Power Iteration Clustering

Frank Lin* William W. Cohen†

Abstract

We show that the power iteration, typically used to approximate the dominant eigenvector of a matrix, can be applied to a normalized affinity matrix to create a one-dimensional embedding of the underlying data. This embedding is then used, as in spectral clustering, to cluster the data via k-means. We demonstrate this method's effectiveness and scalability on several synthetic and real datasets, and conclude that to find a meaningful low-dimensional embedding for clustering, it is *not* necessary to find *any* eigenvectors—we just need a linear combination of the top eigenvectors.

1 Introduction

Spectral clustering is an effective and elegant clustering method based on the pairwise similarity between objects. Here we present a fast and simple spectral-clustering like technique called *power iteration clustering*. As in spectral clustering, points are embedded in a low-dimensional subspace derived from the similarity matrix for the data points; however, while in spectral clustering, the subspace is derived from the bottom eigenvectors of the Laplacian of an affinity matrix, in our proposed method, the subspace is an approximation to a linear combination of these eigenvectors.

We show that our method obtains comparable or better clusters than existing spectral methods; however, the most important advantages of the method are its simplicity and scalability. In particular, the subspace we use is a one-dimensional subspace formed by using the power iteration *with early stopping* on a normalized affinity matrix (the power iteration is normally run to convergence in order to find the dominant eigenvector of a matrix). This technique is simple, scalable, easily parallelized, and quite 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 [15, 12], $s = 0$ 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 a graph with nodes 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 [11], which is 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 [11]. More generally, the k smallest eigenvectors define a subspace where the clusters are well-separated. Thus the second-smallest, third-smallest, ..., k^{th} smallest eigenvectors of L are often well-suited for clustering the graph W into k components.

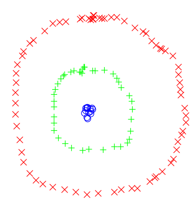
Note that the k *smallest* eigenvectors of L are also the k *largest* eigenvectors of W. One simple method for computing the largest eigenvector of a matrix is *power iteration* (hereafter PI), also called the *power method*. PI is an iterative method, which starts with an arbitrary vector $\mathbf{v}^0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

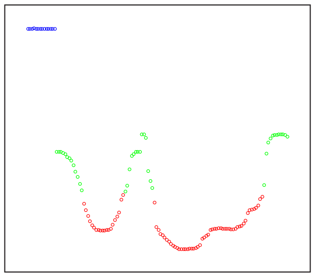
$$\mathbf{v}^{t+1} = cW\mathbf{v}^t$$

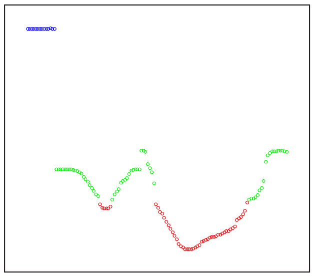
where c is a normalizing constant to keep $\mathbf{v}^t$ from getting too large and typically $c = 1/\|W\mathbf{v}^t\|_1$ or $1/\|W\mathbf{v}^t\|_2$. When applied to a *column*-normalized affinity matrix W^T, each iteration simulates a step in a Markov random walk with starting distribution $\mathbf{v}^0$ and transition probability matrix W^T, and $\mathbf{v}^t$ converges to a stable distribution if the graph underlying W^T is non-bipartite. Since the operations are simple (matrix-vector multiplications), fast (if W^T is sparse), space-efficient (only $\mathbf{v}^t$ needs to be stored), parallelizable and easily implemented in a distributed computing environment, power iteration is suitable for sparse, large-scale data—for example, web page ranking in the PageRank algorithm [13].

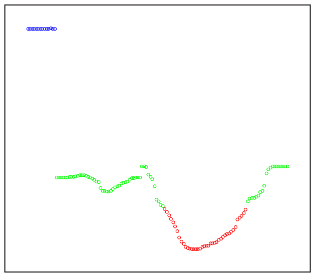
Unfortunately, PI does not seem to be particularly useful in this setting. While the k smallest eigenvectors of L (equivalently, the largest eigenvectors of W) are in general interesting, the very smallest eigenvector of L (the largest eigenvector of W) is not—in fact, it is a

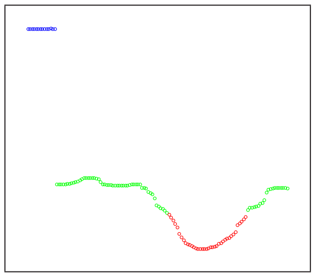
*Carnegie Mellon University
†Carnegie Mellon University

