Power Iteration Clustering

Frank Lin FRANK@CS.CMU.EDU
William W. Cohen WCOHEN@CS.CMU.EDU
Carnegie Mellon University, 5000 Forbes Ave., Pittsburgh, PA 15213 USA

Abstract

We present a simple and scalable graph clustering method called *power iteration clustering* (PIC). PIC finds a very low-dimensional embedding of a dataset using truncated power iteration on a normalized pair-wise similarity matrix of the data. This embedding turns out to be an effective cluster indicator, consistently outperforming widely used spectral methods such as NCut on real datasets. PIC is very fast on large datasets, running over 1,000 times faster than an NCut implementation based on the state-of-the-art IRAM eigenvector computation technique.

1. Introduction

We present a *simple* and *scalable* clustering method called *power iteration clustering* (hereafter PIC). In essence, it finds a very low-dimensional data embedding using truncated power iteration on a normalized pair-wise similarity matrix of the data points, and this embedding turns out to be an effective cluster indicator.

In presenting PIC, we make connections to and make comparisons with spectral clustering, a well-known, elegant and effective clustering method. PIC and spectral clustering methods are similar in that both embed data points in a low-dimensional subspace derived from the similarity matrix, and this embedding provides clustering results directly or through a k-means algorithm. They are different in what this embedding is and how it is derived. In spectral clustering the embedding is formed by the bottom eigenvectors of the Laplacian of an similarity matrix. In PIC the embedding is an approximation to a *eigenvalue-weighted linear combination* of *all* the eigenvectors of an normalized similarity matrix. This embedding turns out to be very effective for clustering, and in comparison to spectral clustering, the cost (in space and time) of explicitly calculating eigenvectors is replaced by that of a small number of matrix-vector multiplications.

We test PIC on a number of different types of datasets and obtain comparable or better clusters than existing spectral methods. However, the highlights of PIC are its simplicity and scalability — we demonstrate that a basic implementation of this method is able to partition a network dataset of 100 million edges within a few seconds on a single machine, without sampling, grouping, or other preprocessing of the data.

This work is presented as follows: we begin by describing power iteration and how its convergence property indicates cluster membership and how we can use it to cluster data (Section 2). Then we show experimental results of PIC on a number of real and synthetic datasets and compare them to those of spectral clustering, both in cluster quality (Section 3) and scalability (Section 4). Next, we survey related work (Section 5), differentiating PIC from clustering methods that employ matrix powering and from methods that modifies the "traditional" spectral clustering to improve on accuracy or scalability. Finally, we conclude with why we believe this simple and scalable clustering method is very practical — easily implemented, parallelized, and well-suited to very large datasets.

2. Power Iteration Clustering

2.1. Notation and Background

Given a dataset $X = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n\}$, a *similarity function* $s(\mathbf{x}_i, \mathbf{x}_j)$ is a function where $s(\mathbf{x}_i, \mathbf{x}_j) = s(\mathbf{x}_j, \mathbf{x}_i)$ and $s \geq 0$ if $i \neq j$, and following previous work (Shi & Malik, 2000), $s = 0$ if $i = j$. An *affinity matrix* $A \in \mathcal{R}^{n \times n}$ is defined by $A_{ij} = s(\mathbf{x}_i, \mathbf{x}_j)$. The *degree matrix* D associated with A is a diagonal matrix with $d_{ii} = \sum_j Aij$. A *normalized affinity matrix* W is defined as $D^{-1}A$. Below we will view W interchangeably as a matrix, and an undirected graph with nodes

Appearing in *Proceedings of the 27th International Conference on Machine Learning*, Haifa, Israel, 2010. Copyright 2010 by the author(s)/owner(s).

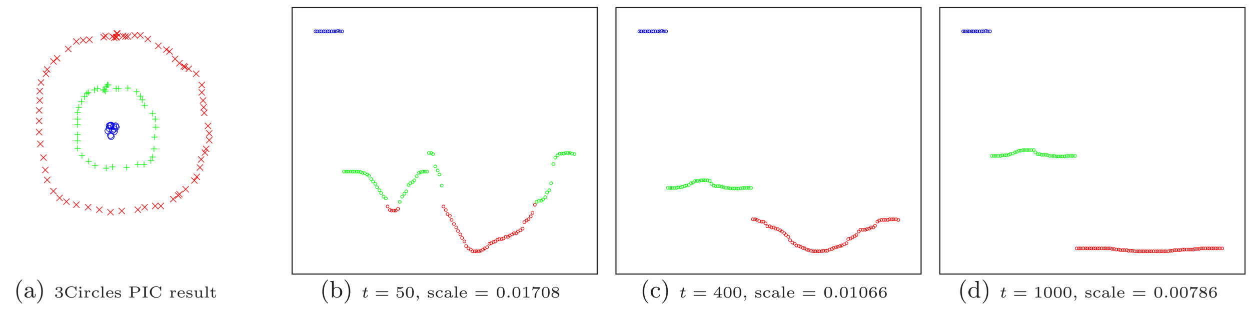

Figure 1. Clustering result and the embedding provided by $\mathbf{v}^t$ for the 3Circles dataset. In (b) through (d), the value of each component of $\mathbf{v}^t$ is plotted against its index. Plots (b) through (d) are re-scaled so the largest value is always at the very top and the minimum value at the very bottom, and *scale* is the maximum value minus the minimum value.

