On a Recursive Spectral Algorithm for Clustering from Pairwise Similarities

David Cheng
MIT
drcheng@mit.edu

Ravi Kannan
Yale University
kannan@cs.yale.edu

Santosh Vempala
MIT
vempala@math.mit.edu

Grant Wang
MIT
gjw@theory.lcs.mit.edu

Abstract

We present a practical implementation of the clustering algorithm described in [20]. The clustering algorithm is given either an implicit or explicit representation of the pairwise similarities between n objects and produces a complete hierarchical clustering of the n objects. The implementation runs in $O(M \log n)$ time per cluster where M is the number of non-zero entries in the "document-term" matrix, a common implicit representation of similarities between data objects. We perform a thorough experimental evaluation of the algorithm in practice. The results show that the algorithm is better or competitive with existing clustering algorithms (e.g. k-means [21], ROCK [18], p-QR [37]).

1 Introduction

The last decade has witnessed an unprecedented explosion in the volume of readily accessible data. The challenge for computer scientists is to find methods that can locate relevant information and organize it in an intelligible way. This is different from the classical database problem in at least two ways: first, there may neither be the time nor (in the long term) the computer memory to store and structure all the data (e.g. the world-wide web or a portion of it) in a central location. Second, one would like to find interesting patterns in the data without knowing in advance what exactly one is looking for.

Clustering refers to the process of classifying a set of data objects into groups so that each group consists of similar objects. The classification could either be flat (i.e. a partition of the dataset) or hierarchical. Clustering has been proposed as a method to aid information retrieval in many contexts [15]. Document clustering can help improve precision and recall [16], in finding nearest neighbors [12], to generate a hierarchical taxonomy efficiently [11], and more recently in organizing the results of a web search [36]. It has also been used to learn (or fit) mixture models to datasets [19] and for image segmentation [30].

In this paper, we will assume that the data objects are presented in a way which makes it easy to compute the similarity of any pair of objects. For example, the coordinates of points in space, document-term matrices, feature vectors etc. (we could also be presented with just the pairwise similarities).

There are many methods that have been proposed in the literature, especially in the context of document clustering [39, 11, 13]. Broadly speaking these fall into agglomerative (i.e. bottom-up) methods and partitioning methods (i.e. top-down). Many of them take advantage of the special structure of document-term methods and rely on a careful choice of the similarity function. On the other hand, a general-purpose heuristic known as k-means often performs quite well, and it has been argued that it outperforms many other methods [21, 29].

The class of algorithms known as *spectral* methods seem to be applicable in a variety of clustering contexts. This has been investigated in [7, 35, 27, 22, 17]. Roughly speaking, a spectral algorithm uses the eigenvalues and eigenvectors of the similarity matrix to find a clustering. Our principal motivation is the spectral algorithm analyzed in [20]. There, the authors develop a measure of a good clustering based on the graph of pairwise similarities (this measure cannot be universal, but it seems to be quite general). They prove that a simple spectral algorithm whose input is a similarity matrix has reasonable worst-case guarantees with respect to this measure. However the running time for a dataset with n objects (e.g. documents) could be $O(n^4)$. Further, there is no experimental evidence provided and the theoretical guarantees involve large constant factors.

In this paper, we describe an efficient implementation of the algorithm in [20] and perform a thorough experimental evaluation. The main advantage of the implementation (described in Section 2) is that it maintains the sparsity of input matrices. In Section 2.3.6 we show that if a document term matrix has M non-zero entries, then the clustering algorithm uses only $O(M)$ space overall and takes $O(M \log n)$ time per cluster that it produces. The algorithm can also be used to generate a complete hierarchical classification of the documents (i.e. a tree whose leaves are the documents) in time $O(Mn \log n)$ time. On commodity hardware, the algorithm took 20 minutes for datasets with $n = 20,000$ documents and and $M = 500,000$ terms (see figures 1(a) and 1(b)). Section 2.3.6 also gives a more comprehensive timing evaluation.

Our main contribution is an experimental evaluation of the algorithm on document-term matrices of standard datasets, randomly generated data, and two-dimensional pictures. As reported in Section 3, in all cases the algorithm did quite well, and for the document clustering, it consistently outperformed many recent methods (the only one that sometimes did better was the heuristic known as bisecting k-means [29, 32]). The performance was evaluated using the F-measure, entropy, and accuracy.

In Section 4, we discuss these results and offer new problems to consider.

2 The Algorithm

2.1 Setup

The input to the algorithm is a matrix A whose rows are the data objects and columns are the features. Although the data could arise in many contexts, we will refer to the input matrix as the document-term matrix. We denote the ith document, a row vector in A, by $A_{(i)}$.

The similarity of two documents is defined as the dot product of their term vectors: $A_{(i)} \cdot A_{(j)}$. The similarity matrix is therefore just AA^T. We typically make each document a unit vector so the similarity between two documents becomes the cosine of the angle between their vectors. We can also think of the similarity matrix as a complete graph where the vertex set is the set of documents with edge weights as the similarity between two vertices (documents). This is the similarity graph.

2.2 Brief Description

The algorithm constructs a hierarchical clustering of the vertices of the similarity graph by recursively dividing the graph into two pieces through a cut $(S, V \setminus S)$ of the vertices. The conductance of a *cut* $(S, V \setminus S)$ is

$$\phi(S, V \setminus S) = \frac{c(S, V \setminus S)}{\min(c(S), c(V \setminus S))}$$

where $c(A, B) = \sum_{i \in A, j \in B} A_{(i)} \cdot A_{(j)}$ and $c(A) = c(A, V)$. Informally, the less edges crossing the cut and the more even the size of S and $V \setminus S$ are, the lower the conductance and the "better" the cut is for the graph. The conductance of a *graph* is the minimum conductance achieved by any cut, which we denote C^*.

The algorithm works by finding a cut C that is not much worse than the cut C^* in terms of conductance. It then partitions the graph using C, and recurses on the subparts. To find the cut C, we group vertices according to their ordering based on the second eigenvector of the similarity matrix AA^T. Here, by *not much worse* we are being imprecise; theoretical guarantees on the quality of the cut can be found in [20]. The algorithm we describe here is an efficient, practical implementation of the algorithm in [20] for which the same theoretical guarantees hold. The algorithm is given below in the box.