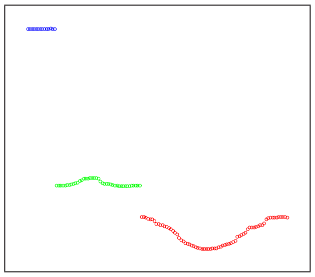

(a) 3Circles PIC result

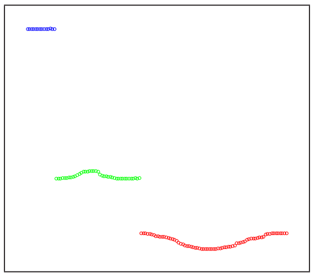

(b) $t = 10$, scale $= 0.01925$


(c) $t = 50$, scale $= 0.01708$


(d) $t = 100$, scale $= 0.01537$


(e) $t = 200$, scale $= 0.01316$


(f) $t = 400$, scale $= 0.01066$


(g) $t = 600$, scale $= 0.00929$

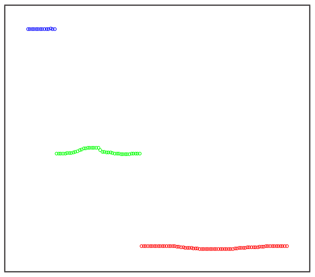

(h) $t = 1000$, scale $= 0.00786$

Figure 1: Clustering result and the embedding provided by $\mathbf{v}^t$ for the 3Circles dataset. In (b) through (h), the value of each component of $\mathbf{v}^t$ is plotted against its index. Plots (b) through (h) 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.

constant vector. To see this, note that 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(h) 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, followed by instances in the outer ring. We have also re-scaled the plots to span the same vertical distance—the scale (difference between the maximum and the minimum value of $\mathbf{v}^t$) 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 $t = 600$ 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.

This behavior can be understood by recognizing that PI is a sort of iterative averaging. Variants of iterative averaging are often used to propagate class label information through the graph in graph-based semi-supervised learning methods (e.g., [22], [10], [3, 16]). For PI, at each iteration, an element i in $\mathbf{v}^t$ is updated to $\mathbf{v}^t(i) \leftarrow \frac{1}{c}\sum_j W_{ij}\mathbf{v}^{t-1}(j)$ where c is some constant; this means that the element i is set to a weighted average of its neighbors in the underlying graph weighted according to W_{ij}. If the underlying graph is connected, repeated averaging will eventually make all nodes have the same value; however, sets of nodes that are near each other will quickly attain a similar value, while nodes that are far away in the graph will converge in value more slowly.

2.3 A Random Walks View If PI on a column-normalized affinity matrix W^T simulates a Markov random walk and $\mathbf{v}^t$ tells us the walk distribution at time t if we started with distribution $\mathbf{v}^0$, then PI on a row-normalized affinity W simulates a Markov random walk *backward* and $\mathbf{v}^t$ tells us what is the *most likely* distribution t steps in the past if the walk ended with distribution $\mathbf{v}^0$.

Suppose that the data consists of two clusters such that the graph underlying A consists of two components A_1 and A_2 and the sum of intra-component edge weights are high and the sum of inter-component edge weights are low. Let $\mathbf{v}^0$ be the distribution of a Markov random walk after t steps from an unknown starting distribution, and let d be the sum of elements of $\mathbf{v}^0$ that are in component A_1 and $d-1$ be those of component A_2. We then ask this question: what is the most likely starting distribution? Tracing random walks backwards, we see that, if t is not too large, a random walker in A_1 is still most likely in A_1 t steps in the past and a random walker in A_2 is also most likely in A_2 in the past. However, since intro-component nodes are highly connected, a random walker that ends up in a particular node in A_1 could have come from any where in A_1 t steps in the past, making the distribution close to uniform within A_1 (the same could be said of A_2). Of course, if t is sufficiently large or infinite, then there would have been enough time for walkers to mix between A_1 and A_2, making distribution uniform over the entire graph.

2.4 Analysis of PI's Convergence Rate 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

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

We will say that $\mathbf{v}^0$ is *(c_{lo}, c_{hi})-bounded* if $\mathbf{v}^0 = c_1\mathbf{e}_1 + \ldots + c_n\mathbf{e}_n$ and $\forall i \in \{1, \ldots, n\}$, $c_{lo} \leq c_i \leq c_{hi}$. For brevity, we will usually drop $\mathbf{v}^0$ from our notation (e.g., writing $pic^t(a, b)$) but assume that $\mathbf{v}^0$ is (c_{lo}, c_{hi})-bounded.