X and the edge from x_i to x_j weighted by $s(\mathbf{x}_i, \mathbf{x}_j)$.

W is closely related to the *normalized random-walk Laplacian* matrix L of Meilă and Shi (2001), defined as $L = I - D^{-1}A$. L has a number of useful properties: most importantly to this paper, the second-smallest eigenvector of L (the eigenvector with the second-smallest eigenvalue) defines a partition of the graph W that approximately maximizes the *Normalized Cut* criteria. More generally, the k smallest eigenvectors define a subspace where the clusters are often well-separated. Thus the second-smallest, third-smallest, $\ldots$, k^{th} smallest eigenvectors of L are often well-suited for clustering the graph W into k components.

Note that the k *smallest* eigenvectors of L are also the k *largest* eigenvectors of W. One simple method for computing the largest eigenvector of a matrix is *power iteration* (PI), also called the *power method*. PI is an iterative method, which starts with an arbitrary vector $\mathbf{v}^0 \neq \mathbf{0}$ and repeatedly performs the update

$$\mathbf{v}^{t+1} = cW\mathbf{v}^t$$

where c is a normalizing constant to keep $\mathbf{v}^t$ from getting too large (here $c = 1/\|W\mathbf{v}^t\|_1$). Unfortunately, PI does not seem to be particularly useful in this setting. While the k smallest eigenvectors of L (equivalently, the largest eigenvectors of W) are in general interesting (Meilă & Shi, 2001), the very smallest eigenvector of L (the largest of W) is not—in fact, it is a constant vector: since the sum of each row of W is 1, a constant vector transformed by W will never change in direction or magnitude, and is hence a constant eigenvector of W with eigenvalue $\lambda_1 = 1$.

2.2. Power Iteration Convergence

The central observation of this paper is that, while running PI *to convergence* on W does not lead to an interesting result, the intermediate vectors obtained by PI during the convergence process are extremely interesting. This is best illustrated by example. Figure 1(a) shows a simple dataset—i.e., each $\mathbf{x}_i$ is a point in $\mathcal{R}^2$ space, with $s(\mathbf{x}_i, \mathbf{x}_j)$ defined as $exp\left(\frac{-\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}\right)$. Figures 1(b) to 1(d) shows $\mathbf{v}^t$ at various values of t, each illustrated by plotting $\mathbf{v}^t(i)$ for each $\mathbf{x}_i$. For purposes of visualization, the instances $\mathbf{x}$ in the "bulls-eye" are ordered first, followed by instances in the central ring, then by those in the outer ring. We have also re-scaled the plots to span the same vertical distance—the scale is shown below each plot; as we can see the differences between the distinct values of the $\mathbf{v}^t$'s become smaller as t increases. Qualitatively, PI first converges *locally* within a cluster: by $t = 400$ the points from each cluster have approximately the same value in $\mathbf{v}^t$, leading to three disjoint line segments in the visualization. Then, after local convergence, the line segments draw closer together more slowly.

2.3. Further Analysis of PI's Convergence

Let us assume that W has eigenvectors $\mathbf{e}_1, \ldots, \mathbf{e}_n$ with eigenvalues $\lambda_1, \ldots, \lambda_n$, where $\lambda_1 = 1$ and $\mathbf{e}_1$ is constant. Given W, we define the *spectral representation* of a value $a \in \{1, \ldots, n\}$ to be the vector $\mathbf{s}_a = \langle \mathbf{e}_1(a), \ldots, \mathbf{e}_k(a) \rangle$, and define the *spectral distance between a and b* as

$$spec(a, b) \equiv \|\mathbf{s}_a - \mathbf{s}_b\|_2 = \sqrt{\sum_{i=2}^{k} (\mathbf{e}_i(a) - \mathbf{e}_i(b))^2}$$

Usually in spectral clustering it is assumed that the eigenvalues $\lambda_2, \ldots, \lambda_k$ are larger than the remaining ones. We define W to have an (α, β)-*eigengap between the k^{th} and $(k+1)^{th}$ eigenvector* if $\lambda_k/\lambda_2 \geq \alpha$ and $\lambda_{k+1}/\lambda_2 \leq \beta$. We will also say that W is γ_e-*bounded* if $\forall i, a, b \in \{1, \ldots, n\}$, $|\mathbf{e}_i(a) - \mathbf{e}_i(b)| \leq \gamma_e$; note that every W is γ_e-bounded for some γ_e. Letting $\mathbf{v}^t$ be the result of of the t^{th} iteration of PI, we define the

$(t, \mathbf{v}^0)$-*distance between a and b* as

$$pic^t(\mathbf{v}^0; a, b) \equiv |\mathbf{v}^t(a) - \mathbf{v}^t(b)|$$

For brevity, we will usually drop $\mathbf{v}^0$ from our notation (e.g., writing $pic^t(a, b)$). Our goal is to relate $pic^t(a, b)$ and $spec(a, b)$. Let us first define

$$signal^t(a, b) \equiv \sum_{i=2}^{k} [\mathbf{e}_i(a) - \mathbf{e}_i(b)] c_i \lambda_i^t$$
$$noise^t(a, b) \equiv \sum_{j=k+1}^{n} [\mathbf{e}_j(a) - \mathbf{e}_j(b)] c_j \lambda_j^t$$

