A Comparison of Spectral Clustering Algorithms

Deepak Verma
Department of CSE
University of Washington
Seattle, WA 98195-2350
deepak@cs.washington.edu

Marina Meilă
Department of Statistics
University of Washington
Seattle, WA 98195-4322
mmp@stat.washington.edu

Abstract

Spectral Clustering has become quite popular over the last few years and several new algorithms have been published. In this paper, we compare several of the best-known algorithms from the point of view of clustering quality over artificial and real datasets. We implement many variations of the existing spectral algorithms and compare their performance to see which features are more important. We also demonstrate that spectral methods show competitive performance on real dataset with respect to existing methods.

1 Introduction

Clustering has always been a hard problem and an active topic of research. Recently, a new approach has started to get a lot of attention namely spectral methods. The spectral methods for clustering usually involve taking the top eigen vectors of some matrix based on the distance between points (or other properties) and then using them to cluster the various points.

Spectral clustering techniques have seen an explosive development and proliferation over the past few years. They promise to become strong competitors for other clustering methods. Several successes have already been registered (LSA,[9]). Spectral methods are attractive because they are easy to implement and are reasonably fast (for *sparse* data sets up to several thousands). Also they do not intrinsically suffer from the problem of local optima. (Though depending on the exact algorithm some local optima might be there.)

In spite of the large number of papers on spectral clustering, so far no systematic comparison between the existing algorithms has been published. This is what we set out to do here. We intend to take a look at four spectral algorithms and take them apart. We generate a list of algorithms which are made from different parts of different algorithms and compare their performance. We hope to be able to find out which of these sub-components are important and which not, and to see if some combination of these works the best.

2 Spectral Clustering Algorithms

The algorithms we selected are at this date among the most popular of the published ones. We have also aimed at representing a diverse set of algorithmic features. The algorithms are: (1) the image segmentation algorithm introduced by Shi and Malik (SM) [9], (2)A variant by Kannan, Vempala and Vetta (KVV) [2], (3) the algorithm of Ng, Jordan and Weiss (NJW) [8], (4) an algorithm suggested by Meilă and Shi (Multicut) [5]. We also present the results obtained with the Single and Ward linkage algorithms (denoted by MST ,Ward) as a "strawman" in order to demonstrate that the clustering tasks that we chose are not trivial.

Before describing the algorithms in more detail, we introduce some notation. Then we would describe the four spectral and two grouping algorithms.

2.1 Notation

The set of data points to be clustered will be denoted by I, with $|I| = n$. For each pair of points $i, j \in I$ a similarity $S_{ij} = S_{ji} \geq 0$ is given. The similarities S_{ij} can be viewed as weights on the *undirected* edges ij of a graph G over I. The matrix $S = [S_{ij}]$ plays the role of a "real-valued" adjacency matrix for G. Let $D_i = \sum_{j \in I} S_{ij}$ be called the *degree* of node i, and the *volume* of a set $A \subset I$ be $Vol\,A = \sum_{i \in A} D_i$. The set of edges between two disjoint sets $A, B \subseteq I$ is called the *edge cut* or in short the *cut* between A, B.

A clustering $\mathcal{C} = \{C_1, C_2, \ldots C_K\}$ is a partitioning of I into the nonempty mutually disjoint subsets $C_1, \ldots C_K$. In the graph theoretical paradigm a clustering represents a *multiway cut* in the graph G.

All the algorithms that we use here just need a similarity matrix between points (except anchor. But we never us it on the original (unmapped) points). So all we need is a data with similarity matrix and there may not be an actual set of distinct points in the initial domain or the points may even come from an infinite dimensional space. (something an output or an kernel function).

2.2 Overview

The algorithms presented here can be thought of as consisting of 3 stages:

- **Preprocessing:** This is a form of normalization of the similarity matrix S. We did some smoothing initially to make sure that matrix is not too ill conditioned. However to make results more relevant w.r.t. to other papers, we eventually dropped the smoothing.
- **Spectral Mapping:** Some eigenvectors of the preprocessed similarity matrix are computed. Each data point i is mapped to a tuple representing the values of component i in the aforementioned eigenvectors.
- **Postprocessing/Grouping:** A (usually simple) grouping algorithm clusters the data (in the original or spectral domain).

There are three kind of algorithms presented here.

- **Recursive Spectral:** These algorithms try to split the data into two partitions based on a single eigenvector and are then are recursively used to generate more partitions.
- **Multiway Spectral:** These use more information in multiple eigenvectors to do a direct multiway partition of data.
- **Non spectral:** A (usually simple) grouping algorithm that clusters the data quickly. These are used in conjunction with the Multiway spectral algorithms and also provide a baseline for performance.

For some of the algorithms (SM , KVV) heuristic methods for finding the number of clusters K have been suggested [9, 2]. In this work, to provide for a fair comparison, we have assumed for all the algorithms that the number of clusters K is given in advance.

2.3 The Shi and Malik (SM) algorithm

This algorithm was introduced by [9] as a heuristic algorithm aimed to minimize the *Normalized Cut* criterion proposed by the same authors. The normalized cut between two sets $A, B \subseteq I$ is defined as

$$NCut(A, B) = Cut(A, B) \left(\frac{1}{Vol\,A} + \frac{1}{Vol\,B} \right)$$

According to [9] the set I is partitioned into two clusters $C, C' = I \setminus C$ that minimize $NCut(C, C')$ over all possible two way partitions of I. This problem is provably NP-hard, but the authors show that under certain special conditions a spectral algorithm exists that finds the optimum.

Algorithm SM

1. Compute
$$P = D^{-1} S \tag{1}$$

2. Let $1 = \lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n$ be the eigenvalues of P and $v^1, v^2, \ldots, v^n$ the corresponding eigenvectors[1] Compute v^2.
3. MIN-RATIO-CUT
   (a) Sort the elements of v^2 in increasing order. Denote by v_i^2 the i-th element in the sorted list.
   (b) For $i = 1, \ldots n-1$
   Compute $NCut(C_i, C_i')$ where $C = \{1, \ldots i\}$, $C' = \{i+1, i+2 \ldots n\}$
   (c) Partition I into the two clusters C_{i_0}, C_{i_0}' where $i_0 = \min_i NCut(C_i, C_i')$
4. Repeat steps 1–3 recursively on the cluster with the largest λ_2 until K clusters are obtained.

MIN-RATIO-CUT is inspired by [10] which suggests that a line search along the vector produces better cuts than the original heuristic. This algorithm is almost identical with spectral algorithm II in [2]. That is discussed in more detail in 2.4. Based on that another version for this algorithm in implemented using the conductance as a criterion in Min-Ratio-Cut .

2.4 The Kannan, Vempala and Vetta Algorithm (KVV)

The KVV is very similar to to SM algorithm with two differnces. One difference is in step 3, where the optimal cut is found with respect to the Cheeger conductance $\phi(C_i, C_i')$, another measure of cut quality. The conductance of a clustering $\{C, I \setminus C\}$ is defined as

$$\phi(C, I \setminus C) \ = \ \frac{Cut(C, I \setminus C)}{\min Vol\, C,\ Vol\, I \setminus C)}$$

Also in the recursive step the kvv variant decides the next cluster is the one with the minimum conductance.This variant of step 3 will be called Min-Conductance .

Another difference is in the "normalization" past the first iteration of step 1. The SM algorithm just takes the block of S corresponding to the current cluster. The variant in [2] always uses blocks from the P computed at the first iteration. To ensure that the row sums of the blocks equal 1, the diagonal elements P_{ii} of the current block are adjusted. Thus, this variant ignores the self-similarity of the data points in all but the first iteration.

To adjust the P_{ii} there are two possibilities. We can either scale up all the entries in row to sum up to one or add the extra weight to the diagonal element. We call the first variant `kvv_mult` and the second `kvv_add`.

2.5 The Ng, Jordan and Weiss (NJW) algorithm

Algorithm NJW

1. Set the diagonal elements S_{ii} to 0.
2. Compute the matrix
$$L \ = \ D^{-\frac{1}{2}} S D^{-\frac{1}{2}} \tag{2}$$
3. Let $1 = \mu_1 \geq \mu_2 \geq \ldots \geq \mu_K$ be the K largest eigenvalues of L and $u^1, u^2, \ldots, u^K$ the corresponding eigenvectors[2] All eigenvectors are normalized to have unit length. Form the matrix $U = [u^1\, u^2\, \ldots u^K]$ by stacking the eigenvectors in columns.
4. Form the matrix Y from U by renormalizing each of U's rows to have unit length (i.e $Y_{ij} = U_{ij} / \sqrt{\sum_j U_{ij}^2}$).
5. K-Means-Orthogonal Treating each row of Y as a point in K dimensions, cluster them by the K-means algorithm to obtain the final clustering. The K-means algorithm is initialized by the Orthogonal-Initialization method described in [8].