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 2.1. *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)

$$\begin{aligned} \mathbf{v}^t &= W\mathbf{v}^{t-1} = W^2\mathbf{v}^{t-2} = \ldots = W^t\mathbf{v}^0 \\ &= c_1W^t\mathbf{e}_1 + c_2W^t\mathbf{e}_2 + \ldots + c_nW^t\mathbf{e}_n \\ &= c_1\lambda_1^t\mathbf{e}_1 + c_2\lambda_2^t\mathbf{e}_2 + \ldots + c_n\lambda_n^t\mathbf{e}_n \end{aligned}$$

Rearranging terms,

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

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" $R^t \equiv \frac{1}{c_{lo}\lambda_2^t}$ and consider the product of R^t and the quantities above. Clearly

$$R^t noise^t(a, b) \leq (n-k)\gamma_e \frac{c_{hi}}{c_{lo}} \beta^t \tag{2.1}$$

$$R^t signal^t(a, b) = \sum_{i=2}^{k} [\mathbf{e}_i(a) - \mathbf{e}_i(b)] \frac{c_i}{c_{lo}} \left(\frac{\lambda_i}{\lambda_2}\right)^t \tag{2.2}$$

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 some sort of approximate version of *spec*: the differences are that $signal^t$ is (a) compressed to the small radius R^t (b) has components distorted by factors of $\frac{c_i}{c_{lo}}\left(\frac{\lambda_i}{\lambda_2}\right)^2$ 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; also,

```
Input: A row-normalized affinity matrix $W$ and the number of clusters $k$.
Output: Clusters $C_1, C_2, ..., C_k$.

1. Pick an initial vector $\mathbf{v^0}$.
2. Set $\mathbf{v^{t+1}} \leftarrow \frac{W\mathbf{v^t}}{\|W\mathbf{v^t}\|_1}$ and $\boldsymbol{\delta}^{t+1} \leftarrow |\mathbf{v^{t+1}} - \mathbf{v^t}|$.
3. Increment $t$ and repeat above step until $|\boldsymbol{\delta}^t - \boldsymbol{\delta}^{t-1}| \simeq \mathbf{0}$.
4. Use $k$-means to cluster points on $\mathbf{v^t}$ and return clusters $C_1, C_2, ..., C_k$.
```

Figure 2: The PIC algorithm.

if the first few eigenvalues are close to one (i.e., α is large) and c_{lo} is not too small, the distorting factors are bounded. We are left with the problem that the term in the sum defining $signal^t$ are additively combined.

We can make a much stronger connection between $signal^t$ and $spec$ if we are willing to assume more about W. More precisely, let a *clustering* cl for W be a function $cl : \{1, \ldots, n\} \rightarrow \{2, \ldots, k\}$. Meilă and Shi [11] note that for many natural problems, W is approximately block-stochastic, and the first few non-dominant eigenvectors are approximately piecewise constant over clusters. To formalize this, define W to be $(\gamma_{cl}, \delta_{cl})$-*clusterable* if there is a clustering cl such that

1. $\forall a, b$ if $cl(a) = cl(b)$ then

$$\forall i : 2 \leq k, |\mathbf{e}_i(a) - \mathbf{e}_i(b)| \leq \delta_{cl} \tag{2.3}$$

2. $\forall a, b$ if $cl(a) \neq cl(b)$ then there is some $i : 2 \leq i \leq k$ such that

$$|\mathbf{e}_i(a) - \mathbf{e}_i(b)| \geq \gamma_{cl} \tag{2.4}$$

$$\forall j \neq i, 1 \leq j \leq n, |\mathbf{e}_j(a) - \mathbf{e}_j(b)| \leq \delta_{cl} \tag{2.5}$$

The immediate implication of this is that $spec(a, b)$, whenever it is large, is the result of a large difference in a single eigenvector, which means that summing the differences between the individual eigenvectors at a and b is a good approximation. We have the following result.

Theorem 2.1. *If W is $(\gamma_{cl}, \delta_{cl})$-clusterable and has an (α, β)-eigengap, then*

- *If $cl(a) \neq cl(b)$, then $spec(a, b) \geq \gamma_{cl}$ and $R^t signal^t(a, b) \geq \gamma_{cl}\alpha^t - (k-2)\frac{c_{hi}}{c_{lo}}\delta_{cl}$*
- *If $cl(a) = cl(b)$, then $spec(a, b) \leq \sqrt{(k-1)\delta_{cl}^2}$ and $R^t signal^t(a, b) \leq (k-1)\frac{c_{hi}}{c_{lo}}\delta_{cl}$*

Proof. The claim that $spec(a, b) \geq \gamma_{cl}$ when $cl(a) \neq cl(b)$ follows from the definition of $spec(a, b)$ and Eq. 2.4. The claim that $R^t signal^t(a, b) \geq \gamma_{cl}\alpha^t - (k-2)\frac{c_{hi}}{c_{lo}}\delta_{cl}$ when $cl(a) \neq cl(b)$ follows from Eq. 2.2 and Eq. 2.4, and also uses the facts that $c_{hi}/c_{lo} > c_i/c_{lo}$ and Eq. 2.5 to bound the sum

$$\sum_{i=2, i\neq j}^{k} [\mathbf{e}_i(a) - \mathbf{e}_i(b)] \frac{c_i}{c_{lo}} \left(\frac{\lambda_i}{\lambda_2}\right)^t \leq (k-1)\frac{c_{hi}}{c_{lo}}\delta_{cl} \tag{2.6}$$