Proposition 1. *For any W with $\mathbf{e}_1$ a constant vector,*

$$pic^t(a, b) = |signal^t(a, b) + noise^t(a, b)|$$

Proof. To verify the proposition, note that (ignoring renormalization)

$$\begin{aligned} \mathbf{v}^t &= W\mathbf{v}^{t-1} = W^2\mathbf{v}^{t-2} = ... = W^t\mathbf{v}^0 \\ &= c_1 W^t \mathbf{e}_1 + c_2 W^t \mathbf{e}_2 + ... + c_n W^t \mathbf{e}_n \\ &= c_1 \lambda_1^t \mathbf{e}_1 + c_2 \lambda_2^t \mathbf{e}_2 + ... + c_n \lambda_n^t \mathbf{e}_n \end{aligned}$$

Rearranging terms,

$$pic^t(a, b) = \Bigg| [\mathbf{e}_1(a) - \mathbf{e}_1(b)] c_1 \lambda_1^t + \sum_{i=2}^{k} [\mathbf{e}_i(a) - \mathbf{e}_i(b)] c_i \lambda_i^t + \sum_{j=k+1}^{n} [\mathbf{e}_j(a) - \mathbf{e}_j(b)] c_j \lambda_j^t \Bigg|$$

where the second and third terms correspond to $signal^t$ and $noise^t$ respectively, and the first term is zero because $\mathbf{e}_1$ is a constant vector. □

The implications of the proposition are somewhat clearer if we define a "radius" $R^t \equiv \frac{1}{\lambda_2^t}$ and consider the product of R^t and the quantities above:

$$R^t signal^t(a, b) = \sum_{i=2}^{k} [\mathbf{e}_i(a) - \mathbf{e}_i(b)] c_i \left(\frac{\lambda_i}{\lambda_2}\right)^t \quad (1)$$
$$R^t noise^t(a, b) \leq \sum_{j=k+1}^{n} \gamma_e c_j \beta^t \quad (2)$$

So, after rescaling points by R^t, we see that $noise^t$ will shrink quickly, if the β parameter of the eigengap is small. We also see that $signal^t$ is an approximate version of *spec*: the differences are that $signal^t$ is (a) compressed to the small radius R^t (b) has components distorted by c_i and $(\lambda_i/\lambda_2)^t$ and (c) has terms that are additively combined (rather than combined with Euclidean distance).

Note that the size of the radius is of no importance in clustering, since most clustering methods (e.g., k-means) are based on the relative distance between points, not the absolute distance. Furthermore, if the c_i's are not too large or too small, the distorting factors are dominated by the factors of $(\lambda_i/\lambda_2)^t$, which implies that the importance of the dimension associated with the i-th eigenvector is downweighted by (a power of) its eigenvalue; in Section 3.2 we will show that experimentally, this weighting scheme often *improves* performance for spectral methods.

We are then left with the difference (c) that the terms in the sum defining $signal^t$ are additively combined. How does this effect the utility of $signal^t$ as a cluster indicator? In prior work by Meilă and Shi (Meilă & Shi, 2001), it is noted that for many natural problems, W is approximately block-stochastic, and hence the first k eigenvectors are approximately piecewise constant over the k clusters. This means that if a and b are in the same cluster, $spec(a, b)$ would be nearly 0, and conversely if a and b are in different clusters, $spec(a, b)$ would be large.

It is easy to see that when $spec(a, b)$ is small, $signal^t$ must also be small. However, when a and b are in different clusters, since the terms are signed and additively combined, it is possible that they may "cancel each other out" and make a and b seem to be of the same cluster. Fortunately, this seems to be uncommon in practice when the number of clusters k is not too large. Hence, for large enough α, small enough t, $signal^t$ is very likely a good cluster indicator.

2.4. Early Stopping for PI

These observations suggest that an effective clustering algorithm might run PI for some small number of iterations t, stopping *after* it has converged within clusters but *before* final convergence, leadning to an approximately piecewise constant vector, where the elements that are in the same cluster have similar values. Specifically, define the *velocity at t* to be the vector $\boldsymbol{\delta}^t = \mathbf{v}^t - \mathbf{v}^{t-1}$ and define the *acceleration at t* to be the vector $\boldsymbol{\epsilon}^t = \boldsymbol{\delta}^t - \boldsymbol{\delta}^{t-1}$. We pick a small threshold $\hat{\epsilon}$ and stop PI when $\|\boldsymbol{\epsilon}^t\|_\infty \leq \hat{\epsilon}$.