Algorithm.
Input: An $n \times m$ matrix A.
Output: A tree whose leaves are the rows of A.

1. **Initialize**
   (a) Let $P \in \mathbb{R}^{n \times n}$ be a zero matrix. Set $\hat{A} := (A \mid P)$.
   (b) Let $R^2 \in \mathbb{R}^{n \times n}$ be a diagonal matrix whose diagonal entries are the row sums of $\hat{A}\hat{A}^T$.
2. **Compute Singular Vector**
   (a) Compute the second largest right singular vector v' of the matrix $\hat{A}^T R^{-1}$.
   (b) Let $v = R^{-1}v'$.
3. **Cut**
   (a) Sort v so that $v_i \le v_{i+1}$.
   (b) Find the value t that minimizes the conductance of the cut:
   $$(S, T) = (\{v_1, \ldots, v_t\}, \{v_{t+1}, \ldots, v_n\})$$
   (c) Let $\hat{A}_S$, $\hat{A}_T$ be the submatrices of A whose rows are those in S, T.
4. **Normalize**
   Adjust the self-similarities by setting
   $$P_{ii}^2 := P_{ii}^2 + \begin{cases} \sum_{j \in T} A_{(i)} \cdot A_{(j)} & \text{if } i \in S, \\ \sum_{j \in S} A_{(i)} \cdot A_{(j)} & \text{if } i \in T \end{cases}$$
5. **Recurse**
   Recurse (steps 2-4) on the submatrices $\hat{A}_S$ and $\hat{A}_T$.

Note that the recursion tree that results from the algorithm is a hierarchical clustering of the document set. In certain situations, the desired number of clusters is given to us. Here, we can stop the recursion at a certain level when the number of clusters is equal to the desired number.

2.3 Details

In practice, the representation of documents as row vectors are sparse, i.e. there are many more zeros than non-zero entries. Our algorithm takes particular care to work with the similarities only indirectly through the sparse document-term matrix.

Any vector or matrix that the algorithm uses is stored using standard data structures for sparse representation. These allow constant time operations for retrieving a particular value and initialization of the entire vector or matrix to a specific value.

Next, we describe the exact details of each the five steps of the algorithm.

2.3.1 Initialize

Step (a) of initialization is trivial and takes constant time.

To compute R^2 in step (b), we must compute each row sum of the similarity matrix AA^T. Let ρ_i be the sum of the ith row of AA^T. Observe that

$$\rho_i = \sum_{j=1}^{n} A_{(i)} \cdot A_{(j)} = \sum_{j=1}^{n}\sum_{k=1}^{m} A_{ik}A_{jk} = \sum_{k=1}^{m} A_{ik}\left(\sum_{j=1}^{n} A_{jk}\right)$$

Because $\sum_{j=1}^{n} A_{jk}$ does not depend on i, we can compute $u = \sum_{i=1}^{n} A_{(i)}$, and compute each $\rho_i = A_{(i)} \cdot u$ for each row. The computation of u makes exactly one sweep over the non-zero entries of A. Each dot product to compute ρ_i takes time proportional to the non-zeros of $A_{(i)}$. As a result, the total running time is $\Theta(M)$. To store v and each ρ_i, we require $\Theta(n+m)$ space.

2.3.2 Compute Singular Vector

The algorithm described in [20] uses the second largest right eigenvector of the normalized similarity matrix to compute a good cut. To compute this vector, we compute the second largest *singular* vector v of the matrix $\hat{A}^T R^{-1}$. Note that v is the second largest eigenvector of $Q = (R^{-1}\hat{A})(R^{-1}\hat{A})^T = R^{-1}\hat{A}\hat{A}^T R^{-1}$. The key property is that the eigenvalues of Q and B are related – it is easy to check that if $Qv = \lambda v$, then $BR^{-1}v = \lambda R^{-1}v$.

Note that Q is a symmetric matrix; as such, we can compute its second largest eigenvector using the power method, an iterative algorithm whose main computation is a matrix-vector multiplcation. Since $R^{-1}\hat{A}$ is sparse and $Q = (R^{-1}\hat{A})(R^{-1}\hat{A})^T$, computing Qy for some vector y requires only two sparse matrix-vector multiplications, so we maintain sparsity to compute the second eigenvector. The power method is described below:

Power Method

1. Construct a random n-dimensional vector v
2. Make v orthogonal to ρR^{-1}
3. Compute $\lambda = |v|$ and set $v = \frac{v}{|v|}$
4. Set $v = Qv$ and repeat steps 2 to 4 $O(\log n)$ times

Making a random starting vector may be done in many ways; we simply pick each component v_i from the uniform distribution over $[-\frac{1}{2}, \frac{1}{2}]$.

Step 2 ensures that the vector we compute is the second largest eigenvector. Note that $\rho R^{-1} Q = \rho R^{-1}$ so ρR^{-1} is a left eigenvector with eigenvalue 1. A basic fact of linear algebra is that the second largest eigenvector is orthogonal to the largest eigenvector. To keep v orthogonal to a vector x, we arbitrarily choose to change the first component, setting $v_1 = -\frac{1}{x_1}\sum_{i=2}^{n} v_i x_i$. Note that with $x = \rho R^{-1}$, x_1 is never 0.

The power method takes $\Theta(\log n)$ iterations to converge; the following lemma and corollary proves this.