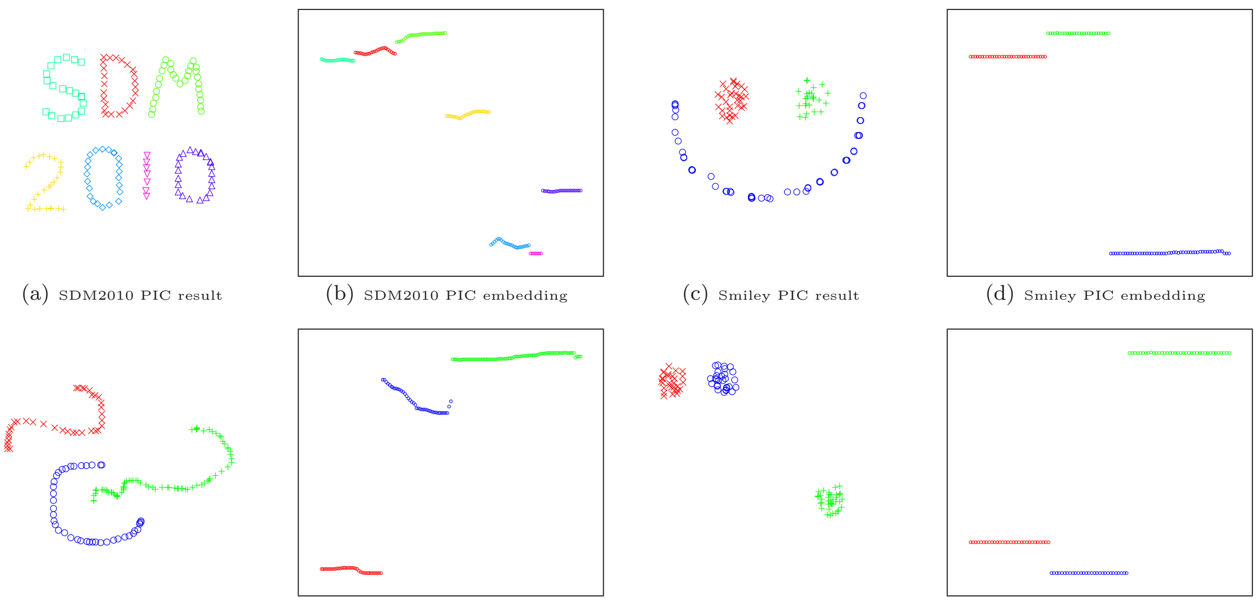
The claim that $spec(a, b) \leq \sqrt{(k-1)\delta_{cl}^2}$ when $cl(a) = cl(b)$a follows from the definition of $spec(a, b)$ and Eq. 2.5. The claim that $R^t signal^t(a, b) \leq (k - 1)\frac{c_{hi}}{c_{lo}}\delta_{cl}$ when $cl(a) = cl(b)$ follows from Eq. 2.2 and Eq.2.3, using a bound similar to that of Eq. 2.6.

Hence, for large enough α and γ_{cl}, small enough t, and small enough δ_{cl}, $signal^t$ is a good approximation of $spec$. Additionally β is small enough, and t is large enough, then $noise^t$ goes to zero faster than $signal^t$. This suggest that for some values of t, $pic^t(a, b)$ will be a good approximation of $spec(a, b)$.

2.5 Clustering Based on PI with Early Stopping These observations suggest that an effective clustering algorithm might run PI for some number of iterations t, stopping when $\mathbf{v}^t$ becomes a useful linear combination of the first k eigenvectors, followed by clustering the eigenvalues of $\mathbf{v^t}$. If we are able to detect and stop PI *after* it has converged within cluster but *before* the entire dataset converges, we will have an approximately piecewise constant vector, where the elements that are in the same cluster have similar values.

Our stopping heuristic is based on the assumption and observation that while the clusters are "locally converging", the rate of convergence also changes rapidly; whereas during the final global convergence, the converge rate appears more stable. This assumption turns out to be well-justified; recall that $\mathbf{v}^t = c_1\lambda_1^t\mathbf{e}_1 + c_2\lambda_2^t\mathbf{e}_2 + ... + c_n\lambda_n^t\mathbf{e}_n$. Then

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



(a) SDM2010 PIC result (b) SDM2010 PIC embedding (c) Smiley PIC result (d) Smiley PIC embedding

(e) Squiggles PIC result (f) Squiggles PIC embedding (g) 3Blobs PIC result (h) 3Blobs PIC embedding

Figure 3: Power iteration clustering results on synthetic datasets.