[1]If the eigenvalues are not distinct, pick the eigenvectors such that ${v^i}^T D v^j = 0$ for $i \neq j$. This is always possible and the Matlab implementation that we used does it automatically.

[2]If the eigenvalues are not distinct, choose u^k's that are orthogonal to each other. L is related to the *Laplacian* of S. See e.g [9] for details.

In [8] it is proved that if the clusters are well separated in the sense that the similarity matrix S is almost block diagonal, and if the sizes of the clusters and degrees of individual nodes don't vary too much, the rows of the Y matrix cluster near K orthonormal vectors in R^K. This fact suggested the orthogonal initialization method.

We implemented a slightly different version of this algorithm which we think has greater numerical stability. The details are in the appendix.

2.6 The Meila-Shi algorithm

This algorithm was suggested in [6]. **Algorithm** MULTICUT

1. Compute the stochastic matrix P as in (1).
2. Compute $v^1, \ldots v^K$ the eigenvectors of P corresponding to the K largest eigenvalues. Form the matrix V whose columns are $v^1, \ldots v^K$.
3. Cluster the rows of V as points in a K-dimensional space.

2.7 Anchor Algorithm

A "flat" version of **the Anchor algorithm** of [7], also very related to the minimum diameter clustering method of [1].

Algorithm ANCHOR

1. Choose a point at random. Set $k = 1$, $k' = 0$. Choose anchor x_1 to be the farthest point from the initial point.
2. Construct C_k the cluster associated with x_k as a list of points that are closer to x_k than to any other anchor. The list is sorted by decreasing distance from x_k.
3. Test if x_k has enough points. If $|C_k| < n_{min}$ then $k' = k' + 1$.
4. Set $k = k + 1$. Choose anchor x_k to be the farthest point from all existing anchors.
5. If $k - k' < K$ go to step 2.

Note that the algorithm may produce more than K clusters. It is also possible that it never produces K clusters having more than n_{min} points. In our experiments, all performed with $n_{min} = 3$, the latter was never observed.

2.8 Linkage algorithms

These are the hierarchical clustering algorithm which work on distances between the points. Since we have similarities, we need to choose a way of mapping similarities to distances. We chose to use the inverse of similarity as the measure (A very small value was added it to similarity to ensure that inverse of zero is not taken).

We used two methods: single linkage and ward linkage. Single linkage is same as performing the MST on the *dissimilarities* graph of the points and ward linkage is similar except that distance is the inner square distance. For more details on the ward algorithms see [12].

As it will be shown in the following sections, the different algorithmsa re much closer then they initially appear. Both the experiments and the theory suggest that the differences between algorithms will depend strongly on the quality of the postprocessing step. Therefore, we experimented with several different clustering methods that will be described here.

3 Theoretical Results

Here we compare the algorithms with respect to what we call their *perfect points*, values of S for which these spectral methods are supposed to perform well. We will show that, even though apparently the four algorithms are different, some of them are very similar near the perfect points.

First we present a modification to the two algorithms to make them numerically more stable and then we present the conditions and proof of similarity. The symbols U, V, Y, S, P, D, K have the same meaning as defined in the previous section.

3.1 A modification to the NJW and Multicut method

As discussed in section 2.5, NJW use the top K eigenvectors of $L = D^{-\frac{1}{2}}SD^{-\frac{1}{2}}$ to map data. We use the top K eigen vectors of the generalized eigen system $Sx = \lambda Dx$. This is a numerically more stable method to compute the eigen vectors as there is no division by D involved, which could contain very small values. (An outlier for example would have a very small degree as it is not close to another point).

To prove why this gives the same vector Y consider the following:

$$\begin{aligned} D^{-\frac{1}{2}}SD^{-\frac{1}{2}}u &= \lambda u, \text{ Premultiplying by } D^{-\frac{1}{2}} \text{ we get} && (3)\\ D^{-1}S(D^{-\frac{1}{2}}x) &= \lambda(D^{-\frac{1}{2}}x) \text{ Putting } v = D^{-\frac{1}{2}}x (\text{ hence } x = D^{\frac{1}{2}}v) \text{ we get} && (4)\\ D^{-1}Sv &= \lambda v \text{ and hence } Sv = \lambda Dv. && (5)\end{aligned}$$

Now consider the top K eigen vectors $U_{n\times K}$ and $V_{n\times K}$. $U = D^{\frac{1}{2}}V$. Since D is a diagonal matrix this means that the i^{th}row of U is same as the i^{th} row of V scaled by $D_{ii}^{\frac{1}{2}}$. So after the row is normalized to length 1, the Y obtained from V is identical to the Y obtained from U.

For the multicut method observe that $P = D^{-1}S$ so the generalized eigen value system $Sv = \lambda Dv$ is mathematically equivalent and numerically more stable than the computing P and its eigenvectors (for the reasons stated above).

3.2 Preliminaries: The perfect S

When each of the clusters in the postprocessing step of a spectral algorithm is reduced to a distinct point we say that S is *perfect* for the respective algorithm. For example, a block diagonal similarity matrix is perfect for all algorithms. A perfect S represents the ideal situation for a clustering algorithm. Here we essentially show that when S is such that the resulting P is block-stochastic, a term that will defined below, then S is perfect for NJW and Multicut and should give good performance on recursive algorithms.

When each of the clusters is reduced to a point ($\gamma_i = \gamma_j \ \forall i, j \in C_s \ \forall s$) by the spectral mapping we say that S is *perfect* for the respective algorithm. A vector $v = [v_1, v_2, \ldots, v_n]^T$ is *piecewise constant* (*PC*) w.r.t a clustering Δ iff $v_i = v_j$ whenever i, j are in the same cluster.

A matrix S is called *block diagonal* (BD) w.r.t. a clustering Δ iff $S_{ij} = 0$ whenever i and j belong to different clusters. It can be easily shown ([2, 9, 6, 8]) that block diagonal S is perfect for all the methods.

Block stochastic P Let P be a stochastic matrix. P is *block stochastic* [6] (BS) w.r.t. a clustering $\Delta = \{C_1, \ldots C_K\}$ iff for all $s, s' = 1, \ldots K$ the sums $P_{is} = \sum_{j\in C_s} P_{ij}$ are equal for all $i \in A_{s'}$ and the matrix $R = [P_{ss'}]$ (with $P_{ss'} = \sum_{j\in A_{s'}} P_{ij}, i \in A_s$) is non-singular. A block stochastic P is guaranteed ([6]) to have some K eigenvectors PC w.r.t. Δ. In the rest of the paper we will assume that the PC vectors of P are always the top K eigenvectors. So from now on we will say that a matrix P has K *piecewise constant vectors* (PCE), or is block-stochastic w.r.t. Δ when its top K eigenvectors are piecewise constant. Thus, if a P has PCE, the corresponding S is perfect for the Multicut algorithm.

Proposition 1 *Let $A_{n\times k}$ be a real matrix and $D_{n\times n}$ be a diagonal matrix such that $A^TDA = I_{k\times k}$ and that A has atmost k unique rows. Then A is guaranteed to have* exactly *k unique* orthogonal *rows. (proved in Appendix).*

This is proved in the appendix. Consider a BS P and the corresponding V. Since $SV = \Lambda DV$, $V^TDV = I$. So the above proposition implies that whenever P is perfect for Multicut , all the clusters in the spectral domain are unique and orthogonal (but not orthonormal).

As shown in **??** Y in the NJW algorithm can also be obtained by normalizing rows of V (from Multicut)). By the above proposition, we have that the rows of Y are also perfectly clustered with the clusters orthogonal to each. other and hence BS P is perfect for NJW as well. This generalizes the result in [8] where this property was shown for block diagonal case and shows that a perfect S for Multicut is also perfect for NJW . This shows that NJW will work well in a much wider range of cases then previously believed. The reverse can also be shown though the proof is slightly more involved and shown in the appendix. It follows that:

Theorem 2 *Whenever the S is perfect for Multicut it is perfect for NJW and vice versa.*

In other words, Multicut and NJW are equivalent when S is perfect, although they use different spectral mappings; NJW maps the clusters to orthonormal vectors, while Multicut maps them to orthogonal vectors of different lengths. Away from the perfect S the behavior of the two algorithms may differ. We will investigate this in the experiments section.

3.3 Recursive Algorithms

We have shown the equivalence of for NJW and Multicut block stochastic P. Now we examine the behavior of the recursive algorithms in the same situation.