Lemma 1. *Let $A \in \mathbb{R}^{n\times n}$ be a symmetric matrix, and let $v \in \mathbb{R}^n$ be chosen uniformly at random from the unit n-dimensional sphere. Then for any positive integer k, the following holds with probability at least $1-\delta$:*

$$\frac{||A^{k+1}v||}{||A^k v||} \geq \left(n \ln \frac{1}{\delta}\right)^{-\frac{1}{2k}} ||A||_2.$$

Proof. Since A is symmetric, we can write

$$A = \sum_{i=1}^{n} \lambda_i u_i u_i^T,$$

where the λ_i 's are the eigenvalues of A arranged in the order - $|\lambda_1| \geq |\lambda_2| \ldots |\lambda_n|$ and the u_i are the corresponding eigenvectors. Express v in this basis as $v = \sum_i \alpha_i u_i$, where $\sum_i \alpha_i^2 = 1$. Since, v is random, we have that with probability at least $1-\delta$, $\alpha_1^2 \geq 1/(n\ln(1/\delta))$. Then, using Hölder's inequality (which says that for any $p, q > 0$ satisfying $(1/p)+(1/q) = 1$ and any $a, b \in \mathbb{R}^n$, we have $\sum_i a_i b_i \leq (\sum_i |a_i|^p)^{1/p} (\sum_i |b_i|^q)^{1/q}$, we have

$$||A^k v||^2 = |\sum_i \alpha_i \lambda_i^k u_i|^2 = \sum_i \alpha_i^2 \lambda_i^{2k} \leq \left(\sum \alpha_i^2 \lambda_i^{2k+2}\right)^{k/(k+1)}$$

where the last inequality holds using Hölder with $p = 1+(1/k) \quad q = k+1 \quad a_i = \alpha_i^{2k/(k+1)} \lambda_i^{2k} \quad b_i = \alpha_i^{2/(k+1)}$. Note that:

$$\left(\sum \alpha_i^2 \lambda_i^{2k+2}\right)^{k/(k+1)} \leq \left(\sum \alpha_i^2 \lambda_i^{2k+2}\right) / \lambda_1^2 \alpha_1^{2/(k+1)}$$

from which the lemma follows. □

Corollary 1. *If $k > \frac{1}{2\epsilon} \ln(n \ln(\frac{1}{\delta}))$, then we have:*

$$\frac{||A^{k+1} v||}{||A^k v||} \geq (1+\epsilon)\lambda_1$$

.

The total running time is $\Theta(M \log n)$, since each iteration requires two sparse matrix-vector multiplications. We require $\Theta(n+m)$ space for the vector v and matrix R.

2.3.3 Cut

Step (a) requires us to sort the coordinates of v so that $v_i \leq v_{i+1}$. This can be done in $\Theta(n \log n)$ time.

In step (b), we choose the cut C of the $n-1$ cuts $(\{1, \ldots, t\}, \{t+1, \ldots, n\})$ which has the smallest conductance. The conductance of the cut C is not much worse than the conductance of the graph (i.e. the smallest conductance cut in the graph); see [20] for a more quantitative statement.

Recall that

$$\phi(\{1, \ldots, i\}, \{i+1, \ldots, n\}) = \frac{\sum_{k=1}^{i} \sum_{j=i+1}^{n} A_{(k)} \cdot A_{(j)}}{\min(\sum_{k=1}^{i} \rho_k, \sum_{k=i+1}^{n} \rho_k)}.$$

Let the numerator of this expression be u_i and the denominator l_i.

We can compute each u_i and each l_i using only two passes through A. Note that each numerator value follows from its preceding value:

$$u_i = u_{i-1} - \sum_{k=1}^{i-1} A_{(k)} \cdot A_{(i)} + \sum_{j=i+1}^{n} A_{(i)} \cdot A_{(j)}$$

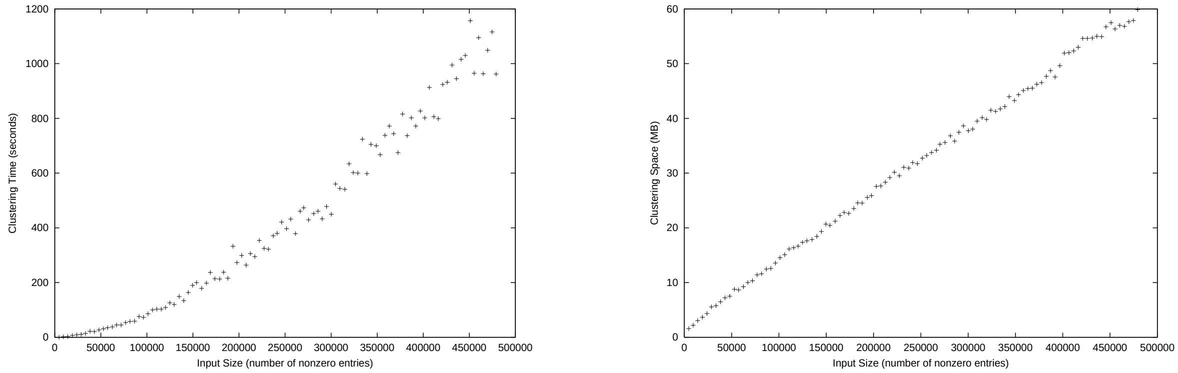
Since $\sum_{k=1}^{i=1} A_{(k)} \cdot A_{(i)} = A_{(i)} \cdot (\sum_{j=1}^{i-1} A_{(j)})$ and $\sum_{j=i+1}^{n} A_{(i)} \cdot A_{(j)} = A_{(i)} \cdot (\sum_{j=i+1}^{j=n} A_{(j)})$, we can maintain both partial sums to compute u_i from u_{i-1}. Initializing the partial sums requires one pass through A; the computation of u requires another pass through A to update the partial sums. The denominator can be computed in a similar fashion. The space required to store u, l and ϕ amounts to $\Theta(n+m)$.

2.3.4 Normalize

To recurse on $\hat{A}_S$ and $\hat{A}_T$, the submatrices of $\hat{A}$ that are defined by the cut (S, T), we must normalize the row sums of their similarity matrices. We can do this by modifying the matrix P. In particular, we add to each diagonal element in P some value P_{ii} so that $\sum_j B_{ij} = 1$ for j in the same set as i. If $i \in S$, then $P_{ii}^2 = \sum_{j \in T} A_{(i)} \cdot A_{(j)} = A_{(i)} \cdot (\sum_{j \in T} A_{(j)})$ and if $i \in T$, then $P_{ii}^2 = \sum_{j \in S} c_{ij} = A_{(i)} \cdot (\sum_{j \in S} A_{(j)})$. To compute the P_{ii} values, we only need two passes through the matrix A.

