Senior Honors Thesis: Multilinear Subspace Clustering

Eric Kernfeld [1]

Tufts University, Medford, MA, USA

May 5, 2014

[1]Electronic address: `ekernf01@u.washington.edu`; Corresponding author

Abstract

In sections 1 to 4, we consider extensions of the *subspace clustering problem*, which has lately inspired algorithmic developments applicable to computer vision, genomics, music analysis, and electroencephalography. This problem is simple to state: given multivariate data grouped along many low-dimensional subspaces, find the subspaces and the points' memberships. Although some applications of subspace clustering algorithms involve data with matrix structure, recent algorithms disregard that, vectorizing data as part of the processing. We modify the subspace clustering problem using a construct from multilinear algebra, the *tensor product* of subspaces. This allows us to derive new algorithms that do not alter the original form of the data. We will argue that the new algorithms can carry out analysis more quickly and accurately than we could through subspace clustering. We mention applications where it may be possible to exploit this performance improvement.

Our algorithms convert the multilinear subspace clustering problem into a *graph clustering problem*, where every point is represented by a node on the graph and points are connected if they belong to the same subspace. The algorithm we present to solve the modified subspace clustering problem is a randomized algorithm, and multiple trials are needed. Since every trial produces a new realization of the graph, we use sections 5 through 8 to consider various ways of combining graph realizations. We conduct tests on simulated data in order to evaluate them.

Contents

Part I: Multilinear Subspace Clustering

1 Introduction

Most clustering algorithms seek to detect disjoint clouds of data. However, in high-dimensional statistics, data can become very sparse, and these types of methods have trouble dealing with noise. In fact, a completely new approach to the geometry of clustering has recently made headway in the analysis of high-dimensional data sets. Called subspace clustering, this approach assumes that data come from subspaces offset at angles, rather than from clouds offset by gaps. Applications have included detection of tightly correlated gene clusters in genomics [1], patient-specific seizure detection from EEG data [2], and image segmentation [3].

All subspace clustering methods must embed data in $\mathbb{R}^n$. However, in some of the high-dimensional data sets where subspace clustering has been applied, the initial structure of the data is not a vector but rather a matrix or tensor (multi-way array). Examples include the auditory temporal modulation features in [4], the image patches in [3], and raw EEG data under the "sliding-window approach" [2]. We seek to develop a clustering method that incorporates the geometric innovation of subspace clustering without vectorizing these higher-order arrays. To do this, we formulate an algebraic generative model for the data, along with methods for inference. We include experiments on simulated data in order to validate our inference methods.

1.1 The Subspace Clustering Problem and a Multilinear Variant

Mathematically, we describe the subspace clustering problem as follows. Given a list of points x_n, $n = 1...N$, suppose each point is an element of one of the K subspaces. The problem is to decide membership for each of the N points. For simplicity, we treat K as known, although our algorithm could be modified to relax this constraint.

In order to take advantage of patterns in two-way data, we modify the assumptions of the subspace clustering problem. Rather than modeling the data as a union of subspaces, we assume they come from a union of *tensor products* of subspaces. Given subspaces $\mathcal{U} \subset \mathbb{R}^n$ and $\mathcal{V} \subset \mathbb{R}^m$, suppose the columns of $\mathbf{U}$ form a basis of $\mathcal{U}$ and likewise for $\mathbf{V}$ and $\mathcal{V}$. The tensor product $\mathcal{U} \otimes \mathcal{V}$ is the set $\{\mathbf{A} | \mathbf{A} = \mathbf{U}\mathbf{Y}\mathbf{V}^T\}$, where $\mathbf{Y}$ is a $\dim(\mathcal{U}) \times \dim(\mathcal{V})$ matrix. In other words, this is a set of matrices with (column/row) space confined to $(\mathcal{U}/\mathcal{V})$. We call this the multilinear subspace clustering (MSC) problem.