Our stopping heuristic is based on the assumption and observation that while the clusters are "locally converging", the rate of convergence changes rapidly; whereas during the final global convergence, the converge rate appears more stable. This assumption turns out to be well-justified. Recall that $\mathbf{v}^t = c_1 \lambda_1^t \mathbf{e}_1 +$

Algorithm 1 The PIC algorithm

```
Input: A row-normalized affinity matrix $W$ and the number of clusters $k$
Pick an initial vector $\mathbf{v}^0$
repeat
    Set $\mathbf{v}^{t+1} \leftarrow \frac{W\mathbf{v}^t}{\|W\mathbf{v}^t\|_1}$ and $\boldsymbol{\delta}^{t+1} \leftarrow |\mathbf{v}^{t+1} - \mathbf{v}^t|$.
    Increment $t$
until $|\boldsymbol{\delta}^t - \boldsymbol{\delta}^{t-1}| \simeq \mathbf{0}$
Use $k$-means to cluster points on $\mathbf{v}^t$
Output: Clusters $C_1, C_2, ..., C_k$
```

$c_2\lambda_2^t\mathbf{e}_2 + ... + c_n\lambda_n^t\mathbf{e}_n$. Then

$$\frac{\mathbf{v}^t}{c_1\lambda_1^t} = \mathbf{e}_1 + \frac{c_2}{c_1}\left(\frac{\lambda_2}{\lambda_1}\right)^t \mathbf{e}_2 + ... + \frac{c_n}{c_1}\left(\frac{\lambda_n}{\lambda_1}\right)^t \mathbf{e}_n$$

It can be seen that the convergence rate of PI towards the dominant eigenvector $\mathbf{e}_1$ depends on $(\lambda_i/\lambda_1)^t$ for the significant terms $i = 2, ..., k$, since their eigenvalues are close to 1 if the clusters are well-separated (Meilă & Shi, 2001), making $(\lambda_i/\lambda_1)^t \simeq 1$. This implies that in the beginning of PI, it converges towards a linear combination of the top k eigenvectors, with terms $k+1, \ldots, n$ diminishing at a rate of $\geq (\lambda_{k+1}/1)^t$. After the noise terms $k+1, \ldots, n$ go away, the convergence rate towards $\mathbf{e}_1$ becomes nearly constant. The complete algorithm, which we call power iteration clustering (PIC), is shown in Figure 1. In all experiments in this paper, we let $\hat{\epsilon} = \frac{1\times10^{-5}}{n}$ where n is the number of data instances.

The convergence trajectory for PI will be the similar for any non-constant initial vector $\mathbf{v}^0$. However, we found it useful to let $\mathbf{v}^0(i)$ be $\frac{\sum_j Aij}{V(A)}$, where $V(A)$ is the volume of the affinity matrix A and $V(A) = \sum_i \sum_j Aij$. Since for each element in this vector also correspond to the degree distribution of the graph underlying the affinity matrix A, we will also call this vector the *degree vector* $\mathbf{d}$. The degree vector gives more initial weight to the high-degree nodes in the underlying graph, which means that, in the averaging view, values will be distributed more evenly and quickly, leading to faster local convergence.

As mentioned near the end of Section 2.3, when k is sufficiently large, collision of clusters on a single dimension may happen. In that case, we compute multiple $\mathbf{v}^t$'s with random $\mathbf{v}^0$'s. We then form a matrix V whose columns are $\mathbf{v}^t$'s and k-means is used rows of V as embedded data points. However, we find that often just one dimension is good enough — in fact, all the experiments done in this paper use only a single vector for embedding.

3. Accuracy of PIC

3.1. Experimental Comparisons

We demonstrate the effectiveness of PIC on a variety of real datasets; they have known labels and have been used for both classification and clustering tasks:

Iris contains flower petal and sepal measurements from three species of irises, two of which non-linearly separable (Fisher, 1936). 150 instances.
PenDigits01 and **PenDigits17** are hand-written digit datasets (Alimoglu & Alpaydin, 1997) with digits "0", "1" and "1", "7", respectively. 200 instances, 100 per digit. PenDigits01 represents an "easy" dataset and PenDigits17 a more "difficult" dataset.
PolBooks is co-purchase network of 105 political books (Newman, 2006). Each book is labeled "liberal", "conservative", or "neutral", mostly in the first two category.
UBMCBlog is a connected network dataset of 404 liberal and conservative political blogs mined from blog posts (Kale et al., 2007).
AGBlog is a connected network dataset of 1222 liberal and conservative political blogs mined from blog homepages (Adamic & Glance, 2005).
20ng* are subsets of the 20 newsgroups text dataset (Mitchell, 1997). 20ngA contains 100 documents from 2 newsgroups: *misc.forsale* and *soc.religion.christian*. 20ngB adds 100 documents to each group of 20ngA. 20ngC adds 200 from *talk.politics.guns* to 20ngB. 20ngD adds 200 from *rec.sport.baseball* to 20ngC.

For the network datasets (Polbooks, UBMGBlog, AGBlog), the affinity matrix is simply $A_{ij} = 1$ if blog i has a link to j or vice versa, otherwise $A_{ij} = 0$. For all other datasets, the affinity matrix is simply the cosine similarity between feature vectors: $\frac{\mathbf{x}_i \cdot \mathbf{x}_j}{\|\mathbf{x}_i\|_2 \|\mathbf{x}_j\|_2}$. Cosine similarity is used instead of the distance function in Figure 1 to avoid having to tune σ^2. For the text datasets, word counts are used as feature vectors with only stop words and singleton words removed.