2.3.5 Recurse

Executing steps 2-4 on $\hat{A}_S$ and $\hat{A}_T$ will result in cuts of S and T. If we recurse until a submatrix contains only one row, the recursion tree is a complete hierarchical clustering of the rows of A.



(a) Time as a function of input size

(b) Space as a function of input size

Figure 1: Performance of clustering algorithm in experiments

2.3.6 Time and Space requirements

Theoretically, the worst-case time to compute a complete hierarchical clustering of the rows of A is $O(Mn \log n)$, as it takes $O(M \log n)$ time to execute steps 2-4.

In practice, our algorithm seems to perform quite well. Figures 1(a) and 1(b) show the results of a performance experiment. In this experiment, we chose N random articles from each newsgroup in the 20 newsgroups data set [1] and ran our clustering algorithm, finding a complete hierarchical clustering. Initially, $N = 10$, and was increased in increments of 10 until $N = 1,000$. For the timing graph, we recorded the number of seconds needed to find such a complete hierarchical clustering. For the space experiment, we recorded the amount of RAM in megabytes that the clustering algorithm used.

Note that even in the worst case, when we chose 1,000 documents from each of the newsgroups (for a total of 20,000 news articles), we were able to compute a complete complete hierarchical clustering in 20 minutes.

3 Clustering Experiments

We tested our clustering algorithm on four types of datasets: standard document-term matrices, categorical clustering data, synthetic data from mixture models and 2-dimensional pictures. We also mention *Eigencluster*, a web search engine implementation that uses the clustering algorithms we developed in this paper.

In each of the datasets, there was a well-defined notion of the right clustering. The performance of our algorithm was evaluated using three measures: F-measure, entropy, and accuracy. Each of these measures is defined with respect to the correct classification. We explain each of these measures below. Let the correct clustering be $C_1 \ldots C_k$, with $C = \cup_i C_i$ and let the nodes of the complete hierarchical clustering found by the spectral algorithm be $\hat{C}_1 \ldots \hat{C}_l$. Each $\hat{C}_i$ defines a cluster – the nodes in the tree below it; $\hat{C}_1$ is the entire set ($\hat{C}_1 = \cup_j C_j$).

1. **F-measure**: For each correct cluster C_i, the F-measure of that cluster is:

$$F(i) = \max_{j=1}^{l} \frac{2P_j R_j}{P_j + R_j}$$

where:

$$P_j = \frac{|C_i \cap \hat{C}_j|}{|\hat{C}_i|}$$
$$R_j = \frac{|C_i \cap \hat{C}_j|}{|C_i|}$$

The F-measure of the clustering is defined as:

$$\sum_{i=1}^{k} F(i) \cdot \frac{|C_i|}{|C|}$$

The F-measure score is in the range $[0, 1]$ and a **higher** F-measure score implies a better clustering. For a more in-depth introduction and justification to the F-measure, see e.g. [31, 21, 29, 10, 24, 34].

2. **Entropy**: For each $\hat{C}_j$, we define the entropy of $\hat{C}_j$ as:

$$E(\hat{C}_j) = \sum_{i=1}^{k} -\left(\frac{|C_i \cap \hat{C}_j|}{|\hat{C}_j|}\right) \log\left(\frac{|C_i \cap \hat{C}_j|}{|\hat{C}_j|}\right)$$

The entropy of a cluster is a measure of the disorder within the cluster. As such, a **lower** entropy score for a clustering implies that the clustering is better. The best possible entropy score is 0, while the worst is 1. Entropy was first introduced in [26] and has been used as a measure of clustering in [11, 17, 9, 14].

The entropy of a k-clustering $D_1 \ldots D_k$ is the weighted sum of the entropies of the clusters:

$$\sum_{i=1}^{k} E(D_i) \cdot \frac{|D_i|}{|C|}$$

The entropy of our complete hierarchical clustering $\{\hat{C}_1 \ldots \hat{C}_l\}$ is the minimum entropy of any choice of k nodes that partition C.

3. **Accuracy**: The accuracy of $\hat{C}_j$ is:

$$A(\hat{C}_j) = \max_{i=1}^{k} \frac{|C_i \cap \hat{C}_j|}{|\hat{C}_j|}.$$

As before, the accuracy of a k-clustering $D_1 \ldots D_k$ is the weighted sum of accuracies. The accuracy of our complete hierarchical clustering is the maximum accuracy of any choice of k nodes that partition C. Note that the range of an accuracy score is between 0 and 1; the **higher** the accuracy score, the better.

Accuracy, which has been used as a measure of performance in supervised learning, has also been used in clustering (see [38, 8, 28]).

3.1 Document-term matrices

These experiments involved common datasets used in prior experimentation. In all of the cases, we either do better or compare favorably with known results.

3.1.1 Reuters

Many clustering algorithms are tested on the Reuters dataset [4], a corpus of $8,654$ news articles that have been classified into 135 distinct news topics.

We ran two experiments on this dataset. In the first one, we attempted to cluster all $8,654$ news articles into 135 distinct new topics. Each of the news articles was kept fully intact, so the clustering algorithm was just given the raw document-term matrix. This same experiment was performed in [10, 21]. The second experiment was the same experiment conducted in [24]. Here, the clustering algorithm was only given the 6,575 news articles from 10 of the 135 largest news topics. The results of our algorithm, along with the prior experimental results can be found in Table 1. The measure of performance in the table is the F-measure of the clustering. Our algorithm outperforms the results of prior experiments.

Table 1: Reuters dataset (F-measure)

	Spectral	BEX02	LA99	NJM01
8,654 articles	.713	.57	.63	N/A
6,575 articles	.733	N/A	N/A	.665

3.1.2 Web pages

Certainly one use for clustering is to attempt to cluster webpages into semantic groups (see section 3.5 for more on this). An experiment was run in [11] in which 185 pages were selected from the web that fall in 10 distinct categories (see [3] for the dataset).