It is then evident that the convergence rate of PI towards the dominant eigenvector $\mathbf{e}_1$ depends on $(\lambda_i/\lambda_1)^t = (\lambda_i/1)^t$ for the significant terms $2, ..., k$, since their eigenvalues are close to 1 if the clusters are well-separated [11, 18] $(\lambda_i/1)^t \simeq 1$. 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.

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 $\|\boldsymbol{\epsilon}^t\|_\infty \leq \hat{\epsilon}$. In all experiments in this paper, we let $\hat{\epsilon} = \frac{1\times 10^{-5}}{n}$ where n is the number of data instances. The complete algorithm, which we call power iteration clustering (PIC), is shown in Figure 2.

Two issues remain to discuss: the details of the k-means clustering method and the choice of a starting point. In practice, k-means converges very fast but is sensitive to the random initial centers and can get stuck in local minima. To avoid this, we can run k-means several times and record the within-cluster sum of squares (hereafter WCSS) of each run and use the result with the smallest WCSS [1]. Multiple trials of k-means are very cheap especially for PIC, since distances for PIC are calculated in a one-dimension space. In fact, for $k = 2$, the optimal WCSS can be calculated exactly in $O(n^2)$ time—an option for smaller datasets.

The convergence trajectory for PI will be the similar for any initial vector $\mathbf{v^0}$ (other than constant vectors, which are multiples of the actual top eigenvector to which PI will converge.) 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 a *degree vector* $\mathbf{d}$. The degree vector is the dominant eigenvector of W^T, i.e., the steady distribution of a Markov random walk with W^T, rather than W, as the transition matrix. In some sense, then, this vector is "very distant" from the dominant eigenvector of W. The degree vector also 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.

3 Behavior on Synthetic Datasets

Figure 3 shows the clustering results of PIC and the one-dimension PIC embedding on several synthetic datasets. The instances in these datasets lie on a 500-by-500 2-dimension plane and $\mathbf{x}_i$ in a dataset X has the x- and y-coordinates as its two components. The affinity matrix A is defined by $A_{ij} = exp\left(\frac{-||\mathbf{x}_i - \mathbf{x}_j||^2}{2\sigma^2}\right)$ and σ^2 is set

[1] In the experimental sections that follows, we take the most frequent clustering of 50 trials instead of the one with the smallest WCSS—this is for a fair evaluation.

		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 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.

to 200. The colors in the clustering results correspond to the colors in the embedding; for the embedding the value of the elements of $\mathbf{v^t}$ is plotted against its index, and instances in the same cluster are indexed consecutively. Note that clusters that are closer to each other in the original data generally are also embedded near each other (e.g., Figure 3(h)).

4 Accuracy on Real Datasets

We also demonstrate the effectiveness of PIC on several real datasets from various domains. These datasets have known labels and are mainly used for classification tasks:

- **Iris** is a dataset where features are flower petal and sepal measurements from three species of irises, two of which are not linearly separable from each other. It contains 150 instances total, 50 per species.
- **PenDigits01** and **PenDigits17** are hand-written digit datasets, containing the digits "0", "1" and "1", "7", respectively. Each dataset contains 200 instances, 100 per digit, randomly chosen from roughly 1000 instances by 44 writers. PenDigits01 represents an "easy" dataset and PenDigits17 a "difficult" dataset; the latter two digits are more difficult to differentiate in hand writing.
- **PolBooks** is a network of books about US politics. Each book is labeled either "liberal", "conservative", or "neutral", with most books falling in the first two category. Edges between books represent frequent co-purchasing of books by the same buyers.
- **UBMCBlog** is a connected network dataset of 404 liberal and conservative political blogs as described in [8]. The dataset has no textual features and contains only link structure mined from posts of these blogs in a month's time; i.e., blog a is linked to blog b if a has a post containing a hypertext link to a post in b.
- **AGBlog** is a connected network dataset of 1222 liberal and conservative political blogs as described in [1]. The dataset has no textual features and contains only link structure mined mostly from the "sidebars"; i.e., it represents a social network of the blogs.
- **20ng*** datasets are subsets of the 20 newsgroups text dataset. 20ngA contains 100 documents from 2 newsgroups: *misc.forsale* and *soc.religion.christian*. 20ngB adds an additional 100 documents to each group from 20ngA. 20ngC adds 200 documents from *talk.politics.guns* to 20ngB. 20ngD adds 200 documents from *rec.sport.baseball* to 20ngC.

4.1 Evaluation Measures 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).

Purity is the best accuracy obtainable by a clustering, subject to the constraint that all elements in a cluster are assigned the same label. To compute purity, for each cluster C returned by a clustering algorithm, the true labels of instances within C are revealed and counted, and the entire cluster is assigned the label of with the highest count. Purity is then simply the accuracy of this assignment. Purity is simple to understand and fast to calculate.