We use these labeled datasets for clustering experiments and evaluate the clustering results against the labels using three measures: *cluster purity* (Purity), *normalized mutual information* (NMI), and *Rand index* (RI). All three measures are used to ensure a more thorough and complete evaluation of clustering results (for example, NMI takes into account cluster size distribution, which is ignored by Purity). Due to space constraints, we refer reader to (Manning et al., 2008) for details regarding these measures.

We also compare the results of PIC against those of spectral clustering methods Normalized Cuts (NCut) (Shi & Malik, 2000; Meilă & Shi, 2001) and the Ng-

Table 1. Clustering performance of PIC and spectral clustering algorithms on several real datasets. For all measures a higher number means better clustering. Bold numbers are the highest in its row.

			NCut			**NJW**			**PIC**	
Dataset	**k**	**Purity**	**NMI**	**RI**	**Purity**	**NMI**	**RI**	**Purity**	**NMI**	**RI**
Iris	3	0.6733	0.7235	0.7779	0.7667	0.6083	0.7978	**0.9800**	**0.9306**	**0.9741**
PenDigits01	2	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**
PenDigits17	2	**0.7550**	**0.2066**	**0.6301**	**0.7550**	0.2043	**0.6301**	**0.7550**	**0.2066**	**0.6301**
PolBooks	3	0.8476	0.5745	0.8447	0.8286	0.5422	0.8329	**0.8667**	**0.6234**	**0.8603**
UBMCBlog	2	**0.9530**	**0.7488**	**0.9104**	**0.9530**	0.7375	**0.9104**	0.9480	0.7193	**0.9014**
AGBlog	2	0.5205	0.0060	0.5006	0.5205	0.0006	0.5007	**0.9574**	**0.7465**	**0.9185**
20ngA	2	**0.9600**	**0.7594**	**0.9232**	**0.9600**	**0.7594**	**0.9232**	**0.9600**	**0.7594**	**0.9232**
20ngB	2	0.5050	0.0096	0.5001	0.5525	0.0842	0.5055	**0.8700**	**0.5230**	**0.7738**
20ngC	3	0.6183	0.3295	0.6750	0.6317	0.3488	0.6860	**0.6933**	**0.4450**	**0.7363**
20ngD	4	0.4750	0.2385	0.6312	0.5150	0.2959	0.6820	**0.5825**	**0.3133**	**0.7149**
Average		0.7308	0.4596	0.7393	0.7483	0.4581	0.7469	**0.8613**	**0.6267**	**0.8433**

Table 2. Clustering performance of eigenvalue-weighted NCut on several real datasets. For all measures a higher number means better clustering. Bold numbers are the highest in its row.

			uniform weights			$\mathbf{e}_i$ **weighted by** λ_i			$\mathbf{e}_i$ **weighted by** λ_i^{15}	
Dataset	**k**	**Purity**	**NMI**	**RI**	**Purity**	**NMI**	**RI**	**Purity**	**NMI**	**RI**
Iris	3	0.6667	0.6507	0.7254	**0.9800**	**0.9306**	**0.9741**	**0.9800**	**0.9306**	**0.9741**
PenDigits01	2	0.7000	0.2746	0.5800	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**	**1.0000**
PenDigits17	2	0.7000	0.1810	0.5800	**0.7550**	**0.2066**	**0.6301**	**0.7550**	**0.2066**	**0.6301**
PolBooks	3	0.4857	0.1040	0.4413	**0.8476**	0.5861	**0.8514**	0.8381	**0.5936**	0.8453
UBMCBlog	2	0.9505	0.7400	0.9059	0.9505	0.7400	0.9059	**0.9530**	**0.7488**	**0.9104**
AGBlog	2	0.9493	0.7143	0.9037	**0.9509**	**0.7223**	**0.9066**	0.9501	0.7175	0.9051
20ngA	2	0.5600	0.0685	0.5072	**0.9600**	**0.7594**	**0.9232**	0.9450	0.7005	0.8961
20ngB	2	0.7125	0.2734	0.5903	**0.9450**	**0.7042**	**0.8961**	0.5050	0.0096	0.5001
20ngC	3	0.6867	0.3866	0.6546	**0.6617**	0.3772	**0.7025**	0.6350	**0.4719**	0.6784
20ngD	4	0.4763	0.2365	0.6368	0.4875	0.2555	0.6425	**0.5263**	**0.2906**	**0.7129**
Average		0.6888	0.3630	0.6525	**0.8538**	**0.6282**	**0.8432**	0.8087	0.5670	0.8052

Jordan-Weiss algorithm (NJW) (Ng et al., 2002) and present the results in Table 1. In every experiment the k-means algorithm is run 100 times with random starting points and the most frequent cluster assignment is used. This is done so no algorithm would get "lucky"; the result reported is the *most likely* result and gives us an idea of the quality of the embedding. In practice, one may want to instead run several k-means trials as time allows and pick the one with the least within-cluster sum of squares, or even use another algorithm instead of k-means.