Take the case when we are splitting the points $I' \subseteq I$ into two clusters. (At the top level or in any of the recursive steps). Let the P' that we have (*for I'*) be block stochastic w.r.t Δ'. In that case the second eigen vector would be PC w.r.t. to the Δ so when we reorder the points based on this eigen vector we have the points in the right *order*. That is, if we chose the right point to partition , none of the clusters in Δ would be split. Also the P'', P''' for these two split cluster be block stochastic (w.r.t to the two parts of Δ) setting the optimal stage for the recursive sub steps.

The criteria that the two algorithms use for splitting (minimum Ncut or conductance) do not ensure that this optimal point of partition is chosen, unless P is block diagonal (then Ncut and conductance are zero at only these positions). So, for the block diagonal case, all algorithms are equivalent, but we cannot say what happens in the general BS case. As it will turn out from the experiments, one of the algorithms, SM , behaves exactly like the multiway algorithms, while KVV behaves differently.

The above remarks also suggest an alternative to the SM and KVV algorithms in which one chooses the partition based on largest *difference* in the sorted eigenvector. We explored this in our experiments.

4 Datasets

As discussed above there is a whole lot of spectral methods each with its little variation that needs to compared to each other. Since it is not possible to visually compare them we needed datasets which are "pre-classified" So that it is possible to compute the clustering error and the VI w.r.t. true clustering. (see section 5) We used both artificial and real datasets as described below. The artificial datasets were primarily used to demonstrate the robustness of algorithms to noise.

4.1 S100: A Block Stochastic Matrix

This is the "ideal" case for the multicut and ang-based spectral algorithms. Her e we constructed an 100×100 matrix which consists of 5 clusters. There are five clusters present of size 10,20,30,20 and 20 respectively. The similarity matrix is also slightly block diagonal. The purpose of using this dataset was to demonstrate the stability of the algorithms w.r.t to noise. This data file is called `block-stochastic`.

4.2 Handwritten digits

This is the set of optical handwritten digit recognition that is available in the NIST site.There are lots of version available with different preprocessing. In particular we used the data set and preprocessing as mentioned in [3]

Here is the description of preprocessing done by . They used preprocessing programs made available by NIST to extract normalized bitmaps of handwritten digits from a preprinted form. From a total of 43 people, 30 contributed to the training set and different 13 to the test set. 32x32 bitmaps are divided into non overlapping blocks of 4x4 and the number of on pixels are counted in each block. This generates an input matrix of 8x8 where each element is an integer in the range 0..16. This reduces dimensionality and gives invariance to small distortions.

We further down sampled the dataset to 100 elements per digit giving a total of thousand 64 dimensional points and 10 clusters. We call this dataset `digit1000`. Some of digits were more easy to distinguish from another and in particular digits `0,2,4,6,7` were a lot more easier to distinguish than the others. So we created another dataset containing the hundred instances of just these five digits. We called this data set `digitFive1000`. (The `1000` just to remind that this is the same the `digit1000` database.

4.3 Gene Expression Data

DNA microarrays provide a way to the biologists to study the variation of many genes together. Using these has been a plethora of gene expression data generated in the community. This has led to the great need for the data to be analyzed. ([13]). We used one such dataset, the yeast cell cycle data, which is publically available at [14]. It shows the fluctuation of the gene expression levels of over 6000 genes over the two cellcycles (which has 17 time points). The dataset is restricted to the 384 genes who's expression level peak at different points corresponding to the five phases of the cell cycle. The objective given these expression levels is to be able to cluster them into clusters corresponding to the five phases.

There are two kinds of pre processing that are suggested in [13]. First is to the take the logarithm of the expression level and second to "standardize" the mean to be zero and variance 1. These data transformations were done so as to make the data fit better to the gaussian model (They were using mixture models to cluster the data. See [13] for more details). We call the first dataset `cellcycle` and the second `cellcycle-std`.

5 Evaluating Clustering Performance

Measuring a clustering performance in general is a very hard problem. The notion of good clustering is intrinsically tied with the definition of what a cluster is which in itself is a big research topic. In our case measuring clustering performance is easier as in all the datasets we have the "true" clustering available. Given that the clustering performance is just a measure of how "different" is the clustering produced w.r.t the true clustering. There are three kind of measures that we used: Clustering Error and Variation of Information . In this section $\mathcal{C}^{true}$ would represent the true (given) clustering and $\mathcal{C}$ the clustering produced by the clustering algorithm.

5.1 Clustering Error

Clustering error is defined as the number of "misclassification" This the error induced in the clustering w.r.t. to the true clustering.

Let $Confusion$ be the confusion matrix of two clusterings. ($Confusion(k_{true}, k) = |\mathcal{C}^{true}_{k_{true}} \cap \mathcal{C}_k|$ i.e. number of points x that are cluster k_{true} in true clustering and cluster k in the clustering produced. Then

$$CE(\mathcal{C}, \mathcal{C}^{true}) = \left(\sum_{k_{true}} \sum_{k \neq k_{true}} Confusion(k_{true}, k) \right) / n \tag{6}$$

where n is the total number of points.

There is a subtle problem with this naive definition. This does not take into account the renumbering that might happen while clustering. Cluster 1 in the true clustering might be assigned cluster 3 in the clustering produced and so on. To counter that the CE is computed for *all* possible renumbering of the clustering produced and the minimum of all those is taken. (This is computed efficiently by modeling the problem as a maximum weighted bipartite matching problem and then computing the solution using linear programming).

5.2 Variation of Information

Variation of Information (VI) is a *metric* introduced in [4] to compare two clusterings. It measures the amount of information that is lost/gained from going from one clustering to another and vice versa. To define it let introduce some more notation as used in [4]. (with some changes).

Let n_k be the number of point in k^{th} cluster in $\mathcal{C}$. Let $P(k) = \frac{n_k}{n}$. Then the *entropy associated with clustering* $\mathcal{C}$ is defined as

$$H(\mathcal{C}) = -\sum_{k=1}^{K} P(k) \log P(k)$$

Define $n_{k_{true}k} = |\mathcal{C}^{true}_{k_{true}} \cap \mathcal{C}_k|$ and $P(k_{true}, k) = \frac{n_{k_{true}k}}{n}$. Then *mutual information* between the clustering $I(\mathcal{C}^{true}, \mathcal{C})$ is defined as

$$I(\mathcal{C}^{true}, \mathcal{C}) = \sum_{k_{true}=1}^{K} \sum_{k=1}^{K} P(k_{true}, k) \log \frac{P(k_{true}, k)}{P^{true}(k_{true})P(k)}$$

Given these the *variation of information* is

$$VI(\mathcal{C}^{true}, \mathcal{C}) = H(\mathcal{C}) + H(\mathcal{C}^{true}) - 2I(\mathcal{C}^{true}, \mathcal{C})$$

See [4] for various properties of this measure.

5.3 Wallace Index

Wallas Index is an *index* introduced in the ([11]) to measure the differences between two clusters. There are two asymmetric criteria defined. We just used one of the indices which represents the probability that a pair of point that were in the same cluster in $\mathcal{C}^{true}$, are also in the same cluster in $\mathcal{C}$. Let $n^{true}_{k_{true}} = |\mathcal{C}^{true}_{k_{true}}|$ and N_{11} be the number of points pairs that are in the same cluster in both $\mathcal{C}$ and $\mathcal{C}^{true}$. Then Wallace Index (one sided) is defined as

$$WI(\mathcal{C}, \mathcal{C}^{true}) = \frac{N_{11}}{\sum_{k_{true}=1}^{K} n^{true}_{k_{true}}(n^{true}_{k_{true}} - 1)/2}$$

This gives the value of 1 if the clustering if perfect (same as true clustering) and zero if the clustering disagree on all point pairs.

6 EXPERIMENTAL SETUP

In this section we provide the specific details on how the experiments were run. Throughout this section we use *AffinityMatrix* of a group vectors $x_1, x_2 \ldots x_n$to be the similarity matrix S such that $S_{ij} = \exp(-||x_i - x_j||^2/2\sigma^2$ where σ would be the parameter used (We would specify the value used). Also K would refer to the input the clustering algorithms to specify the number of cluster to generate. Each algorithm was run multiple number of times and the average taken.

6.1 Exact Algorithms Used

In section 2 we described the algorithms as presented in paper the so called classic version. However to exactly distinguish between the effects of the various components of the spectral algorithms we implemented a whole range of algorithms containing most of the variations of the algorithms mentioned above.

The list of all the algorithms implemented is shown below. In each of the algorithm listed here, the various components represent the application of a particular algorithm in that "stage". So `ang` and `mcut` refer to the spectral mapping using the NJW and Multicut methods into a domain. After mapping the similarity matrix (if required) is obtained by computing the $AffinityMatrix$ with $\sigma = 0.2$ This choice was straight forward in case of NJW algorithm as points lie on a unit sphere. We used the same value for Multicut as well.

