ColSpl and FDSket have slightly worse performance. But DPIE extracts more representative features (always among the top-three methods across different datasets and feature subset sizes), which sometimes are even with better quality than those derived from original spectral embeddings (SE), such as on 20Newsgroups, Reuters21578 and Sector-scale. 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.

Comparably speaking, DPIE-rr outperforms DPIE-ls about 5 percent on the datasets with larger number of required DPIE (for those datasets with higher number of cluster such as RCV1 and Sector-scale). It is because of the shrinkage effect by ridge regression, and the consequently more accuracy of extracting informative DPIE.

7.4 Stability Experiments

We conduct experiments with different acceleration threshold ε and normalized residual threshold η to study the parameter tuning sensitivities of DPIE (for the sake of convenience we only include DPIE-ls in this test). The results are illustrated in Fig. 6. 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 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.

7.5 Time and Space Complexity Comparison

In the end, we give a general complexity comparison among all the baselines, which is shown in Table 6. For FDSket, n' is the size of each stream and ℓ is the size of matrix sketch ($\ell > k$). For IForest, n_{tr} is the number of trees and ρ is the number of sample. It is worth to mention that: 1) In our experiments, DPIE is implemented and run in parallel due to the designed efficiency characteristics. However, other algorithms (such as FDSket and DFL) are not capable to run in parallel; 2) Some operators inside each loop/iteration also contribute to the difference of actual running time, even the algorithms have the same big-O time complexity.

TABLE 6
Complexity Comparison, Where n is # of Instances, m is # of Features, and k is # of Low Embeddings

Algorithm	Space Complexity	Time Complexity
DPIE	$O(nm)$	$O(nmk)$
SE	$O(n^2)$	$O(n^2k)$
PIE	$O(nm)$	$O(nm)$
PIE-k	$O(nm)$	$O(nmk)$
FDSket	$O(m(n'+\ell))$	$O(nm\ell^2)$
DFL	$O(n^2))$	$O(n^2k)$
IForest	$O(nm)$	$O(n_{tr}(n+\rho)\rho)$
ColSpl	$O(nm)$	$O(nmk)$
Rndm	$O(n^2)$	$O(nk^2\ln(nk))$

8 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, inspired by the power iteration embedding (PIE [27]), can eliminate duplicated information due to a few dominant eigenvectors, which makes it achieve outstanding performance. We also proposed a way to calculate Diverse Power Iteration Value which gives weight or importance for each DPIE. Furthermore, we analyzed the effect of using different regression methods to construct DPIE and gave suggestions of choosing the proper model in different cases, and introduced a way to orthogonalize DPIE. The proposed algorithm 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 in the comparison with state-of-the-art baseline approximation algorithms.

Acknowledgments

The authors gratefully thank all the anonymous reviewers for constructive suggestions toward paper improvement. This research is supported in part by US National Science Foundation (NSF) (Nos. IIS-0949467, IIS-1047715, and IIS-1049448), National Natural Science Foundation of China (Nos. 61190120, 61190121, and 61190125) and by Brookhaven National Lab. (BNL) PD 15-025. It is also supported by US DoE, Grant No. DE-SC0003361, funded through the American Recovery and Reinvestment Act of 2009, and BSA/DOE Prime Contract (DE-AC02-98CH10886) to BNL. This paper is an extension of the work published in ICDM 2014 [19]. S. Yoo is the corresponding author.