Normalized mutual information is a information-theoretical measure where the mutual information of the true labeling L and the clustering are normalized by their respective entropies. Formally,

$$NMI(L,C) = \frac{I(L,C)}{[H(L)+H(C)]/2}$$

where $I(L,C)$ is the mutual information of labeling L and clustering C and H is the entropy, given by:

$$I(L,C) = \sum_i \sum_j P(L_i, C_j) log \frac{P(L_i, C_j)}{P(L_i)P(C_j)}$$
$$H(L) = -\sum_i P(L_i) log P(L_i)$$

where $P(L_i)$ is the fraction of instances with ith label and $P(L_i, C_j)$ is the fraction of instances with ith label and lies in jth cluster. Definitions for $P(C_j)$ and $H(C)$ follows.

Rand index compares the labeling and the clustering assignment for every possible pair of instances. The clustering for a particular pair is deemed correct if they have the same labels and are in the same cluster, or if they have different labels and are in different clusters. The Rand index is the number of correctly clustered pairs divided by the total number of possible pairs. This measure is more expensive to calculate as it makes $O(n^2)$ comparisons.

4.2 Experiment Setup For the network datasets (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 Section 3 to avoid having to tune σ^2. For the text datasets (20ng*), word counts are used as feature vectors with only stop words and singleton words removed.

We also compare the results of PIC against those of spectral clustering methods Normalized Cuts (NCut) [15, 11] and the Ng-Jordan-Weiss algorithm (NJW) [12] and present the results in Table 1. In every experiment the k-means algorithm is run 50 times with random starting points and the most frequent cluster assignment is used. This is done so no algorithm would get "lucky" or "unlucky"; the clustering used is the *most likely* clustering. 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.

4.3 Remarks 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 [21, 9, 19]. PIC, on the other hand, avoids having to choose among eigenvectors by embedding a weighted combinations of the most informative eigenvectors (not necessarily k) onto a subspace.

5 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 by definition it stops when $\mathbf{v}^t$ is no longer accelerating towards convergence. Table 2 shows the runtime of PIC and the spectral clustering algorithms on some datasets described in the previous section. Note that PIC converges with relatively few iterations, and for datasets with similar properties (20ng*), fewer iterations seems to be required as dataset size grows. Figure 4(a) and 4(b) plots the runtime data on a log-log scale. NJW is usually takes half the time of NCut because unlike NCut, it uses a symmetric Laplacian matrix.

In addition to the datasets described, we tested PIC on a large gene citation network dataset, which contains genes that are linked to the papers that mentions them, and papers that are linked to the authors that wrote the papers, and the authors. (For further details see [2]). We run PIC on increasing subsamplings of this dataset, with each subsample adding an additional year of publications, from 1983 to 1996. The largest dataset contains roughly 43,000 nodes and 100,000 edges. The runtime is shown in Figure 4(c) and is linear to the number of nodes.

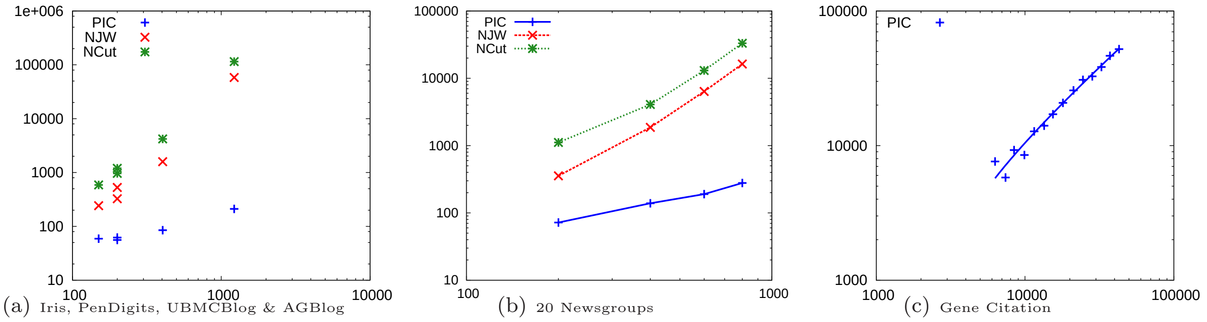
For these experiments, all algorithms are implements in Java, and for the spectral clustering methods the eigenvectors are calculated via eigenvalue decomposition. Experiments are ran on a single Linux machine with a Intel 1.86GHz CPU and 4GB RAM.