The `anchor,ward,kmeans` refer to the respective algorithms applied after the spectral mappings. In `kmeans` we performed kmeans with 5 runs of initializing with orthogonal centers and 20 runs initialized with random centers.

We also had the intuition that the spectral methods might be more effective in clustering points *after* mapping them in the spectral domain. To explore that possibility we implemented the *double spectral* methods like `ang_mcut_ward` in which the first we map the points in the `ang` spectral domain and then those points in the `mcut` spectral domain, finally grouping them using the `ward` method.

Linkage Algorithms:	`single_linkage`	`ward_linkage`	
Multiway Spectral Algorithms:	`njw_ward`	`njw_kmeans`	`njw_anchor`
	`mcut_ward`	`mcut_kmeans`	`mcut_anchor`
	`njw_njw_ward`	`njw_njw_kmeans`	`njw_mcut_ward`
	`njw_mcut_kmeans`	`mcut_mcut_ward`	`mcut_mcut_kmeans`
Recursive Spectral Algorithms:	`sm_ncut`	`kvv1_ncut`	`kvv2_ncut`
	`sm_cond`	`kvv1_cond`	`kvv2_cond`
	`sm_gap`		

Table 1: List of Algorithms

6.2 S100

We used this database primarily to compare the robustness of algorithms w.r.t. to noise. We took the original block stochastic matrix and added uniform noise of increasing magnitude from $\eta = 10^{-0.1}$ to $\eta = 10^{0.7}$ in steps of 0.1 (in the exponent). The noise added was to preserve the *signal-to-noise* ratio in the sense that noise added in S_{ij} was made proportional to degrees of points i, j. More precisely, the new S_{ij} was calculated as follows:

$$S_{ij} = S_{ij} + \left(U(0,1) \times \eta \times \sqrt{D_i \times D_j}\right)/n$$

Where $U(0,1)$ is number between 0 and 1 chosen at random.

For each noise levels 10 such matrices were generated and the average performance of each algorithm taken.

6.3 Handwritten digits

This dataset consisted of vector in the 64 dimensional space ranging from 0 to 16. The similarity matrix was computed as Affinity matrix with $\sigma = 10$. (We experimented with various sigma and the value of 10 seemed to give reasonable results).

For the dataset `digit1000`, we ran each algorithm for 5 iterations for K ranging from 8 to 12. Where as for `digitFive1000`, 10 iterations for $K = 3$ to 7 were executed.

6.4 Gene Expression Data

For both the datasets, `cellcycle` and `cellcycle-std`, the similarity was computed as the correlation coefficients between the gene expression levels of the different genes. (plus 1 to make the similarity matrix positive. So the similarities ranged from 0 to 2.) Five runs were executed for K varying from 3 to 7.

6.5 Implementation

The algorithms are very simple to implemented and we were able to implement each of them using only a few lines of code of matlab. The majority of the time taken was for the eigen decomposition. A full eigen decomposition (using `eig` function of matlab) would take $O(n^3)$ time. However since we just needed the top K eigen vectors, we used `eigs` function to reduce the time taken.

7 Performance Graphs

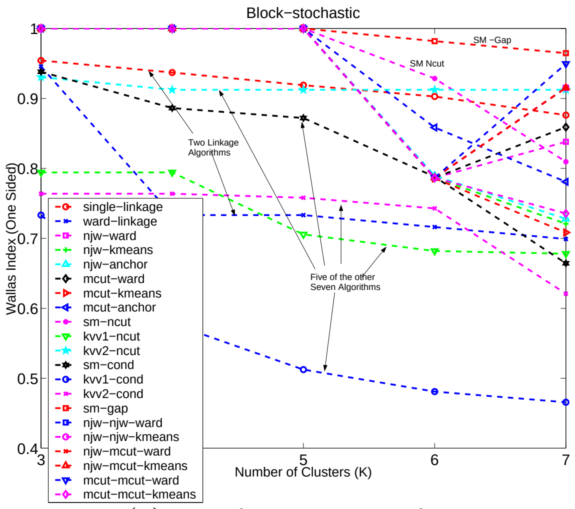
In this section we present the graphs for the various algorithms on the five datasets. Since there are so many algorithms we do not show them all on the same graph. For all the datasets we present six graphs. Three each for the two metrics : Clustering error (CE) and Variation of Information (VI) shown one above the another.

In first column we have the various versions of the multiway spectral algorithms. In the second column the recursive spectral algorithms and the third columns the best five. The best five are chosen as follows: First we pick the best algorithm amongst the linkage, recursive, and multiway spectral classes of algorithms. The other two are the best two of the remaining. (The "best" method was picked by looking at individual graphs) In many of the cases when there were a lot of methods with very similar performance we just chose two which looked better (or arbitrarily if that was hard to decide).

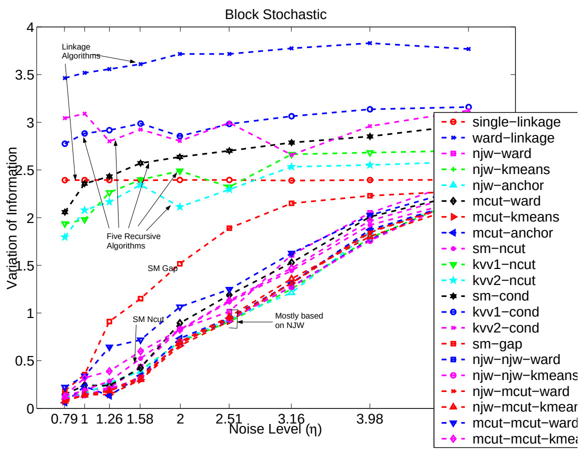
This way we can see how the various classes of spectral methods compare within themselves and w.r.t to each other. Note that **y-axis** of the graphs are *not* the same. And hence different graphs should be compared by just looking at their heights or levels. (This to done to show better contrast in between a particular class, esp. when performance within the class is near identical)

8 Results and Discussion

8.1 S100



(a) Varying K. No noise



(b) True K. Noise added.

Figure 1: **Block Stochastic Dataset**. a) Performance on Wallace index of spectral algorithms on Block Stochastic Datasets. b) Performance on Variation of Information in presence of noise

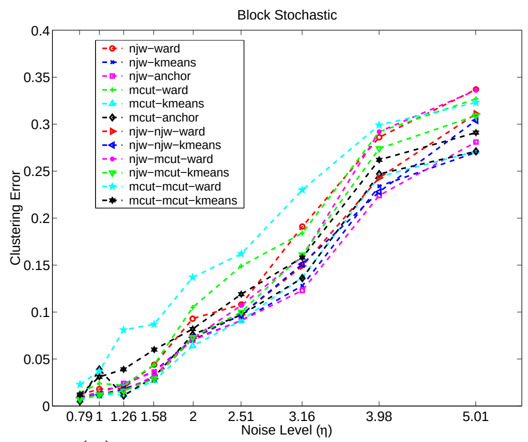
The results of of various algorithms on the block stochastic matrix are presented in figure 1. The first graph (a) shows the Wallace index ([11]) of clustering produce w.r.t to the "true" clustering given various K as input and no noise. The wallace index would have a value of 1 in case the clustering produced does not split the true clusters. i.e. two points which were in the same cluster in the true clustering are in the same cluster in the clustering produced. The graph illustrates a lot of points that in accordance with the theoretical predictions. First of all, the multiway spectral methods, irrespective of the post processing step perform perfectly when $K \leq K_{true} = 5$. This is to be expected as all the points in a single cluster are mapped to the same point in the spectral domain. Also most of the recursive spectral algorithms end up splitting some clusters or another except for `sm_gap` (which in fact performs best) and `sm_ncut`. The reason for this is that conductance or `kvv` based methods are not able to find the optimal point to partition. Another important thing to note is that the multiway spectral algorithms degrade must more steeply if $K > K_{true}$ is used. Eigenvectors corresponding to i^{th} largest eigenvalues are no longer guaranteed to be PC if $i > K_{true}$. This means that those spectral dimensions is essentially "random" w.r.t Δ. The recursive algorithms on the other hand use only the second largest eigenvector which are PC w.r.t. Δ and so the first few cuts are lot more stable leading to better results.

The behavior of algorithms in presence of noise is quite similar. Figure 1(b) shows the how the performance degrades as noise is added. ($K = K_{true} = 5$). As expected the multiway algorithms perform the best. Like above the conductance based algorithm perform the worst. Another observation to make is that spectral algorithms with NJW as the first stage tend to degrade slightly less then those with Multicut as the first stage. This was hinted in [8] and is a result of mapping the points on a unit sphere which gets rid of radial variation. .