References

[1] L. Akoglu, H. Tong, and D. Koutra, "Graph based anomaly detection and description: A survey," *Data Mining Knowl. Discovery*, pp. 1–63, 2014.
[2] O. Axelsson, "A generalized conjugate gradient, least square method," *Numerische Mathematik*, vol. 51, no. 2, pp. 209–227, 1987.
[3] A. Belloni, D. Chen, V. Chernozhukov, and C. Hansen, "Sparse models and methods for optimal instruments with an application to eminent domain," *Econometrica*, vol. 80, no. 6, pp. 2369–2429, 2012.
[4] U. Brandes, M. Gaertler, and D. Wagner, *Experiments on Graph Clustering Algorithms*. New York, NY, USA: Springer, 2003.
[5] D. Cai, C. Zhang, and X. He, "Unsupervised feature selection for Multi-cluster data," in *Proc. 16th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2010, pp. 333–342.
[6] X. Chen and D. Cai, "Large scale spectral clustering with Landmark-based representation," in *Proc. 25th AAAI Conf. Artif. Intell.*, 2011.
[7] R. R. Coifman and S. Lafon, "Diffusion maps," *Appl. Comput. Harmonic Anal.*, vol. 21, pp. 5–30, 2006.
[8] N. R Draper and H. Smith. *Applied Regression Analysis*. Hoboken, NJ, USA: Wiley, 2014.
[9] P. Drineas and A. Zouzias, "A note on Element-wise matrix sparsification via a Matrix-valued Bernstein inequality," *Inf. Process. Lett.*, vol. 11, pp. 385–389, 2011.
[10] C. Fowlkes, S. Belongie, F. Chung, and J. Malik, "Spectral grouping using the nyström method," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 2, pp. 214–225, Jan. 2004.
[11] M. Ghashami and J. M. Phillips, "Relative errors for deterministic Low-rank matrix approximations," in *Proc. ACM-SIAM Symp. Discrete Algorithms*, 2013.
[12] A. Gittens, P. Kambadur, and C. Boutsidis, "Approximate spectral clustering via randomized sketching," Ebay/IBM Research Tech. Rep., 2013.
[13] C. Hansen and D. Kozbur, "Instrumental variables estimation with many weak instruments using regularized jive," *J. Econometrics*, vol. 182, no. 2, pp. 290–308, 2014.
[14] X. He, D. Cai, and P. Niyogi, "Laplacian score for feature selection," in *Adv. Neural Inf. Process. Syst.*, 2006.
[15] H. Huang, H. Qin, S. Yoo, and D. Yu, "Local anomaly descriptor: A robust unsupervised algorithm for anomaly detection based on diffusion space," in *Proc. 21st ACM Int. Conf. Inf. Knowl. Manage.*, 2012, pp. 405–414.
[16] H. Huang, H. Qin, S. Yoo, and D. Yu, "A new anomaly detection algorithm based on quantum mechanics," *Proc. IEEE Int. Conf. Data Mining (ICDM)*, 2012.
[17] H. Huang, H. Qin, S. Yoo, and D. Yu, "Physics-based anomaly detection defined on manifold space," *ACM Trans. Knowl. Discovery Data*, vol. 9, no. 2, p. 14, 2014.
[18] H. Huang, S. Yoo, H. Qin, and D. Yu, "A robust clustering algorithm based on aggregated heat kernel mapping," in *Proc. IEEE Int. Conf. Data Mining*, 2011, pp. 270–279.
[19] H. Huang, S. Yoo, D. Yu, and H. Qin, "Diverse power iteration embeddings and its applications," in *Proc. IEEE Int. Conf. Data Mining (ICDM)*, 2014.
[20] H. Huang, S. Yoo, D. Yu, and H. Qin, "Noise-resistant unsupervised feature selection via Multi-perspective correlations," in *Proc. IEEE Int. Conf. Data Mining*, 2014, pp. 210–219.
[21] N. L. D. Khoa and S. Chawla, "Large scale spectral clustering using resistance distance and Spielman-teng solvers," in *Discovery Sci.*, pp. 7–21, 2012.
[22] D. D. Lewis, Y. Yang, T. G. Rose, and F. Li, "Rcv1: A new benchmark collection for text categorization research," *ACM SIGKDD Explorations Newslett.*, vol. 5, pp. 361–397, 2004.
[23] M. Li, X. Lian, J. T. Kwok, and B. Lu, "Time and space efficient spectral clustering via column sampling," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2011, pp. 2297–2304.
[24] Z. Li, Y. Yang, J. Liu, X. Zhou, and H. Lu, "Unsupervised feature selection using nonnegative spectral analysis," in *Proc. AAAI Conf. Artif. Intell.*, 2012.
[25] E. Liberty, "Simple and deterministic matrix sketching," in *Proc. 19th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2013, pp. 581–588.
[26] F. Lin, "Scalable methods for Graph-based unsupervised and Semi-supervised learning," Doctoral dissertation, Carnegie Mellon University, 2012.
[27] F. Lin and W. W. Cohen, "Power iteration clustering," in *Proc. Int. Conf. Mach. Learn.*, 2010.
[28] F. Lin and W. W. Cohen, "A very fast method for clustering big text datasets," in *Proc. 19th Eur. Conf. Artif. Intell.*, 2010, pp. 303–308.
[29] F. T. Liu and K. M. Ting, "Can Isolation-based anomaly detectors handle arbitrary multi-modal patterns in data?" Tech. Rep., 2010.