As a summary, we step through the process of creating the data below. A key feature of both of these algorithms is that instead of generating synthetic data (x or $\mathbf{X}$) directly, they first generate a representation in a different set of coordinates (y or $\mathbf{Y}$). In machine learning and statistics, y and $\mathbf{Y}$ are known as *latent variables*. Furthermore, in a model such as these, y and x may differ in dimension; the dimension of y is known as the *latent dimensionality*, *intrinsic dimensionality*, or *data dimensionality*, whereas the dimension of x is known as the *ambient dimensionality.*

Algorithm 1: Subspace Clustering Data Generation: N points, K clusters of latent dimension d and ambient dimension D

```
Given $\{\mathbf{U}_1, ..., \mathbf{U}_K\} \in \mathbb{R}^{D \times d}$,
Repeat $N$ times:
    Draw $k$ from $\{1, ..., K\}$
    Draw a random length-$d$ vector $y_n$
    Compute datum $x_n = \mathbf{U}_k y_n$
```

We now discuss this generative process in the context of similar models from statistics and signal processing. As we compare with other work, we argue that our model may be sensible in the context of certain natural data types.

Algorithm 2: MSC Data Generation: N points, K clusters of latent dimension $d_v d_u$ and ambient dimension $D_v D_u$

```
Given $\{\mathbf{U}_1, ..., \mathbf{U}_K\} \in \mathbb{R}^{D_u \times d_u}, \{\mathbf{V}_1, ..., \mathbf{V}_K\} \in \mathbb{R}^{D_v \times d_v}$
Repeat $N$ times:
  Draw $k$ from $\{1, ..., K\}$
  Draw a random $d_u$ by $d_v$ matrix $\mathbf{Y}_n$
  Compute datum $\mathbf{X}_n = \mathbf{U}_k \mathbf{Y}_n \mathbf{V}_k^T$
```

2 Comparison with other work

To begin comparing this project with previous literature, we introduce some ideas from statistics.

Definition A list of vectors x_1, x_2, ... x_N with mean $\overline{x}$ has this covariance matrix.

$$\mathbf{C} = \frac{1}{N-1} \sum_{n=1}^{N} (x_n - \overline{x})(x_n - \overline{x})^T$$

Definition A list of vectors has separable covariance if its covariance matrix can be written as Kronecker product of smaller matrices.

For readers not familiar with the Kronecker product, we define it here.

Definition Given matrices $\mathbf{U} \in \mathbb{R}^{m \times n}$ and $\mathbf{V} \in \mathbb{R}^{p \times q}$, then their Kronecker product is denoted by $\mathbf{U} \otimes \mathbf{V}$. It is an element of $\mathbb{R}^{mp \times nq}$, and it is given by

$$\begin{bmatrix} u_{11}\mathbf{V} & u_{12}\mathbf{V} & \dots & u_{1n}\mathbf{V} \\ u_{21}\mathbf{V} & u_{22}\mathbf{V} & \dots & u_{2n}\mathbf{V} \\ \vdots & \vdots & \ddots & \vdots \\ u_{m1}\mathbf{V} & u_{m2}\mathbf{V} & \dots & u_{mn}\mathbf{V} \end{bmatrix}.$$

It is not immediately apparent that our model has anything to do with separable covariance. However, according to the following theorem from [5], every individual subspace in our model exhibits separable covariance (given some assumptions on the distribution of the latent variables). In this theorem, vec($\cdot$) maps an $n \times n$ matrix to an $n^2 \times 1$ vector by transposing and lexicographically stacking the rows (indexing from 0, $\text{vec}(\mathbf{X})_{\text{j+ni}} = \mathbf{X}_{\text{ij}}$).