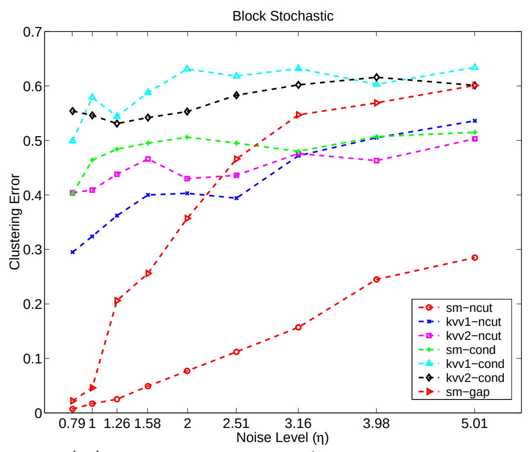
The purpose of this dataset was to demonstrate the robustness to noise. So this is the only dataset on which the error bars are shown (except for the first column in which all methods performed nearly the same with similar error bars. We omitted them to make the graphs more clear).

As we can see in figure 2 (c) and (f) linkage algorithm are too sensitive to noise and infact could not find out the correct clustering even when (almost) no noise was added to the block stochastic matrix. The multiway spectral methods as expected perform the best as this is their perfect S. Within this class All the algorithms seems to perform nearly same with `mcut` methods performing slightly better than the **ang** methods. This suggests that for the block stochastic similarity matrices it might be slightly preferable to use the Multicut base algorithms. The reason for this might be the that NJW maps the clusters to the unit sphere and this might blow up distances between points that are in the same cluster. However the experimental proof is not conclusive.

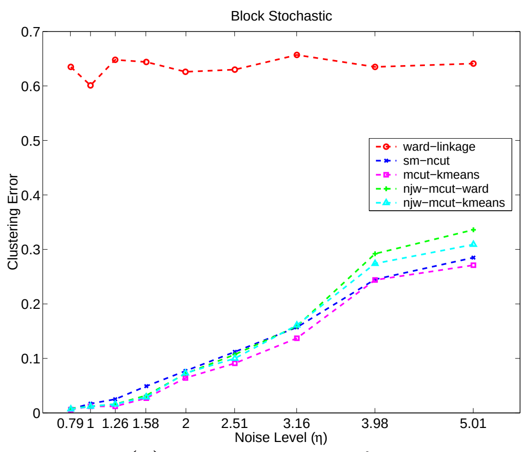
In the recursive algorithms only the `shi-r-ncut` gets the perfect clusters in case of low noise though other variants based on also perform well. It is interesting to note that conductance based performs significantly worse. This is again expected as the the conductance only takes the smaller cluster size into account while $Ncut$ is based on both the cluster sizes.



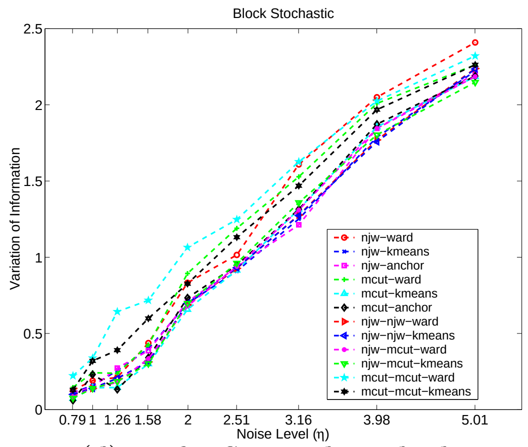
(a) CE for Spectral Methods



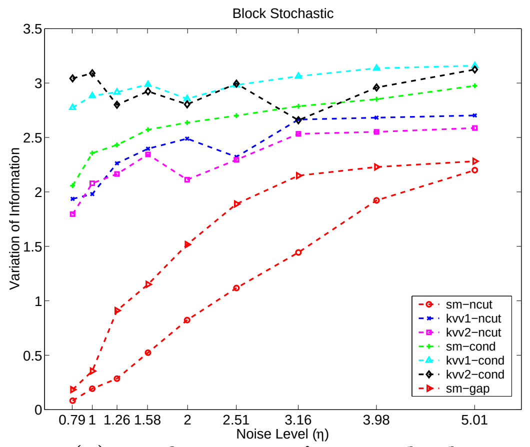
(b) CE for Recursive Methods



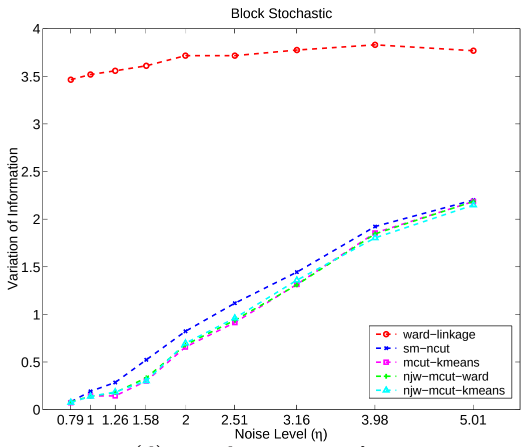
(c) CE for Best Five



(d) VI for Spectral Methods

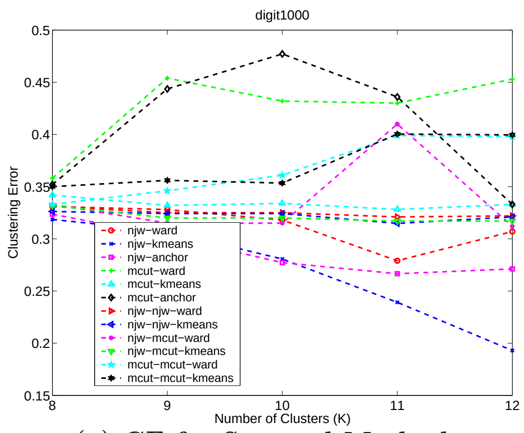


(e) VI for Recursive Methods

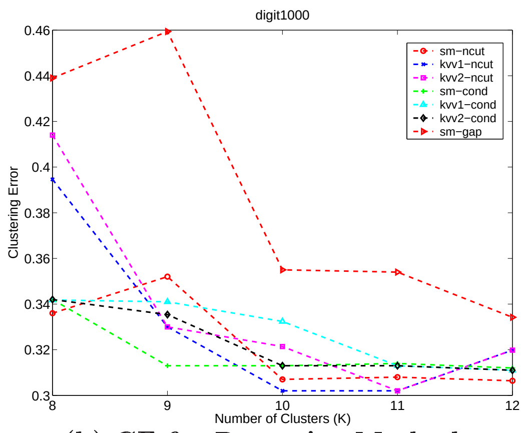


(f) VI for Best Five

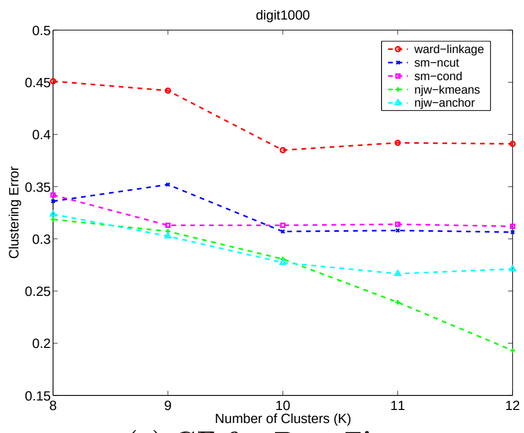
Figure 2: **Block Stochastic Dataset**. The x-axis is the $log_{10}(\eta)$ where η is the noise added.