[30] F. T. Liu, K. M. Ting, and Z. H. Zhou, "Isolation forest," in *Proc. 8th IEEE Int. Conf. Data Mining*, 2008, pp. 413–422.
[31] J. Liu, C. Wang, M. Danilevsky, and J. Han, "Large-scale spectral clustering on graphs," in *Proc. 23rd Int. Joint Conf. Artif. Intell.*, 2013.
[32] U. V. Luxburg, "A tutorial on spectral clustering," *Statist. Comput.*, vol. 17, no. 4, pp. 395–416, 2007.
[33] C. Marzban, "A comment on the ROC curve and the area under it as performance measures," Tech. Rep., 2004.
[34] A. Ng, M. Jordan, and Y. Weiss, "On spectral clustering: Analysis and an algorithm," in *Proc. Adv. Neural Inf. Process. Syst.*, 2002, pp. 849–856.
[35] A. Rahimi and B. Recht, "Random features for Large-scale kernel machines," in *Proc. Adv. Neural Inf. Process. Syst.*, 2007.
[36] O. Shamir and N. Tishby, "Spectral clustering on a budget," *Proc. Int. Conf. Artif. Intell. Statist.*, 2011.
[37] J. R. Shewchuk, An introduction to the conjugate gradient method without the agonizing pain, 1994.
[38] Y. Song, W. Chen, H. Bai, C. Jin, and E. Y. Chang, "Parallel spectral clustering," *Mach. Learn. Knowl. Discovery Databases*, 2008, pp. 374–389.
[39] A. Strehl and J. Ghosh, "Cluster Ensembles - a knowledge reuse framework for combining multiple partitions," *J. Mach. Learn. Res.*, vol. 3, pp. 583–617, 2003.
[40] N. D. Thang, Y. K. Lee, and S. Lee, "Deflation-based power iteration clustering," *Appl. Intell.*, vol. 39, pp. 367–385, 2013.
[41] L. Wang, C. Leckie, K. Ramamohanarao, and J. Bezdek, "Approximate spectral clustering," in *Proc. Adv. Knowl. Discovery Data Mining*, 2009, pp. 134–146.
[42] S. White and P. Smyth, "A spectral clustering approach to finding communities in graph," in *Proc. SIAM Int. Conf. Data Mining*, 2005.
[43] D. Yan, L. Huang, and M. I. Jordan, "Fast approximate spectral clustering," in *Proc. 15th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2009, pp. 907–916.
[44] Z. Zhao and H. Liu, "Spectral feature selection for supervised and unsupervised learning," in *Proc. 24th Int. Conf. Mach. Learn.*, 2007, pp. 1151–1157.

Hao Huang received the BS degree from Sun Yat-sen University, China, and the PhD degree from Stony Brook University. He is currently working in the Machine Learning Lab at GE Global Research. His research focus on large-scale scientific data mining research, including manifold learning, anomaly detection, feature selection, clustering/classification, etc.

Shinjae Yoo received the bachelor's degree from Soongsil University, Korea, the MS degree from Seoul National University, and the PhD and MS degrees from Carnegie Mellon University. He is a computational scientist at BNL. His research interests are scientific data mining, text mining, network analysis, and scalable learning methods.

Dantong Yu received the BS degree in computer science from Beijing University and the PhD degree in computer science from the State University of New York at Buffalo. He is a research scientist in the Computational Science Center at BNL. His research interests cover renewable energy research, data mining, etc.

Hong Qin received the BS and MS degrees in computer science from Peking University, and the PhD degree in computer science from the University of Toronto. He is a professor in the Computer Science Department at Stony Brook University. His research interests include computer graphics, geometric and physics-based modeling, data mining, etc.