Theorem 2.1 (Eq (1), [5]) Given fixed $\mathbf{U} \in \mathbb{R}^{D_u \times D_u}$ and $\mathbf{V} \in \mathbb{R}^{D_v \times D_v}$, suppose a random matrix $\mathbf{Y} \in \mathbb{R}^{D_v \times D_u}$ has uncorrelated, zero-mean, unit-variance entries. If $\mathbf{X} = \mathbf{U}\mathbf{Y}\mathbf{V}^T$, then $\text{Cov}[\text{vec}(\mathbf{X})] = \mathbf{U} \otimes \mathbf{V}$.

The main difference between $\mathbf{X} = \mathbf{U}\mathbf{Y}\mathbf{V}^T$ in this theorem and $\mathbf{X} = \mathbf{U}\mathbf{Y}\mathbf{V}^T$ in the MSC problem is that in the theorem, $\mathbf{U}$ and $\mathbf{V}$ are square and $\mathbf{Y}$ is the size of $\mathbf{X}$, while in MSC, $\mathbf{U}$ and $\mathbf{V}$ are rectangular and $\mathbf{Y}$ is smaller than $\mathbf{X}$. Augmenting our rectangular $\mathbf{U}$ and $\mathbf{V}$ with zeros allows us to apply this result to our model.

Separable covariance models are thought to be appropriate, for instance, in neurometry, where one covariance structure is associated with autocorrelation in individual brain areas over time and another captures correlations between regions frozen in time. Along with basic methodology research on estimation and hypothesis testing [6, 7], models with separable covariance have been applied to multivariate repeated measures data [8], electroencephalography [9], and relational data on international trade [5].

To expand the scope of this comparison towards machine learning and signal processing, we introduce the idea of Principal Component Analysis (PCA). PCA is used to reduce data dimensionality for tasks like visualization. In PCA, the data are projected onto the axes of greatest variance, and their representation in those coordinates is used as a proxy for the original data. PCA takes advantage of covariance structure: it can be shown that the subspace used in PCA corresponds with eigenvectors of the covariance matrix whose eigenvalues are maximal. PCA has also been phrased as maximum likelihood estimation of a subspace in a probabilistic model known as PPCA [10]. In this view, it is a simple counterpart to subspace clustering methods, positing just one subspace instead of many. Indeed, among early subspace clustering algorithms were some that employed Bayesian mixture modeling using PPCA for individual clusters [11].

Thus far, we have mentioned covariance, single-subspace inference (PCA), and multiple-subspace inference (subspace clustering). We have also mentioned separable covariance, and we are proposing a multiple-subspace model where every cluster has separable covariance. The second list of topics, to properly parallel the first, ought to mention a single-subspace model with separable covariance structure. Indeed, this has appeared in the signal processing literature under the name 2DPCA [12]. Although [12] does not mention the term separable covariance, Hoff [5] makes explicit connections between separable covariance matrices and the *Tucker model* of array decompositions, which is used in [12]. For the sake of thoroughness, we also comment that 2DPCA is only one of several PCA variants that are phrased in terms of decompositions of three- (or more) way arrays. Another is called *low multi-rank approximation*; algorithms for that task appear in [13, 14, 15]. The review [16] introduces a host of PCA-like approximate array decompositions, while [17] and [18] showcase more recent work.

3 Subspace Clustering: Inference

Approaches to the subspace clustering problem have variously relied upon information theory, algebra, statistics, and geometry. For a review of many subspace clustering methods, see [19]. More recent work includes [20] and [21]. In this work, we confine our focus to one recent method. Named Subspace Clustering via Thresholding (TSC), it was proposed and described in [21, 22]. It is relatively cheap, with the dominant cost scaling as (DN^2) for N data points of dimension D, but it has an impressive list of qualifications. Probabilistic analysis shows that TSC is robust to additive noise, that it provably detects outliers, and that it can still work with incompletely observed data. The TSC authors confirm their theoretical results with experiments on synthetic data, and they perform competitively with many other methods on real data.