In a series of 11 experiments where the term vector for each webpage is constructed in a slightly different manner, they ran their algorithm and measured the entropy of the clustering found by their algorithm. We ran our spectral algorithm on the exact same tests; a comparison of results can be found in Table 2, and the exact details of the 11 experiments can be found in [11]. In 7 of the 11 experiments, our algorithm outperforms their algorithm.

Table 2: Webpage dataset (Entropy)

	Spectral	B97
J1	.77	.69
J2	.81	1.12
J3	.54	.85
J4	1.12	1.10
J5	.81	.74
J6	.81	.83
J7	.63	.90
J8	.84	.96
J9	.65	1.07
J10	1.77	1.17
J11	.90	1.05

3.1.3 SMART Dataset

The SMART dataset is a set of abstracts originating from Cornell University [5]. The makeup of the abstracts is as follows: 1033 medical abstracts (Medline), 1400 aeronautical systems abstracts (Cranfield), and 1460 information retrieval abstracts (Cisi).

We performed an identical set of experiments as [17]. One type of experiment involved clustering the document set of two distinct sets of abstracts (e.g. Medline and Cranfield), while the other experiment involved clustering the document set of all the abstracts. In Table 3, the results of our experiment and the results of [17] are listed. Here, MedCran is the combination of the Medline and Cranfield datasets (and likewise for MedCisi, CisiCran). Classic3 is the combination of all 3 datasets. The measure of performance in this table is entropy; we compare competitively.

3.1.4 CLUTO Datasets

CLUTO is a software package for clustering datasets from a variety of different areas [40]. To evaluate the performance of the software package, a series of tests on document-term matrices was run on the software. The document-term matrices come from different sources – San Jose Mercury Times, LA Times, Reuters, TREC, etc, each of which is divided into its own categories (see [2] for the exact dataset).

Table 3: SMART dataset (Entropy)

	Spectral	Dhillon 2001
MedCran	.032	.026
MedCisi	.092	.152
CisiCran	.045	.046
Classic3	.090	.089

Table 4: CLUTO dataset (F-measure)

	Spectral	ZK02
fbis	0.579849	0.558070
hitech	0.540306	0.465919
k1a	0.573836	0.462622
k1b	0.844459	0.561328
la1	0.654640	0.551499
la2	0.561199	0.504116
re0	0.596008	0.409478
re1	0.582314	0.437812
reviews	0.732537	0.706416
tr31	0.838855	0.591150
tr41	0.743863	0.576615
wap	0.586106	0.441109

The experiments we performed were exactly those in [40], where they attempt to cluster each collection. In particular, we ran our clustering algorithm on the same collections, and additionally we ran their software package on these collections.[1] The details of the results can be found in Table 4. Here, the quality of the clustering is the F-measure. Note that the spectral algorithm performed better on every dataset.

3.1.5 20 Newsgroups

The 20 newsgroups resource [1] is a corpus of about 20,000 articles that come from 20 specific Usenet newsgroups (see Table 5). We performed the same experiments as [37]. Each experiment involved choosing 50 random newsgroup articles from a specific subset of the 20 newsgroups. We stripped the Usenet headers from each newsgroup article, and used the first 2000 terms for each article (all but a few articles were under 2000 terms).[2] Otherwise, the article was left unchanged. The clustering algorithm was run on the corresponding document-term matrix.

The results in table 6 are experiments in which the clustering algorithm is given 100 documents – 50 random documents each from two distinct newsgroups (e.g. `alt.atheism` and `comp.graphics`). This was the same set of experiments conducted for clustering two newsgroups done by [37]. Note that we perform better than p-QR, their proposed algorithm on all but one of the experiments for two-way clustering. We also outperform K-means and a variation of the K-means algorithm, p-Kmeans. In each of these experiments, the measure of performance was accuracy. Since the experiment involves choosing 50 random newsgroup articles, the experiment was run 100 times and the mean and standard deviation of the results were recorded. It should be noted that [37] also performed experiments in which the clustering algorithm was given articles from more than 2 newsgroups and asked to cluster them. Here, we do not perform as well; reasons for this are discussed in section 3.6.

[1] The data for ZK02 in the table is much worse than the data they give in their paper. There are several reasons why this could be the case. For example, preprocessing steps not explicitly mentioned in their paper could have been used.

[2] We used the BOW toolkit for processing the newsgroup data. More information on the BOW toolkit can be found on `http://www-2.cs.cmu.edu/~mccallum/bow`

Table 5: Newsgroups and Labellings

NG1	`alt.atheism`	**NG11**	`rec.sport.hockey`
NG2	`comp.graphics`	**NG12**	`sci.crypt`
NG3	`comp.os.ms-windows.misc`	**NG13**	`sci.electronics`
NG4	`comp.sys.ibm.pc.hardware`	**NG14**	`sci.med`
NG5	`comp.sys.mac.hardware`	**NG15**	`sci.space`
NG6	`comp.windows.x`	**NG16**	`soc.religion.christian`
NG7	`misc.forsale`	**NG17**	`talk.politics.guns`
NG8	`rec.autos`	**NG18**	`talk.politics.mideast`
NG9	`rec.motorcycles`	**NG19**	`talk.politics.misc`
NG10	`rec.sport.baseball`	**NG20**	`talk.religion.misc`

Table 6: 20 Newsgroups dataset (Accuracy)

	Spectral	p-QR	p-Kmeans	K-means
NG1/NG2	93.64 ± 2.55%	89.29 ± 7.51%	89.62 ± 6.90%	76.25 ± 13.06%
NG2/NG3	81.86 ± 6.30%	62.37 ± 8.39%	63.84 ± 8.74%	61.62 ± 8.03%
NG8/NG9	80.30 ± 8.36%	75.88 ± 8.88%	77.64 ± 9.00%	65.65 ± 9.26%
NG10/NG11	70.12 ± 8.85%	73.32 ± 9.08%	74.86 ± 8.89%	62.04 ± 8.61%
NG1/NG15	94.25 ± 4.59%	73.72 ± 9.08%	74.86 ± 8.89%	62.04 ± 8.61%
NG18/NG19	69.34 ± 11.75%	63.86 ± 6.09%	64.04 ± 7.23%	63.96 ± 8.48%

3.2 Categorical clustering

Categorical clustering is similar in flavor to document-term clustering. A data object is not a document containing terms, but rather a vector of characteristics. A particular example is Congressional voting data [6]. Each data object is a Congressman, and the vector of characteristics is how he voted on every bill or law put through Congress. A natural clustering of congressmen exists – political party affiliations. In this particular case, categorical clustering asks the question: given the voting record of each congressman, can we partition congressmen into two clusters that closely match their party lines?