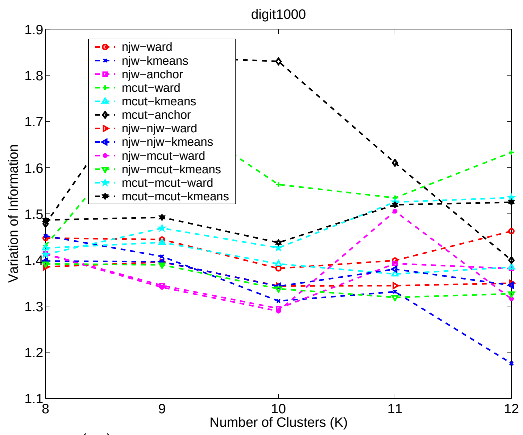
(a) CE for Spectral Methods



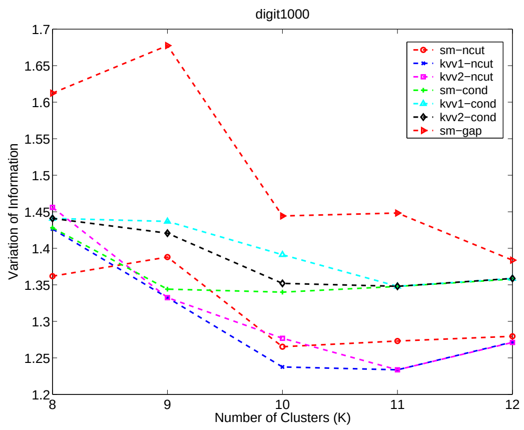
(b) CE for Recursive Methods



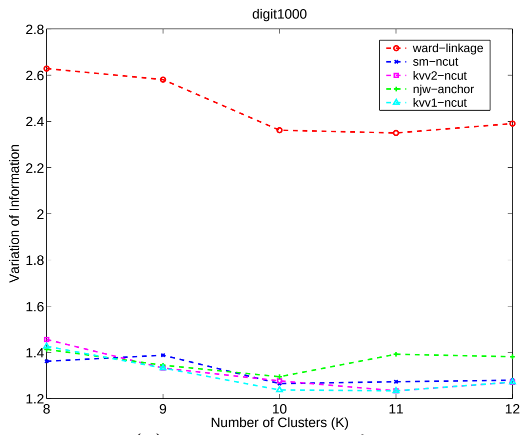
(c) CE for Best Five



(d) VI for Spectral Methods

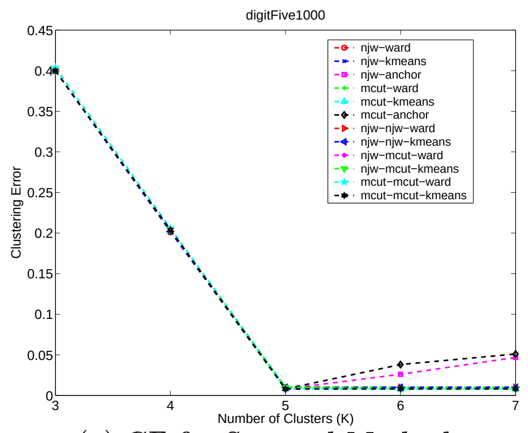


(e) VI for Recursive Methods

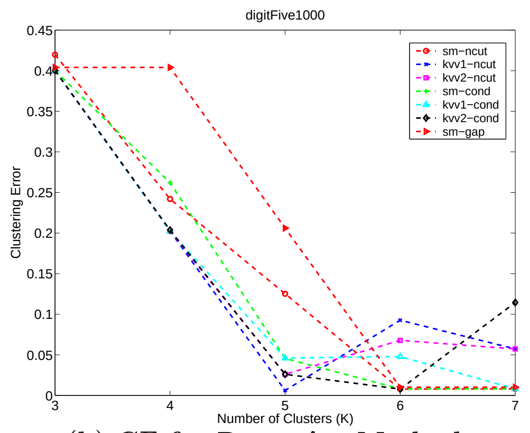


(f) VI for Best Five

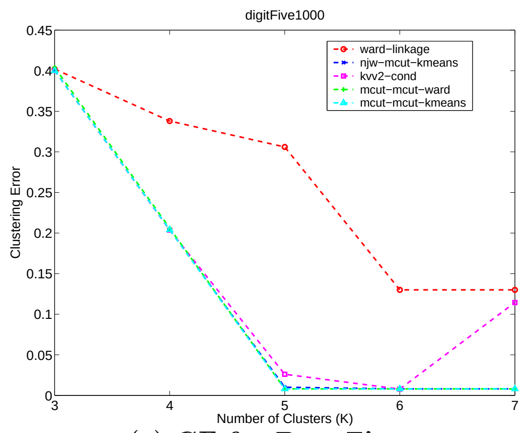
Figure 3: Handwritten digits (`digit1000`)



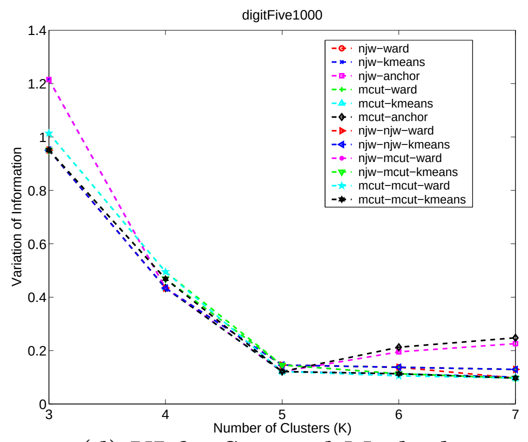
(a) CE for Spectral Methods



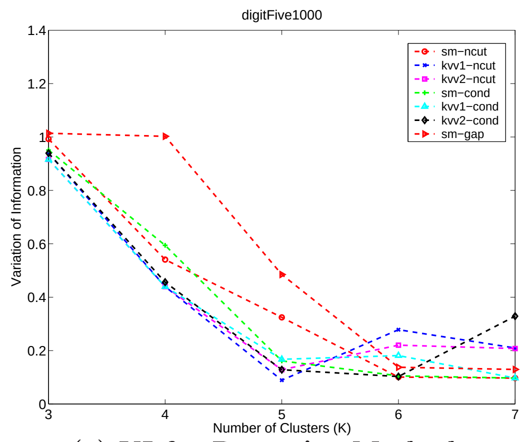
(b) CE for Recursive Methods



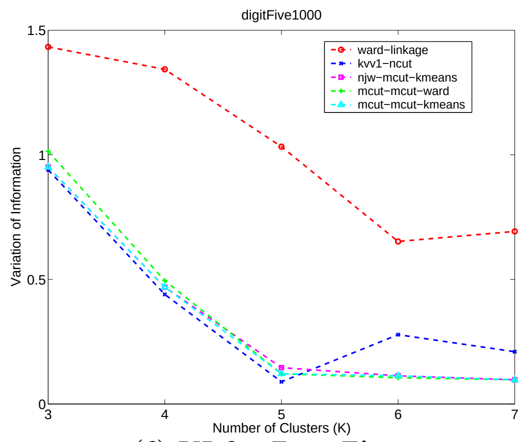
(c) CE for Best Five



(d) VI for Spectral Methods

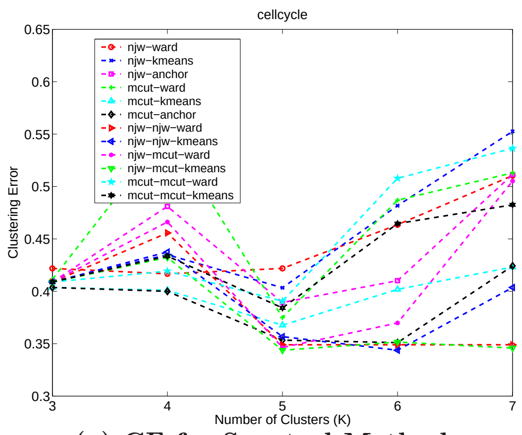


(e) VI for Recursive Methods

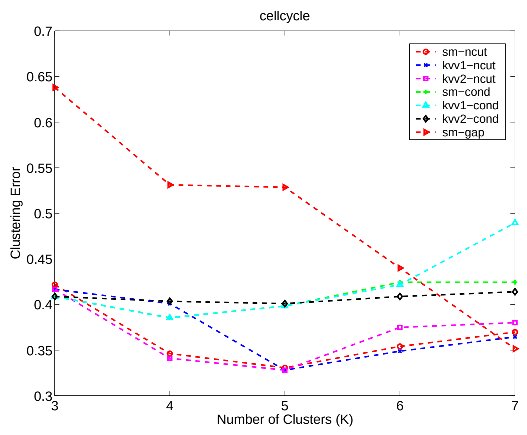


(f) VI for Best Five

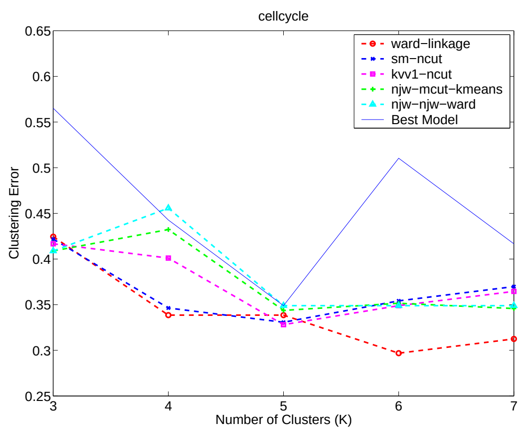
Figure 4: Handwritten digits : Five Digits (0,2,4,6,7) (`digitFive1000`