The general scheme of TSC is to construct a weighted graph where two points share a highly weighted edge if they belong to the same subspace. Then, TSC looks for approximate connected components on the graph. In order to make this precise, we discuss two important subroutines, spectral clustering and K-means, in the following subsections.

3.1 The K-means algorithm

The K-means algorithm is a machine learning workhorse. Proposed in the 1960s [23], it is simple and well-understood. The K-means scheme, to acquire a sensible K-way partition of the data, begins by initializing K cluster centers at random locations. Then, an alternating iteration begins. Each round, every data point is assigned to the nearest cluster center, and then every cluster center is moved to the mean of the data points belonging to it. As of April 2014, the process is wonderfully illustrated on the website of Andrey Shabalin.

K-means converges in a finite number of steps to a local minimum of $\sum_{k=1}^{K}\sum_{n=1}^{N} 1_{n,k}\|x_n - c_k\|^2$, where c_k is a cluster center, x_n is a datum, and $1_{n,k}$ is one if datum n is in cluster k and zero otherwise [24]. However, research such as [25] indicates that this may not be the right metric to optimize. Quoting from its abstract, "In high dimensional space, the concept of proximity, distance or nearest neighbor may not even be qualitatively meaningful." Thus, various strategies use different metrics [26], regularization [24], and dimensionality-reducing projections [27] in order to improve results.

3.2 Spectral Clustering for Weighted Graphs

A different perspective on clustering begins with a weighted graph, rather than with a collection of points in Euclidean space. Called spectral clustering, this algorithm is so effective that many clustering algorithms convert vector-space data into a weighted graph in order to apply the method. Some examples of this conversion from Euclidean space to graphs use edge weights based on proximity [28], whereas others use more complicated tricks in order to function as subspace clustering algorithms [29, 30, 20]. TSC does this as well. Consequently, we will sometimes refer to the data points as nodes or vertices of the graph.

Suppose we have a weighted, undirected graph V with a symmetric adjacency matrix $\mathbf{C}$. If there are N data points, then $\mathbf{C}$ will be $N \times N$ and symmetric with nonnegative real entries. The magnitude of the i, j entry indicates the strength of the connection between data points i and j. We define the degree d_i of vertex

i to be a sum of weights over all incident edges (edges between i and some other node). So, $d_i := \sum_j \mathbf{C}_{ij}$. The degree matrix is a diagonal matrix such that $\mathbf{D}_{ii} = d_i$.

Much of this subsection is abbreviated from [31]; in line with their terminology, we define a few more quantities.

Definition Given a subset of vertices A, the cut function Cut(A) is $\frac{1}{2}\sum_{i\in A, j\notin A} \mathbf{C}_{ij}$.

The cut function measures the number of edges a partition violates.

Definition Given a subset of vertices A, its volume is $\mathrm{Vol}(A) := \sum_{i\in A} d_i$, the sum of the degrees of its nodes.

The volume function can be used to help balance cluster sizes.

Definition Given a partition of a graph into into $A_1, ..., A_K$, the Ncut functional is defined as $\mathrm{Ncut}(A_1, ..., A_K) := \sum_{k=1}^{K} \mathrm{Cut(A_k)/Vol(A_k)}$.

Definition Given a partition of a graph into into $A_1, ..., A_K$, the indicator vector $\mathbb{1}_{A_k}$ has 1 for the ith entry if node i is in A_k. Otherwise, the ith entry is zero.

The Ncut objective function provides a sensible tradeoff between respecting the graph structure and balancing the clusters, because it grows both as edges are violated and as clusters shrink. (Some authors argue that it could be replaced by a better objective [32].)