On most datasets PIC does equally well or does better than the other methods for most evaluation measures. In the case where NCut or NJW fails badly (AGBlog, 20ngB), the most likely cause is that the top k eigenvectors of the graph Laplacian fail to provide a good low-dimensional embedding for k-means. This might be improved by use of additional heuristics to choose the "good" eigenvectors and discard the "bad" eigenvectors (Zelnik-Manor & Perona, 2005; Li et al., 2007; Xiang & Gong, 2008). PIC, on the other hand, does not choose among eigenvectors — the embedding uses a weighted linear combinations of the *all* eigenvectors.

3.2. On Eigenvalue Weighting

As we can see in Section 2.3, in distance metric used by PIC, the i-th eigenvector is weighted by $c_i\lambda_i^t$, with λ^t dominating the weight term as the number of iteration t grows. In other words, the eigenvectors are weighted according to their corresponding eigenvalues raised to the power t. Based on analyses in (Meilă & Shi, 2001; von Luxburg, 2007), weighting the eigenvectors according to eigenvalues seems reasonable, since eigenvalues of a row-normalized affinity matrix range from 0 to 1 and good cluster indicators should have eigenvalues close to 1 and "spurious" eigenvalues close to 0 (the opposite is true in the case of normalized Laplacian matrices). To test this on real data, we run the NCut algorithm with the following modification: instead of using the first k eigenvectors, we use the first 10 vectors and weight them in different ways. First, we use them directly without any additional weighting. Second, we scale them by their corresponding eigenvalues. Lastly, we scale them by their corresponding eigenvalue raised to a power t. The result is shown in Table 2.

In Table 2, we can see that using the eigenvectors with uniform weights produces poor clustering on most datasets, even on PenDigits01, which is a relatively easy dataset. However, note that it does rather well on the blog datasets, and it even outperforms the original NCut algorithm on AGBlog, showing that original NCut is missing an important cluster indicator by choosing only the first k eigenvectors. In this light, we can view PIC as an approximation to the eigenvalue-weighted modification of NCut.

4. Scalability

Perhaps one of the greatest advantages of PIC lies in its scalability. Space-wise it needs only a single vector of size n for $\mathbf{v}^t$ and two more of the same to keep track of convergence. Speed-wise, power iteration is known to be fast on sparse matrices and converges fast on many real-world datasets; yet PIC converges *even more quickly* than power iteration, since it naturally stops when $\mathbf{v}^t$ is no longer accelerating towards convergence. Table 3 shows the runtime of PIC and the spectral clustering algorithms on datasets described in the previous section, and Table 4 shows the runtime on large, synthetic datasets.

For testing the runtime of the spectral clustering algorithms, we tried two different ways of finding eigenvectors. NCutE uses the slower, classic eigenvalue decomposition method to find all the eigenvectors. NCutI uses the fast *Implicitly Restarted Arnoldi Method* (IRAM) (Lehoucq & Sorensen, 1996), a memory-efficient version of the fast Lanczos algorithm for non-symmetric matrices. With IRAM we can ask only for the top k eigenvectors, resulting in less computation time.[1]

As we see PIC to be particularly useful for large datasets with sparse features, and large datasets with cluster labels are generally hard to find, we generated synthetic datasets with a modified version of the Erdős-Rényi random network model, which generates a connected block-stochastic graph with two components.[2] Results are averaged over five random network datasets and three trials for each dataset. We do not report the accuracy due to space constraints but all accuracies are above 0.99. The number of iteration PIC requires does not seem to increase with dataset size; real datasets averages 13 iterations and the largest synthetic dataset converges in 3 iterations. NCutE and NCutI were not run on the largest synthetic datasets because they required more memory than was available.[3]

Table 3. Runtime comparison (in milliseconds) of PIC and spectral clustering algorithms on several real datasets.

Dataset	Size	NCutE	NCutI	PIC
Iris	150	17	61	1
PenDigits01	200	28	23	1
PenDigits17	200	30	36	1
PolBooks	102	7	22	1
UBMCBlog	404	104	32	1
AGBlog	1,222	1095	70	3
20ngA	200	32	37	1
20ngB	400	124	56	3
20ngC	600	348	213	5
20ngD	800	584	385	10

Table 4. Runtime comparison (in milliseconds) of PIC and spectral clustering algorithms on synthetic datasets.

Nodes	Edges	NCutE	NCutI	PIC
1k	10k	1,885	177	1
5k	250k	154,797	6,939	7
10k	1,000k	1,111,441	42,045	34
50k	25,000k	-	-	849
100k	100,000k	-	-	2,960

5. Related Work

Clustering with Matrix Powering. (Tishby & Slonim, 2000) describes a clustering method that turns the similarity matrix into a Markov process and then examine the decay of mutual-information during the relaxation of this process, and then as clusters emerge, they are extracted using the information bottleneck method (Tishby et al., 1999). Besides having a information-theoretic approach, this method is very different from PIC, (a) it uses matrix-matrix instead of matrix-vector multiplication, and (b) the clusters are extracted from the relaxed matrix using the information bottleneck method. Note these differences make this approach less appealing in terms of scalability. (Zhou & Woodruff, 2004) describes a simpler algorithm, also involving matrix-matrix multiplication, on an unnormalized similarity matrix. However, it is not clear how to choose two crucial parameters: the power t and the two-cluster threshold ϵ and no results on real datasets were reported. A more general framework using diffusion kernels (powered similarity matrices) for dimensionality reduction, clustering, and semi-supervised learning is presented in (Lafon & Lee,

[1]Due to space constraints NJW runtimes are not reported, but they are very similar to that of NCut.