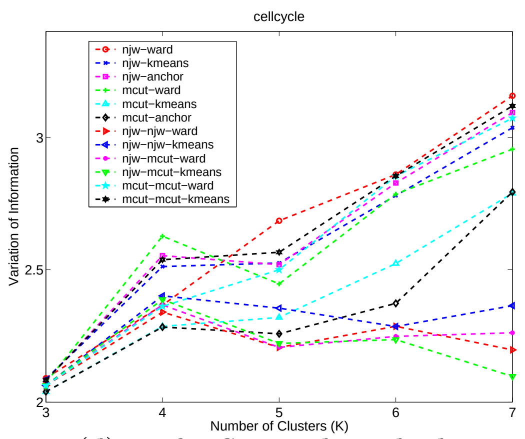
(a) CE for Spectral Methods



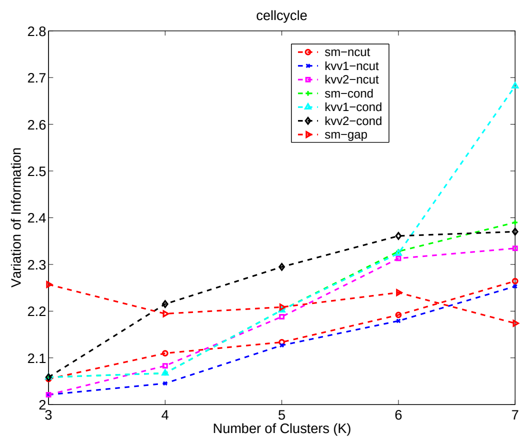
(b) CE for Recursive Methods



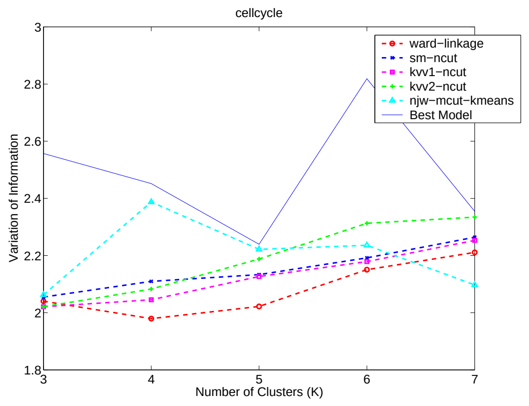
(c) CE for Best Five



(d) VI for Spectral Methods

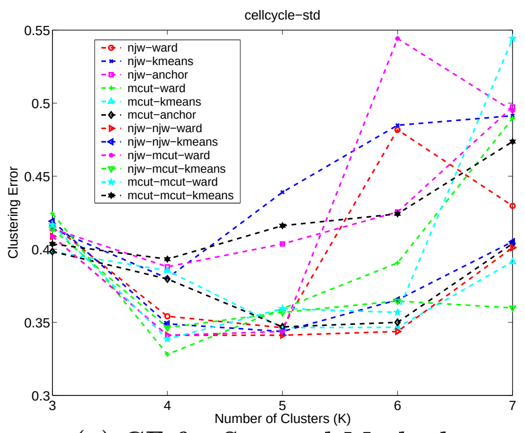


(e) VI for Recursive Methods

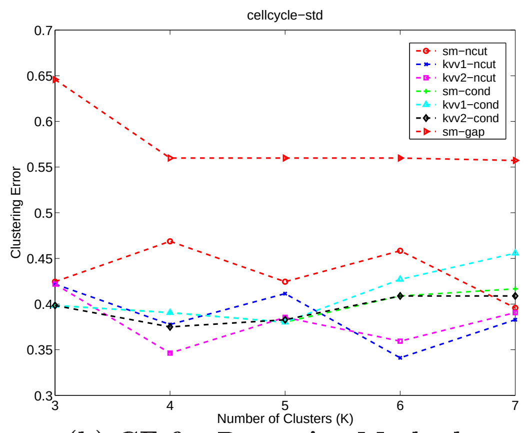


(f) VI for Best Five

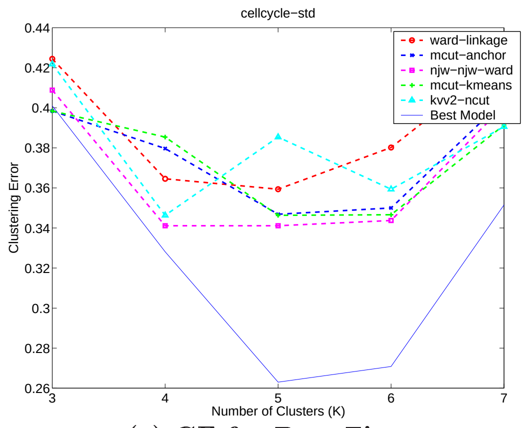
Figure 5: Log normalized yeast cell cycle data (`cellcycle`)



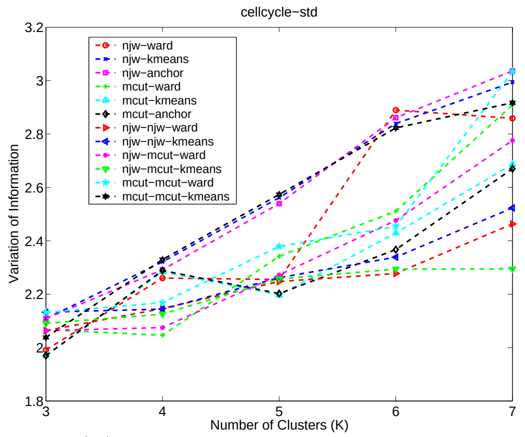
(a) CE for Spectral Methods



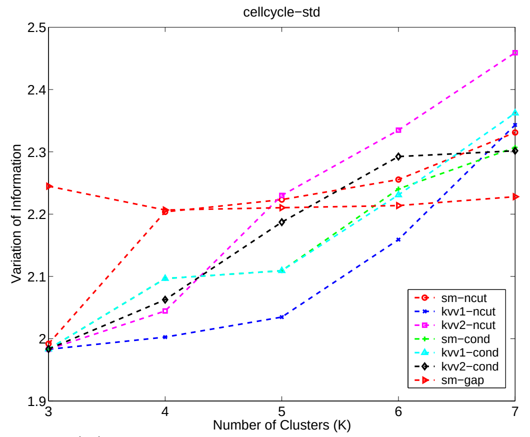
(b) CE for Recursive Methods



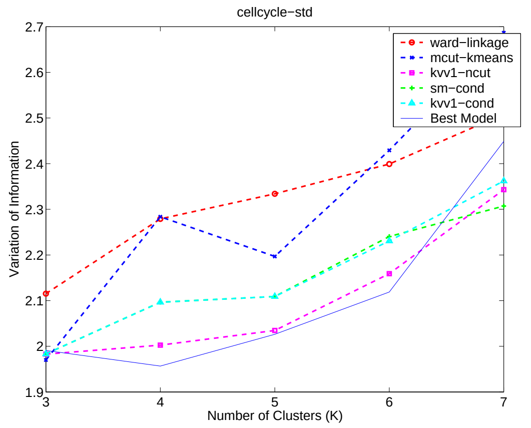
(c) CE for Best Five



(d) VI for Spectral Methods



(e) VI for Recursive Methods



(f) VI for Best Five

Figure 6: Standardized yeast cell cycle data `cellcycle-std`

8.2 Handwritten Digits

This is the first real dataset the we tested the algorithm on. On the complete dataset `digit1000` (figure 3) the multiway spectral methods perform slightly better than the recursive spectral. However the performance difference is not that significant (and in case of the VI measure almost zero) to make any conclusive statement. The linkage algorithms performed a lot worse. Within the multiway spectral algorithms, `mcut_mcut_ward` seems to be the clear winner.

The results on the `digitFive1000` dataset are much more interesting. The performance is near perfect (at $K = 5$) and hence the comparison could be done in light of a dataset with well established structure. If we take a look at figure 4 (a) then is is easy to see that all the multiway spectral algorithm give nearly identical results from $K = 3$ to 5. This is a strong empirical justification of the similarity of the NJW and Multicut which was theoretically proved above. Also in this particular the clusters are obviously well formed as the result in this sections are independent of the grouping algorithm that is used in the third stage.

This is also one dataset in which the multiway spectral methods seem to dominate over the recursive methods. The linkage methods are as expected far behind. One surprising thing observed is that the $Ncut$ methods are lagging behind in the performance as compared to those based on *conductance*.

8.3 Gene Expression Data

This dataset was more interesting of the two real datasets we used. There are a variety of reasons. First of all, since we had results from the model based algorithms for this dataset (from [13]) there was something to compare the spectral algorithms with rather than just amongst themselves. Secondly this contained the same dataset with different data transformation applied to them (See section 4.3), we could see how much the clustering algorithms are dependent on preprocessing.