In her tutorial, Luxburg [31] shows how, given a K-way segmentation of the data $\{A_1, ..., A_K\}$, one version of spectral clustering attempts to minimize the Ncut functional. This view of the algorithm begins with the construction of scaled indicator vectors for the clusters: $h_k := \mathrm{vol}(A_k)^{\frac{-1}{2}}\mathbb{1}_{A_k}$. Taking them together as a matrix $\mathbf{H} = [h_1, ..., h_K]$, certain properties can be written succinctly. The weighting and the disjointness of the clusters implies $\mathbf{H}^T\mathbf{D}\mathbf{H} = \mathbf{I}$, while the objective can be written as $\mathrm{Ncut}(A_1, ..., A_K) = \mathrm{Tr}(\mathbf{H}^T(\mathbf{D}-\mathbf{C})\mathbf{H})$. The matrix in the middle is called the unnormalized graph Laplacian, defined as $\mathbf{L} := \mathbf{D} - \mathbf{C}$.

It is NP-hard to optimize this over the set of all segmentations, but we can make the problem tractable by "relaxing" the search. Instead of explicitly constructing the indicator vectors, we just impose the constraint $\mathbf{H}^T\mathbf{D}\mathbf{H} = \mathbf{I}$. Then the optimal choice of $\mathbf{H}$ (optimal with respect to the Ncut penalty) consists of the K eigenvectors of $\mathbf{L}_{rw} := (\mathbf{I} - \mathbf{D}^{-1}\mathbf{C})$ with the smallest eigenvalues. $\mathbf{L}_{rw}$ is one of the normalized Laplacians defined in [31]. Spectral clustering projects the data onto the subspace spanned by those eigenvectors before clustering by some other method, with the idea that once something close to indicator vectors is obtained, a simpler method such as K-means will likely succeed.

3.3 An Algorithm for Subspace Clustering

As we mentioned in previous sections, TSC [22] turns the subspace clustering problem into a clustering problem where each data point is represented by a node on a weighted graph. TSC uses the inner product between two data points as the edge weight. The idea is that when points share a subspace, the inner product between them is probably higher than otherwise. One key alteration from TSC's predecessors is the thresholding step: only the q largest edges are preserved. Thresholding is meant to filter out noise from the graph. We outline the procedure here in Algorithm 3.

4 Multilinear Subspace Clustering: Inference

A tensor product of subspaces of $\mathbb{R}^n$ and $\mathbb{R}^m$, for example the set of n by m matrices with nonzeroes confined to the 5 by 5 block in the upper left, is itself still a subspace of $\mathbb{R}^{n\times m}$. It would be possible to cluster the data under our model using a traditional subspace clustering method. However, we hope to develop a method that, rather than simply being compatible with our assumptions, will actively make use of them. To do so, we turn to the matrices' column and row spaces, presenting a fast randomized algorithm that chooses rows and columns instead of processing each entire data matrix. In Section 5, we justify this new method by demonstrating that it performs better than TSC on simulated data.

Algorithm 3: TSC (Subspace Clustering by Thresholding)

```
Inputs:
  Vectors $x_1$, ..., $x_N$ of dimension $D$ arranged as columns of a matrix $X$
  The desired number of clusters $K$
  A parameter $q$ (a natural number)
Outputs:
  A vector of length $n$ where the $i$th entry is a cluster label, from 1 to $K$, for datum $x_i$.
Procedure:
  Normalize each data point to unit length.
  Compute $\mathbf{C} = \mathbf{X}^T\mathbf{X}$, the matrix containing all inner products of the data vectors.
  Replace negative entries with their absolute values.
  For each row of $\mathbf{C}$, set all but the $q$ highest elements to zero.
  With the new sparse $\mathbf{C}$, perform normalized spectral clustering [33]. In other words:
    Compute $\Lambda$, a diagonal matrix with row sums of $\mathbf{C}$. Form the Laplacian matrix $\mathbf{L} = \mathbf{I} - \Lambda^{-1}\mathbf{C}$.
    Compute the leading $K$ eigenvectors of $\mathbf{L}$ and place them as columns of a matrix $\mathbf{P}$.
    Let $p_i$ be the vector corresponding to the $i$th row of $\mathbf{P}$.
    Cluster the points $p_1$,...,$p_n$ with the K-means algorithm [23].
```