[2]The n nodes are in two equal-sized blocks, and the number of edges is $0.01n^2$. We define an additional parameter $e = 0.2$, and when generating a edge between two nodes, with probability $1-e$ the edge is placed between two random nodes of the same cluster and otherwise between two nodes of different clusters.

[3]Implemented in MATLAB and ran on a Linux machine with two quad-core 2.26Ghz CPUs and 24GB RAM.

2006), and a connection is made between diffusion kernels and spectral clustering methods.

Spectral Clustering. Spectral clustering began with the discovery of the correlation between the eigenvalues of the Laplacian matrix and the connectivity of a graph (Fiedler, 1973). Much later it was introduced to the machine learning community through Ratio Cut (Roxborough & Sen, 1997) and Normalized Cuts (Shi & Malik, 2000), and since it has sparked much interest and lead to further analyses and modifications (Ng et al., 2002; von Luxburg, 2007). Typically, a spectral clustering algorithm first defines a Laplacian matrix. Then the k smallest eigenvectors (those with the smallest corresponding eigenvalues), deemed the most informative, are used to embed the data onto a k-dimensional space. Finally, clusters are obtained with a k-means algorithm. While shown to be effective on many datasets, and noted particularly for dealing with non-linearly separable data, these "classical" spectral clustering approaches have two inherent shortcomings:

(1) Finding eigenvectors of a large matrix is computationally costly. It takes $O(n^3)$ in general, and even with fast approximating techniques like IRAM (which can run into convergence issues), much space and time are required for larger datasets. Recent work aims to address this problem with sampling techniques (Fowlkes et al., 2004; Yan et al., 2009) or a multilevel approach (Tolliver et al., 2005). By making certain assumptions about the structure of the data, the eigenvector-finding is done only on a small subset of the data, avoiding costly computation.

(2) Spectral clustering decides the "informative" eigenvectors used for the embedding *a priori* (usually the smallest k eigenvectors). This assumption seems to fail on many real datasets, especially when there is much noise. It is not unlikely that the k-th eigenvector corresponds to some particularly salient noise in the data, and while the k+1-th eigenvector contains good cluster indicators, it is missed entirely. This prompted much work on selecting "good" eigenvectors and dealing noise in spectral clustering (Zelnik-Manor & Perona, 2005; Li et al., 2007; Xiang & Gong, 2008). However, for every additional eigenvector that is evaluated for its quality, more computational cost is incurred for both finding the eigenvector and evaluating it.

PIC is related to spectral clustering in that it finds a low-dimensional embedding of data, and a k-means algorithm is used to produce the final clusters. But as the results in this paper show, it is *not* necessary to find *any* eigenvector (as most spectral clustering methods do), in order to find a low-dimensional embedding for clustering—the embedding just needs to be a good linear combination of the eigenvectors. In this respect, PIC is a very different approach from the spectral methods mentioned above.

Another recent graph clustering approach that has shown substantial speed improvement over spectral clustering methods is *multilevel kernel k-means* (Dhillon et al., 2007), where the general *weighted kernel k-means* is shown to be equivalent to a number of spectral clustering methods in its objective when the right weights and kernel matrix are used. Performance wise, spectral clustering methods are slow but tend to get globally better solutions, whereas kernel k-means is faster but get stuck easily in a local minima. This work exploits this trade-off using a multilevel approach: first an iterative coarsening heuristic is used to reduce the graph to one with $5k$ nodes where k is the number of desired clusters. Then spectral clustering is used on this coarse graph to produce a base clustering, and then the graph and is refined iteratively (to undo the coarsening), and at each refinement iteration the clustering results of the previous iteration is used as the starting point for kernel k-means. Additional point-swapping can be used to further avoid being trapped in local minima.

Semi-Supervised Methods. PIC's particular repeated matrix-vector multiplication can be viewed as a sort of iterative averaging or a backward random walk. While this behavior has been used in the graph-based semi-supervised learning community to propagate class label information, (Zhu et al., 2003; Macskassy & Provost, 2007; Baluja et al., 2008; Talukdar et al., 2008), and its "clustering effect" has been noted in (Crawell & Szummer, 2007), the current work, as far as we know, is the first to take advantage of it to develop a simple and scalable clustering method and test it on synthetic and real datasets.

6. Conclusion

We describe a novel and simple clustering method based on applying power iteration to the row-normalized affinity matrix of the data points. It is easy to understand and implement (matrix-vector multiplications), readily parallelizable (Page et al., 1998; Kang et al., 2009), and very efficient and scalable in terms of time and space. Experiments on a number of different types of labeled datasets show that PIC is able to obtain clusters that are just as well, if not better, than some of the spectral clustering methods. Experiments also show that in practice PIC is very fast without using sampling techniques — which also makes PIC a good candidate for sampling techniques for potentially even greater scalability gains.

Acknowledgments

This work was funded by NSF under grant IIS-0811562, by NIH under grant R01 GM081293, and by a gift from Google.