6 Related Work

Spectral clustering began with the discovery of the correlation between the eigenvalues of the Laplacian matrix and the connectivity of a graph [5]. Later it was introduced to the machine learning community through Ratio Cut [14] and Normalized Cuts [15, 11]. Since then it has sparked much interest and lead to further analyses and modifications [12, 18]. Typically, a spectral clustering algorithm defines a Laplacian matrix, such as the

Dataset	Size	NCut Runtime	NJW Runtime	PIC Runtime	PIC Iterations
Iris	150	640	336	113	6
PenDigits01	200	1124	511	121	6
PenDigits17	200	1343	602	345	5
PolBooks	105	719	273	156	29
AGBlog	1222	117448	60072	378	34
20ngA	200	1268	424	185	15
20ngB	400	4520	1863	257	13
20ngC	600	14109	8252	338	13
20ngD	800	33416	17779	413	12

Table 2: Runtime comparison (in milliseconds) of PIC and spectral clustering algorithms on several real datasets.



(a) Iris, PenDigits, UBMCBlog & AGBlog (b) 20 Newsgroups (c) Gene Citation

Figure 4: Comparison of algorithm runtime on different datasets. The number of instances is plotted against the runtime in milliseconds on a log-log scale.

row-normalized one in [15] or the symmetrically normalized one in [12]. Then the first k smallest eigenvectors, deemed the most informative, are used to embed the data onto a k-dimensional space on which a k-means algorithm is used to obtain the final clusters. However, these "classical" spectral clustering approaches have two shortcomings. First, finding eigenvectors of a matrix takes $O(n^3)$ in general where n is the number of data points. Which recent work has tried to address this with multilevel [17] and data preprocessing [6, 20] techniques. Second, simply using the first k eigenvectors seems to fail on many real datasets because they often turns out to be uninformative, especially in the presence of noise. This prompted work on selecting "good" eigenvectors and dealing with noisy data [21, 9, 19].

PIC is related to spectral clustering in that eigenvectors play an important role in a low-dimensional embedding of data. PIC also uses k-means on the one-dimension embedding to produce the final clusters. PIC's main advantage is its simplicity, and low computational cost, as it is not necessary to find several eigenvectors. In fact, the experiments show that it is *not* necessary to find *any* eigenvector in order to find a low-dimensional embedding for clustering—the embedding just needs to be a good linear combination of the top eigenvectors. In this respect, PIC is very different from multilevel spectral clustering [17] or approaches that use data sampling techniques [6, 20]. These methods usually requires a preprocessing step where sample data points are chosen as "representatives", and then spectral clustering is done on this sample instead of the entire dataset. The final clustering result is obtained by propagating the clustering of these representatives to the other data points. PIC is a fast method without data sampling; however, the techniques are complementary, and we may be able to apply these techniques to further speed up PIC in future work.

Another recent graph clustering approach that has shown substantial speed improvement over spectral clustering methods is *multilevel kernel k-means* [4]. In [4], 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; [4] 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

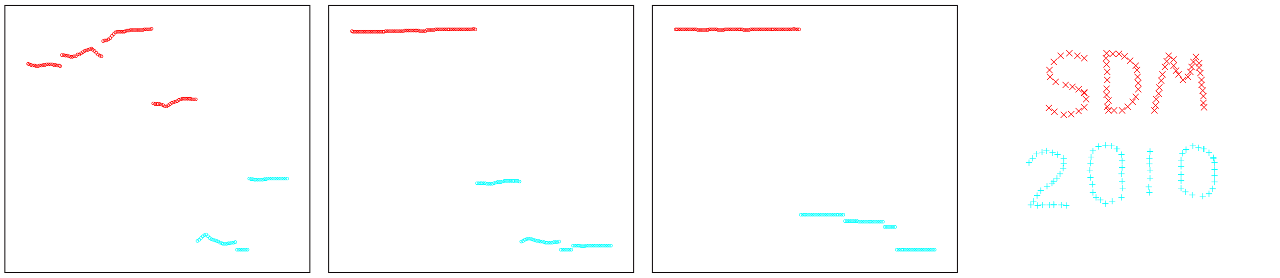

(a) $t = 500$ (b) $t = 1000$ (c) $t = 2000$ (d) SDM2010 two-cluster PIC result

Figure 5: "SDM" and "2010" merging into two clusters

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. In comparison, while both are extremely fast compared to traditional spectral clustering methods, PIC is a much simpler algorithm, and like spectral clustering methods, it is much less susceptible to being trapped in local minima.

7 Conclusion and Future Work

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 intuitive (can be understood in terms of PI convergence, random walks, and iterative averaging), easy to implement (matrix-vector multiplications), and efficient both in terms of time and space. Based on the analysis of the convergence behavior of power iteration, we propose and proof the conditions under which PIC would do well. 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 popular spectral clustering methods; they also show that PIC is very fast in practice, without using sampling techniques, and can scale up to very large datasets.