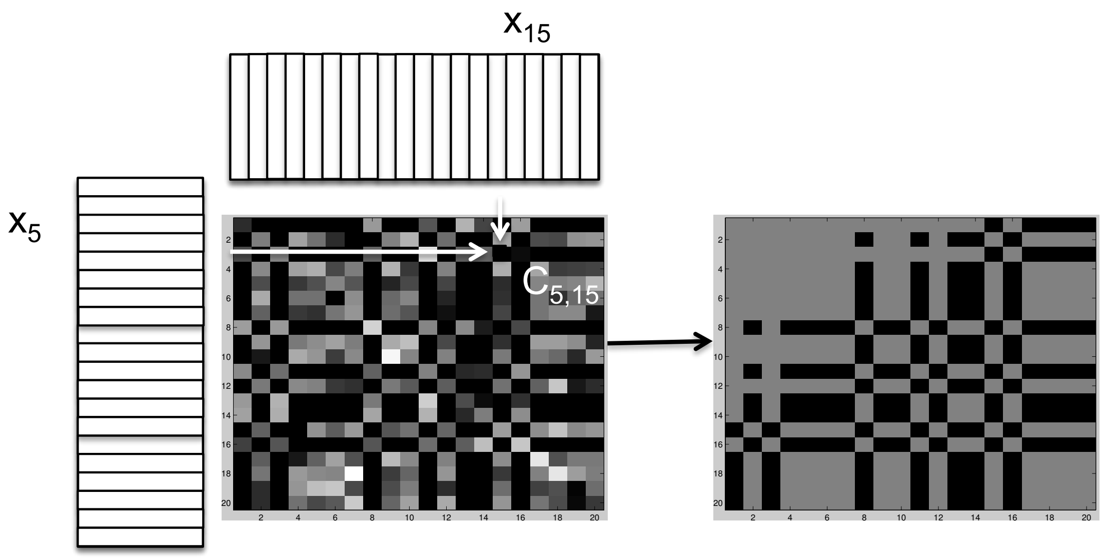

Figure 1: A schematic of the TSC algorithm, discussed in Algorithm 3 and Section 3. In this graphic, entries with high values are darker. TSC first fills an adjacency matrix with inner-product similarities, then thresholds its entries to reveal block structure. (Block structure in adjacency matrices corresponds to connected component structure in graphs.) The post-processing step, spectral clustering, is not depicted here.