References

Adamic, Lada and Glance, Natalie. The political blogosphere and the 2004 U.S. election: Divided they blog. In *WWW Workshop on the Weblogging Ecosystem*, 2005.

Alimoglu, F. and Alpaydin, Ethem. Combining multiple representations and classifiers for handwritten digit recognition. In *ICDAR*, 1997.

Baluja, Shumeet, Seth, Rohan, Sivakumar, D., Jing, Yushi, Yagnik, Jay, Kumar, Shankar, Ravichandran, Deepak, and Aly, Mohamed. Video suggestion and discovery for YouTube: Taking random walks through the view graph. In *WWW*, 2008.

Crawell, Nick and Szummer, Martin. Random walks on the click graph. In *SIGIR*, 2007.

Dhillon, Inderjit S., Guan, Yuqiang, and Kulis, Brian. Weighted graph cuts without eigenvectors: A multilevel approach. *PAMI*, 29(11):1944–1957, 2007.

Fiedler, Miroslav. Algebraic connectivity of graphs. *Czechoslovak Mathematical Jour.*, 23(98):298–305, 1973.

Fisher, R. A. The use of multiple measurements in taxonomic problems. *Annual Eugenics*, 7(2):179–188, 1936.

Fowlkes, Charless, Belongie, Serge, Chung, Fan, and Malik, Jitendra. Spectral grouping using the Nyström Method. In *PAMI*, 2004.

Kale, Anubhav, Karandikar, Amit, Kolari, Pranam, Java, Akshay, Finin, Tim, and Joshi, Anupam. Modeling trust and influence in the blogosphere using link polarity. In *ICWSM 2007*, 2007.

Kang, U, Tsourakakis, Charalampos E., and Faloutsos, Christos. Pegasus: A peta-scale graph mining system - implementation and observations. In *ICDM*, 2009.

Lafon, Stéphane and Lee, Ann B. Diffusion maps and coarse-graining: A unified framework for dimensionality reduction, graph partitioning, and data set parameterization. *PAMI*, 28(9):1393–1403, 2006.

Lehoucq, R.B. and Sorensen, D. C. Deflation techniques for an implicitly re-started arnoldi iteration. *SIMAX*, 17:789–821, 1996.

Li, Zhenguo, Liu, Jianzhuang, Chen, Shifeng, and Tang, Xiaoou. Noise robust spectral clustering. In *ICCV*, 2007.

Macskassy, Sofus A. and Provost, Foster. Classification in networked data: A toolkit and a univariate case study. *JMLR*, 8:935–983, 2007.

Manning, Christopher D., Raghavan, Prabhakar, and Schtze, Hinrich. *Introduction to Information Retrieval.* Cambridge University Press, 2008.

Meilă, Marina and Shi, Jianbo. A random walks view of spectral segmentation. In *AISTAT*, 2001.

Mitchell, Tom. *Machine Learning.* McGraw Hill, 1997.

Newman, M. E. J. Finding community structure in networks using the eigenvectors of matrices. *Physical Review E*, 74:036104, 2006.

Ng, Andrew Y., Jordan, Michael, and Weiss, Yair. On spectral clustering: Analysis and an algorithm. In *NIPS*, 2002.

Page, Lawrence, Brin, Sergey, Motwani, Rajeev, and Winograd, Terry. The PageRank citation ranking: Bringing order to the web. Technical report, Stanford Digital Library Technologies Project, 1998.

Roxborough, Tom and Sen, Arunabha. Graph clustering using multiway ratio cut. In *Graph Drawing*, 1997.

Shi, Jianbo and Malik, Jitendra. Normalized cuts and image segmentation. *PAMI*, 22(8):888–905, 2000.

Talukdar, Partha Pratim, Reisinger, Joseph, Paşca, Marius, Ravichandran, Deepak, Bhagat, Rahul, and Pereira, Fernando. Weakly-supervised acquisition of labeled class instances using graph random walks. In *EMNLP*, 2008.

Tishby, Naftali and Slonim, Noam. Data clustering by markovian relaxation and the information bottleneck method. In *NIPS*, 2000.

Tishby, Naftali, Pereira, Fernando C., and Bialek, William. The information bottleneck method. In *The 37th Annual Allerton Conference on Communication, Control and Computing*, 1999.

Tolliver, David, Collins, Robert T., and Baker, Simon. Multilevel spectral partitioning for efficient image segmentation and tracking. In *The Seventh IEEE Workshops on Application of Computer Vision*, 2005.

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

Xiang, Tao and Gong, Shaogang. Spectral clustering with eigenvector selection. *Pattern Recognition*, 41 (3):1012–1029, 2008.

Yan, Donghui, Huang, Ling, and Jordan, Michael I. Fast approximate spectral clustering. In *KDD*, 2009.

Zelnik-Manor, Lihi and Perona, Pietro. Self-tuning spectral clustering. In *NIPS*, 2005.

Zhou, Hanson and Woodruff, David. Clustering via matrix powering. In *PODS*, 2004.

Zhu, Xiaojin, Ghahramani, Zoubin, and Lafferty, John. Semi-supervised learning using Gaussian fields and harmonic functions. In *ICML*, 2003.