We believe this work is only an introduction to clustering with power iteration. Here we propose a few research directions based on properties of PIC that would welcome further development and study:

Hierarchical clustering in Krylov subspace Krylov subspace is the linear subspace spanned by the vectors $\mathbf{v}^0, \mathbf{v}^1, ..., \mathbf{v}^t$ that are generated by PI at each iteration. Krylov subspace is used in fast methods for approximating eigenvectors such as Lanczos iteration and Arnoldi iteration. The current work uses only one vector from the set of vectors that span the subspace, but this needs not to be the case. Due to the convergence characteristic of PI, two clusters that are "closer" to each other should converge to the same value (merge into a super-cluster) before they do with a third, more "distant" cluster. Figure 5 shows what happens to the "SDM 2010" dataset from Figure 3 if we keep on iterating—"SDM" and "2010" eventually become two clusters, even though "2" is closer to "S" then to the last "0".

Choosing k automatically from embedding Since PIC produces a low-dimensional embedding of the data *independent of k* (unlike most spectral clustering methods), methods that determines the k in k-means automatically, such as the *G-means* algorithm [7], can be used directly on the embedding.

Speeding up PI Modifications of PI that speed up its convergence (e.g., the *Aitken's delta-squared process*) and other iterative methods for tracking the Krylov subspace of a matrix can potentially be used to speed up the calculation of the PIC embedding.

References

[1] L. Adamic and N. Glance. The political blogosphere and the 2004 u.s. election: Divided they blog. In *Proceedings of the WWW-2005 Workshop on the Weblogging Ecosystem*, 2005.

[2] A. Arnold and W. W. Cohen. Information extraction as link prediction: Using curated citation networks to improve gene detection. In *Proceedings of the AAAI Conference on Weblogs and Social Media*, 2009.

[3] S. Baluja, R. Seth, D. Sivakumar, Y. Jing, J. Yagnik, S. Kumar, D. Ravichandran, and M. Aly. Video suggestion and discovery for YouTube: Taking random walks through the view graph. In *Proceeding of the 17th International Conference on World Wide Web*, 2008.

[4] I. S. Dhillon, Y. Guan, and B. Kulis. Weighted graph cuts without eigenvectors: A multilevel approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(11):1944–1957, 2007.

[5] M. Fiedler. Algebraic connectivity of graphs. *Czechoslovak Mathematical Journal*, 23(98):298–305, 1973.

[6] C. Fowlkes, S. Belongie, F. Chung, and J. Malik. Spectral grouping using the Nyström Method. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2004.

[7] G. Hamerly and C. Elkan. Learning the k in k-means. In *Advances in Neural Information Processing Systems 17*, 2003.

[8] A. Kale, A. Karandikar, P. Kolari, A. Java, T. Finin, and A. Joshi. Modeling trust and influence in the blogosphere using link polarity. In *Proceedings of the International Conference on Weblogs and Social Media (ICWSM 2007)*, 2007.

[9] Z. Li, J. Liu, S. Chen, and X. Tang. Noise robust spectral clustering. In *IEEE 11th International Conference on Computer Vision*, 2007.

[10] S. A. Macskassy and F. Provost. Classification in networked data: A toolkit and a univariate case study. *The Journal of Machine Learning Research*, 8:935–983, 2007.

[11] M. Meilă and J. Shi. A random walks view of spectral segmentation. In *IEEE International Conference on Artificial Intelligence and Statistics*, 2001.

[12] A. Y. Ng, M. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In *Advances in Neural Information Processing Systems 14*, 2002.

[13] L. Page, S. Brin, R. Motwani, and T. Winograd. The PageRank citation ranking: Bringing order to the web. Technical report, Stanford Digital Library Technologies Project, 1998.

[14] T. Roxborough and A. Sen. Graph clustering using multiway ratio cut. In *Proceedings of Graph Drawing*, pages 291–296, 1997.

[15] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

[16] P. P. Talukdar, J. Reisinger, M. Paşca, D. Ravichandran, R. Bhagat, and F. Pereira. Weakly-supervised acquisition of labeled class instances using graph random walks. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, 2008.

[17] D. Tolliver, R. T. Collins, and S. Baker. Multilevel spectral partitioning for efficient image segmentation and tracking. In *Proceedings of the Seventh IEEE Workshops on Application of Computer Vision*, 2005.

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

[19] T. Xiang and S. Gong. Spectral clustering with eigenvector selection. *Pattern Recognition*, 41(3):1012–1029, 2008.

[20] D. Yan, L. Huang, and M. I. Jordan. Fast approximate spectral clustering. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2009.

[21] L. Zelnik-Manor and P. Perona. Self-tuning spectral clustering. In *Advances in Neural Information Processing Systems 17*, 2005.

[22] X. Zhu, Z. Ghahramani, and J. Lafferty. Semi-supervised learning using Gaussian fields and harmonic functions. In *The 20th International Conference on Machine Learning*, 2003.