For the `cellcycle` dataset the best of the spectral algorithms perform slightly better than the model based algorithms. This is encouraging as this shows that spectral methods are competitive even on real dataset and not just the perfect case. The recursive algorithm show similar performance as the multiway algorithms except that $Ncut$ based algorithms are a little better and the conductance based a little worse. While the ordering within the recursive algorithms is expected it is not clear why some of them are better than the multiway algorithms. It is possible that in presence of noise depending on the later eigenvectors is not always the best thing to do and it is better to do the process recursively which ensures that atleast the first few partitions are correct.

However what is even more surprising is the performance of `cluster-ward-linkage`. This simple linkage algorithm gives nearly the best performance on both measures!. We think that in case of such high error rates as we are observing here it is really anybody's game unless there is a dominant structure known to be in the data which corresponds to the clustering algorithm.

In comparison the situation for `cellcycle_std` is completely reversed! The model based algorithm performs the best. The reason for this is that this data transformation is known to fit better to gaussian model and hence the better performance. The performance of best spectral methods remains the same, though the multiway methods perform better now than recursive ones. (with conductance based methods now just slightly worse and even better at $K = 5$.).

8.4 Future work

In this paper we did not address the problem of how to go about choosing the number of clusters. We intend to explore methods which could find the number of clusters based on the data.

There also are two other algorithms that we did not implement for the lack of time.. The first one is another variant of the SM algorithm which theoretically should perform very well on block stochastic matrix. We did not use it because we think it might be too sensitive to noise. The second is a non-spectral method based on single linkage and runt analysis which we expect to be a lot more robust to noise. We wish to explore how using this algorithm as the grouping algorithm after spectral mapping affects the performance of various methods.

9 Conclusion

The goal of the present paper was to analyze comparatively the features of a number of published spectral clustering algorithms. Rather than establishing which of the published algorithms is better, we aimed at evaluating what features make a spectral clustering more valuable.

Because for clustering a data set the "goodness" is in the eye of the beholder, one should look at clustering algorithms not only as competing with each other but also as *complementing* each other's strengths and weaknesses. Hence, a second goal of our research, was to see how different the various algorithms are in their approach.

The answer to the second question is largely negative. The theory predicts that the perfect S for all three algorithms is the same, a result that is strongly supported by the experiments. All algorithms work very well in the cases when S is almost perfect and there is not clear winner in case it is not. We did find the multiway spectral clustering algorithm to be slightly better performing especially when there is structure to be easily found in the data. For the recursive methods we recommend using the $Ncut$ measure over others though other than that there is no clear winner. As compared to other method we showed that spectral methods give competitive performance to the existing methods and are definitely worth further exploration.

Acknowledgements

. I am thankful to Marina Meila, as my collaborator, in the project without whose immense contribution this work would not have been possible, and as an advisor who has been very supportive and thoughtful. Special thanks to Thomas Richardson for providing support for Deepak Verma from NSF grant DMS-9972008. We would also like to thank Ka Yee Yeung for providing the yeast cell cycle dataset and the classifications. And finally to Jayant Madhavan who provided feedback on earlier version of this paper.

Appendix

Proof of Proposition 1

A matrix $A_{n\times k}$ with orthonormal columns and atmost k unique rows would have exactly k unique orthogonal rows.

Proof: First of since A has rank k it has to have k independent rows, which means that it has *exactly* k unique rows.

Now, Rearrange the rows so that the identical rows are next to each other. (This does not affect the result so we assume that A has this property). So now A can written as $A = C_{n\times k}B_{k\times k}$ where B is the matrix with the k unique rows and $C_{ij} = 1$ if the i^{th} row of A is same as the j^{th} row of B. We just need to prove that B has orthogonal rows.

Since the columns of A are orthonormal, $A^TA = I_{k\times k}$. This implies $B^TC^TCB = I$, Now it is easy to see that C^TC is a diagonal matrix (say D). Define $Z = D^{\frac{1}{2}}B$. This gives us $Z^TZ = I$ which means that Z is orthonormal with orthogonal rows. Which proves that B has orthogonal rows. (Premultiplying a matrix with a diagonal matrix just scales its rows). QED.

Consider the algorithms NJW and Multicut . They use the top K eigen vectors which are orthonormal. So if these eigenvectors are piecewise constant w.r.t to a clustering Δ then they have atmost K unique rows and hence would need have exactly K unique rows which would make the S perfect. So all we need is for the eigenvectors to be PC w.r.t Δ. Also note that these rows would be orthogonal.

Proof of Theorem 2 (reverse direction)

Whenever the S is perfect for Multicut it is perfect for NJW and vice versa.

Proof: Let S be perfect for NJW w.r.t. Δ for the first K eigen vectors. We would prove the S is perfect for Multicut as well. (We would continue the notation introduced in section 2.)

Let P be the corresponding stochastic matrix for S. And let P be block diagonal with l blocks (l would have the trivial value 1 when P is not BD). There are two cases possible. First case is when $l \geq K$. This a block diagonal S (with more than K blocks) and is trivially perfect for Multicut .

The second case is when $l < K$. For P with l blocks ones can always rotate the first l eigen vectors such that they are the indicator functions for the block. i.e. $V_{ij} = c_j > 0$ iff point i belongs to block j and zero otherwise. So, $Y_{ij} \neq 0$ iff i belongs to block j. (for $i = 1 \ldots n, j = 1 \ldots l$). This means two ($K$-dim) points in different blocks would have to differ in atleast one dimension. So, if $p, q \in C_s$ (in Δ) then they have to be in the same block, say j. Since S is perfect w.r.t. Δ for NJW this means that rows of Y are equal w.r.t to Δ. Now consider *arbitrary* $p, q \in C_s$ where s is *arbitrarily* chosen from $1, 2, \ldots K$. By the assumption, $y_p = y_p = \tilde{y}_s$(Let). This implies $x_p = |x_p| y_q = |x_p| \tilde{y}_s$ and $x_q = |x_j| y_q = |x_q| \tilde{y}_s$. In particular the j^{th} dimension of the x_p and x_q (which are also in j^{th} column of V) are proportional to $|x_i|$ and $|x_j|$. i.e. $V_{pj} = |x_p| \tilde{y}_s^j$ and $V_{qj} = |x_q| \tilde{y}_s^j$. But we know that $V_{pj} = V_{qj}$ which means that $|x_p| = |x_q|$ and hence $x_p = x_q$. Since p, q, s were chosen arbitarily this proves that S is perfect for Multicut .

References

[1] Sanjoy Dasgupta. Performance guarantees for hierarchical clustering. In *Proceedings of the 15th Annual Conference on Computational Learning Theory (COLT 2002)*, pages 351–363. Springer, 2002.

[2] Ravi Kannan, Santosh Vempala, and Adrian Vetta. On clusterings - good, bad and spectral. In *FOCS*, pages 367–377, 2000.

[3] C. Kaynak. Methods of combining multiple classifiers and their applications to handwritten digit recognition. Master's thesis, Institute of Graduate Studies in Science and Engineering, Bogazici University., 1995.

[4] Marina Meila. Comparing clusterings. Technical Report 418, UW Statistics Department, 2002.

[5] Marina Meila and Jianbo Shi. Learning segmentation by random walks. In *NIPS*, pages 873–879, 2000.

[6] Marina Meila and Jianbo Shi. A random walks view of spectral segmentation, 2001.

[7] Andrew Moore. The anchors hierarchy: Using the triangle inequality to survive high-dimensional data. In *Proceedings of the Twelfth Conference on Uncertainty in Artificial Intelligence*, pages 397–405. AAAI Press, 2000.

[8] A. Y. Ng, M. I. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, pages 849–856, Cambridge, MA, 2002. MIT Press.

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

[10] Daniel A. Spielman and Shang-Hua Teng. Spectral partitioning works: Planar graphs and finite element meshes. In *IEEE Symposium on Foundations of Computer Science*, pages 96–105, 1996.

[11] David L. Wallace. Comment. *Journal of the American Statistical Association*, 78(383):569–576, 1983.

[12] J.H. Ward. Hierarchical grouping to optimize an objective function. *J. Amer. Statist. Assoc.*, pages 236 – 244, 1963.

[13] K. Yeung, C. Fraley, A. Murua, A. Raftery, and W. Ruzzo. Model-based clustering and data transformations for gene expression data. Technical Report UW-CSE-01-04-02, Dept. of Computer Science and Engineering, University of Washington, 2001.

[14] Ka Yee Yeung. Model-based clustering and data transformations for gene expression data. *http://staff.washington.edu/kayee/model/*.