Table 7: Congressional Voting Data (Entropy)

Spectral	COOLCAT	ROCK
.480	.498	.499

We show that our spectral algorithm can also be applied in such a scenario. In Table 7, we see that we do better than both COOLCAT [9] and ROCK [18].

3.3 Synthetic data from mixture models

We also tested our algorithm on data generated from a mixture model. The mixture model is similar to others proposed in the literature [25], and was devised to model document clustering.

Our mixture model consists of k clusters with mixing weights $w_1 \ldots w_k$. Each cluster is defined by a probability distribution p_i on m terms. We generate a data object (document) according to the following procedure:

- Pick a cluster p_i according to the mixing weights w_i
- Independently pick t terms according to the probability distribution p_i.

- Return the document that is comprised of the t terms.

Note that since each data object is generated from a particular distribution, there is a notion of correct classification – each document belongs to the distribution it was generated from.

The variation distance between two distributions $p, q \in \mathbb{R}^m$ is:

$$\sum_{i=1}^{m} |p(i) - q(i)|.$$

The smaller the variation distance, the closer the distributions are to each other. In terms of the mixture model, we expect to correctly classify two documents if their respective probability distributions have large variation distance. As the variation distance decreases, the overlap may be too strong to efficiently determine the correct clustering.

Our experiment studies the effect of the minimum variation distance between any two clusters in the mixture model (which we call the variation distance of the mixture model) on the ability of the algorithm to cluster correctly. To this end, we generated documents from mixture models at different variation distances and ran our clustering algorithm on the corresponding document-term matrices. The accuracy was recorded at each variation distance.

The exact details of the experiment follow. We constructed 100 mixture models $M_1 \ldots M_{100}$ differing only in their clusters (distributions on terms). The ith mixture model has distributions $D_1^{(i)} \ldots D_{10}^{(i)}$ on 200 terms; the pairwise variation distance for any two is exactly $2 - \frac{i}{50}$. We divide the terms into 10 equally sized blocks $B_1 \ldots B_{10}$ of size 20. Distribution $D_j^{(i)}$ is uniform on the jth block, which contains $1 - \frac{9i}{1000}$ of the probability mass, and uniform on the other 9 blocks, which contains the rest of the probability mass.

For each of the mixture models, we performed 100 trials in which we generated N documents each with $t = 30$ terms, and ran the clustering algorithm on the corresponding document-term matrix. Figure 2 shows the corresponding plot of variation distance vs. average accuracy over 100 trials when $N = 100$ and $N = 1000$. It is worth noting that performance increases when the number of documents generated increases – this can be attributed to the conductance increasing within a cluster.

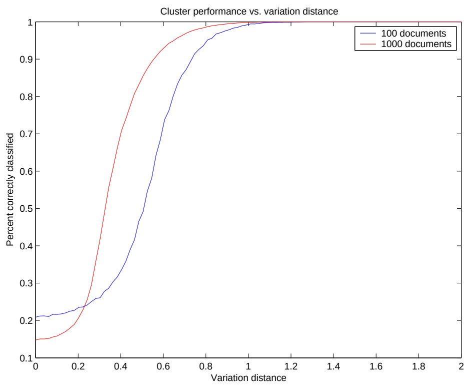


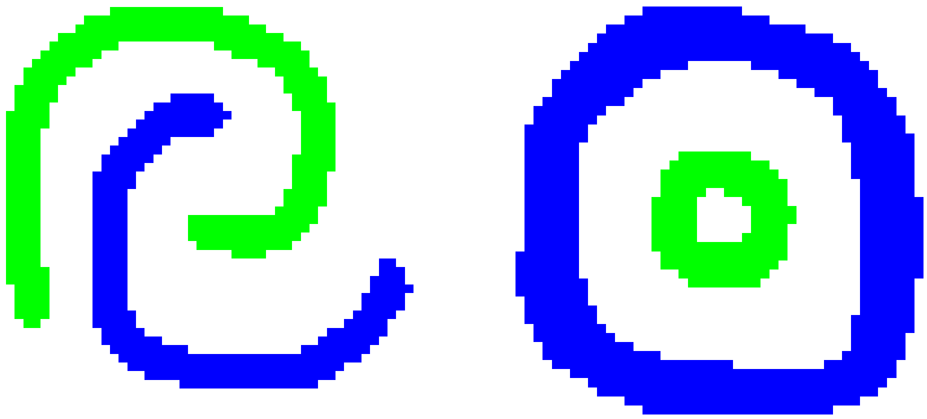
Figure 2: Performance of algorithm on synthetic mixture model

3.4 2-d images

Spatial clustering has also been studied in the literature; see [23, 33]. The main idea is to classify visual points in space as objects.

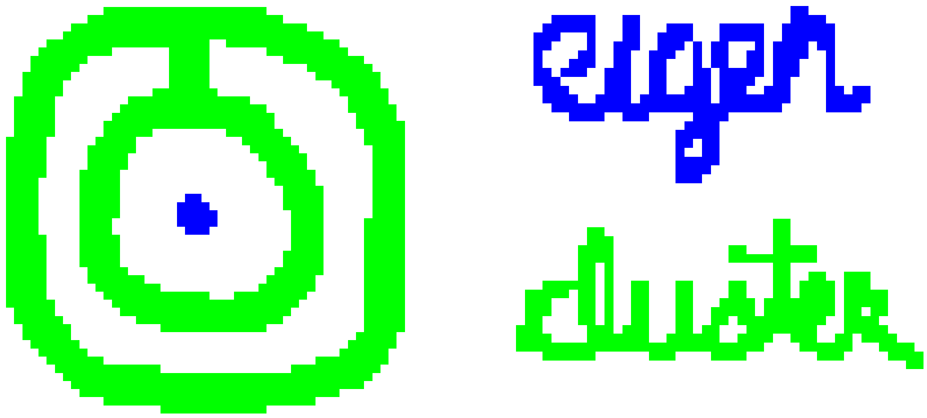
Our experiments are very similar in nature to [22]. In these experiments, our clustering algorithm is given the similarity matrix of an image. Each non-zero pixel is a data object, and the similarity of two pixels p, q is defined as:

$$s(p, q) = e^{-\frac{||x-y||^2}{\sigma^2}}$$



(a) Spirals (b) Concentric circles

Figure 3: Spatial clustering images



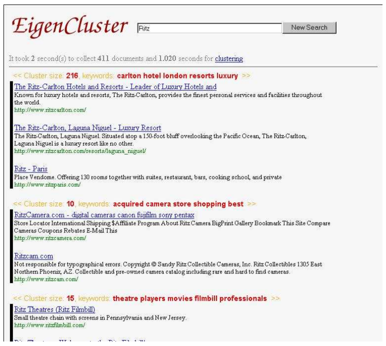
(a) Concentric circles with band (b) Eigencluster

Figure 4: Spatial clustering images

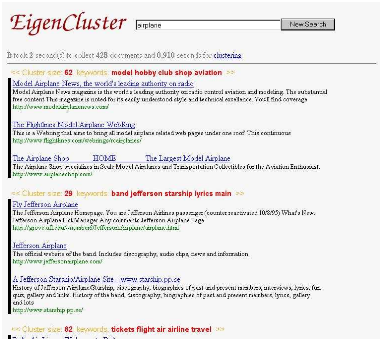
for a suitable definition of σ^2. In each of the images, there is an obvious correct classification of two objects. Note that in many of these images, the objects are non-convex and may be interwoven in various and complex ways. However, the notion of conductance correctly quantifies the similarity of the objects. Figures 3(a), 3(b), 4(a), and 4(b) show a few of the more interesting images. In these figures, pixels in blue are one cluster while pixels in green make up the other cluster.

3.5 Eigencluster: a web search implementation

Eigencluster is a working implementation of our clustering algorithm as a web search engine available at (`http://www-math.mit.edu/cluster`). Given terms to search for, *Eigencluster* returns a set of webpages clustered into topics. Figures 5(a), 5(b), 6(a), and 6(b) show example output of a web search. *Eigencluster* first takes the resulting web pages of the search from a standard web search engine and clusters the document-term matrix. Note that the amount of time to cluster is very small; all example searches take less than 3 seconds on commodity hardware. In all of the example searches, the term that is searched for exhibits *polysemy*; that is, the term can stand for many different concepts. *Eigencluster* identifies these concepts and

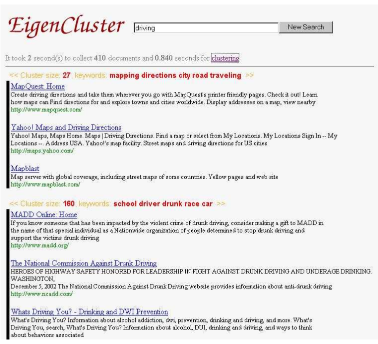


(a) Search term: Ritz



(b) Search term: Airplane

Figure 5: Example searches



(a) Search term: Driving

(b) Search term: Cookie

Figure 6: Example searches

puts webpages into the appropriate clusters.

3.6 Discussion

The performance of any clustering algorithm in the clustering experiments described above is very sensitive to the similarities between data objects. In particular, when clustering documents, there are several preprocessing steps that are often done which modify pairwise similarities. Some preprocessing steps include:

1. Removal of terms that appear too frequently
2. Removal of terms that appear infrequently
3. Weighting terms based on term frequency
4. Using only the first N terms in the document

In the experiments performed on document clustering, we attempted to match the exact preprocessing steps. When this was not possible (because the preprocessing steps were not described explicitly), we modified parameter 4 and, otherwise keeping the document (term vector) in its raw form. This may explain why we perform worse than [37] when clustering more than two newsgroups.

4 Conclusion

We have presented a practical clustering algorithm with worst-case theoretical guarantees. The timing experiments show that it is reasonable to use the clustering algorithm in practice. Furthermore, the algorithm is quite versatile in that it works with any similarity matrix, given either implicitly or explicitly, without taking into account the structure of the specific application. Nevertheless, on a variety of experiments with real as well as synthetic data, it mostly seems to outperform existing algorithms.

Several questions remain. One aspect worth further investigation is how the similarities affect the results. In experimentation, it seems that the quality of the clustering is very dependent on the degree of similarity. Is there a range or pattern of similarities for which the algorithm is better suited? It would also be good to evaluate whether conductance (or the eigenvalue gap) truly reflects clustering quality in real data. If so, in what applications is this the case? This would allow us to characterize the application area for which spectral methods are suitable.

References

[1] 20 Newsgroups Data Set. `http://www.ai.mit.edu/people/jrennie/20Newsgroups/`.

[2] CLUTO: A Software Package for Clustering High-Dimensional Datasets. `http://www-users.cs.umn.edu/~karypis/cluto/files/datasets.tar.gz`.

[3] PDDP Data. `http://www-users.cs.umn.edu/~boley/`.

[4] Reuters Data Set. `http://www.daviddlewis.com/resources/testcollections/reuters21578/`.

[5] SMART Data Set. `ftp://ftp.cs.cornell.edu/pub/smart`.

[6] UCI Machine Learning Repository. `http://www.ics.uci.edu/~mlearn/MLRepository.html`.

[7] C. J. Alpert and S. Z. Yao. Spectral partitioning, the more eigenvectors the better. In *32nd ACM/IEEE Design Automation Conference*, pages 195–200, 1995.

[8] L. Douglas Baker and Andrew Kachites McCallum. Distributional clustering of words for text classification. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 96–103. ACM Press, 1998.

[9] Daniel Barbara, Yi Li, and Julia Couto. Coolcat: an entropy-based algorithm for categorical clustering. In *Proceedings of the eleventh international conference on Information and knowledge management*, pages 582–589. ACM Press, 2002.

[10] F. Beil, M. Ester, and X. Xu. Frequent term-based text clustering. In *Proc. 8th Int. Conf. on Knowledge Discovery and Data Mining*, 2002.

[11] Daniel Boley. Principal direction divisive partitioning. *Data Mining and Knowledge Discovery*, 2(4):325–344, 1998.

[12] C. Buckley and A.F. Lewit. Optimization of inverted vector searches. In *In Proc. ACM SIGIR*, pages 97–110, 1985.

[13] P. Cheeseman and J. Stutz. Bayesian classification (AUTOCLASS): Theory and results. In U. M. Fayyad, G. Piatetsky-Shapiro, P Smyth, and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*. 1995.

[14] C.H. Cheng, A.W. Fu, and Y. Zhang. Entropy-based subspace clustering for mining numerical data. In *Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1999.

[15] Douglass R. Cutting, David R. Karger, Jan O. Pedersen, and John W. Tukey. Scatter/gather: a cluster-based approach to browsing large document collections. In *Proceedings of the 15th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 318–329. ACM Press, 1992.

[16] Scott C. Deerwester, Susan T. Dumais, Thomas K. Landauer, George W. Furnas, and Richard A. Harshman. Indexing by latent semantic analysis. *Journal of the American Society of Information Science*, 41(6):391–407, 1990.

[17] Inderjit S. Dhillon. Co-clustering documents and words using bipartite spectral graph partitioning. In *Knowledge Discovery and Data Mining*, pages 269–274, 2001.

[18] Sudipto Guha, Rajeev Rastogi, and Kyuseok Shim. ROCK: A robust clustering algorithm for categorical attributes. *Information Systems*, 25(5):345–366, 2000.

[19] Thomas Hofmann. The cluster-abstraction model: Unsupervised learning of topic hierarchies from text data. In *IJCAI*, pages 682–687, 1999.

[20] R. Kannan, S. Vempala, and A. Vetta. On clusterings: Good, bad, and spectral. In *Proceedings of IEEE Foundations of Computer Science*, 1999.

[21] Bjornar Larsen and Chinatsu Aone. Fast and effective text mining using linear-time document clustering. In *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 16–22. ACM Press, 1999.

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

[23] R. T. Ng and J. Han. Efficient and effective clustering methods for spatial data mining. In Jorgeesh Bocca, Matthias Jarke, and Carlo Zaniolo, editors, *20th International Conference on Very Large Data Bases, September 12–15, 1994, Santiago, Chile proceedings*, pages 144–155, Los Altos, CA 94022, USA, 1994. Morgan Kaufmann Publishers.

[24] Adam Nickerson, Nathalie Japkowicz, and Evangelos Milios. Using unsupervised learning to guide re-sampling in imbalanced data sets. In *Proceedings of the Eighth International Workshop on AI and Statitsics*, pages 261–265, 2001.

[25] Christos H. Papadimitriou, Hisao Tamaki, Prabhakar Raghavan, and Santosh Vempala. Latent semantic indexing: A probabilistic analysis. In *Proocedings of the ACM Conference on Principles of Database Systems*, 1998.

[26] C. E. Shannon. A mathematical theory of communication. In *Bell Systems Technical Journal*, volume 27, pages 379–423, 1948.

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

[28] Noam Slonim and Naftali Tishby. Document clustering using word clusters via the information bottleneck method. In *Research and Development in Information Retrieval*, pages 208–215, 2000.

[29] M. Steinbach, G. Karypis, and V. Kumar. A comparison of document clustering techniques. In *In KDD Workshop on Text Mining*, 2000.

[30] J. Theiler and G. Gisler. A contiguity-enhanced k-means clustering algorithm for unsupervised multispectral image segmentation. In *Proceedings of the Society of Optical Engineering*, pages 108–111, 1997.

[31] C. J. Van Rijsbergen. *Information Retrieval, 2nd edition.* Dept. of Computer Science, University of Glasgow, 1979.

[32] J.Z. Wang, G. Wiederhold, O. Firschein, and S.X. Wei. Content-based image indexing and searching using daubechies' wavelets. In *Int. J. Digital Library*, volume 1, pages 311–328.

[33] Wei Wang, Jiong Yang, and Richard R. Muntz. STING: A statistical information grid approach to spatial data mining. In Matthias Jarke, Michael J. Carey, Klaus R. Dittrich, Frederick H. Lochovsky, Pericles Loucopoulos, and Manfred A. Jeusfeld, editors, *Twenty-Third International Conference on Very Large Data Bases*, pages 186–195, Athens, Greece, 1997. Morgan Kaufmann.

[34] W. Wong and A. Fu. Incremental document clustering for web page classification. In *IEEE 2000 Int. Conf. on Info. Society in the 21st century: emerging technologies and new challenges*, 2000.

[35] Weiss Y. Segmentation using eigenvectors: a unifying view. In *Proceedings IEEE International Conference on Computer Vision*, pages 975–982, 1999.

[36] Oren Zamir, Oren Etzioni, Omid Madani, and Richard M. Karp. Fast and intuitive clustering of web documents. In *Knowledge Discovery and Data Mining*, pages 287–290, 1997.

[37] H. Zha, C. Ding, M. Gu, X. He, and H. Simon. Spectral relaxation for k-means clustering. In *Neural Info. Processing Systems (NIPS 2001)*, 2001.

[38] H. Zha, X. He, C. Ding, M. Gu, and H. Simon. Spectral relaxation for k-means clustering. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, pages 1057–1064, Cambridge, MA, 2002. MIT Press.

[39] Tian Zhang, Raghu Ramakrishnan, and Miron Livny. Birch: An efficient data clustering method for very large databases. In H. V. Jagadish and Inderpal Singh Mumick, editors, *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data, Montreal, Quebec, Canada, June 4-6, 1996*, pages 103–114. ACM Press, 1996.

[40] Ying Zhao and George Karypis. Evaluation of hierarchical clustering algorithms for document datasets. In *Proceedings of the eleventh international conference on Information and knowledge management*, pages 515–524. ACM Press, 2002.