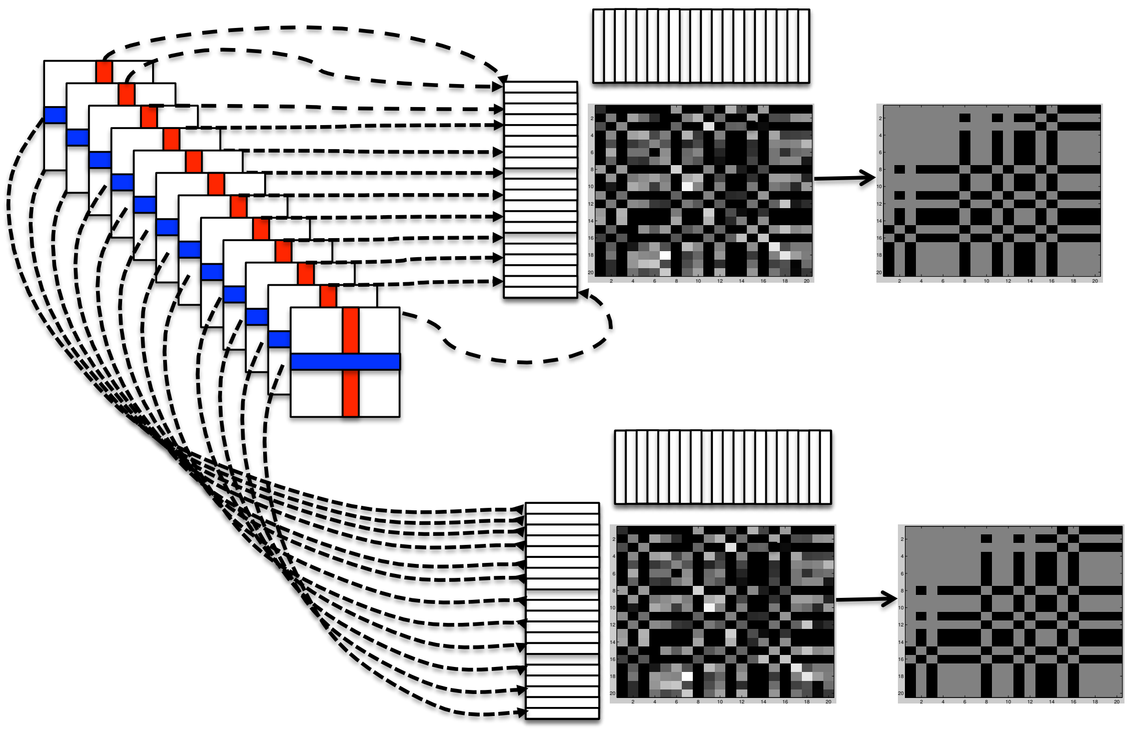

Figure 2: In a single run of the randomized MSC algorithm, we first choose a row from each data point. We then assemble the rows and run TSC (see Figure 1). We do the same for the columns.

Given a collection of data matrices $\{\mathbf{A}_n\}_{n=1}^{N}$, generated according to our model, the columns $\mathbf{A}_n(:,j)$ for all $j = 1, 2, .., D_v$ and for all $n = 1, 2, ..., N$ (N is the number of data points) come from the union of subspaces $\mathcal{U}_1 \bigcup ... \bigcup \mathcal{U}_K$ (and the rows come from $\mathcal{V}_1 \bigcup ... \bigcup \mathcal{V}_K$). So, one expensive strategy to first identify these individual subspaces would be to perform a subspace clustering algorithm on the $D_v N$ columns and $D_u N$ rows.

This would be computationally demanding, especially as many potential applications (e.g. image segmentation) involve large volumes of data. A more feasible strategy to produce affinities is to randomly pick a column and a row from each of $\mathbf{A}_n$, $n = 1, 2, ..., N$. TSC could be run on these collections (one collection has N rows and the other has N columns). We can find a typical affinity matrix $\mathbf{C}$ by doing this experiment many times over the random choices of the columns and the rows. We address the number of trials in section 4.2, and we outline the procedure in Algorithm 4.

With either the naive algorithm or the randomized variant, we could encounter ambiguities: what if the rows and columns give different clusterings? What if two rows from the same data matrix end up in different clusters? Fortunately, the modularity of TSC helps resolve these dilemmas: we halt it before spectral clustering and reduce the graph whose nodes represent columns or rows to a graph whose nodes are the N original data points. In Part II, we discuss the different strategies to do this.

4.1 Algorithmic Complexity and Scalability

The dominant cost of TSC is $\mathrm{O}(DN^2)$ for N data points of dimension D. Suppose we have N data matrices $\mathbf{A}_n$, $n = 1, 2, ..., N$, and suppose each one is D_c by D_r. To run the randomized algorithm from 4, suppose we run T trials. Then we pay $\mathrm{O}(D_c N^2)$ for one round of TSC and $\mathrm{O}(D_r N^2)$ for the next. Those two, multiplied by T, gives the dominant component the cost. For comparison, vectorizing the data and running TSC costs $\mathrm{O}(D_r D_c N^2)$. This differs from the expression in Section 3 because once a datum is stretched into a vector, it has length $D_r D_c$.

Thus, if MSC can succeed with few trials, it will have an advantage especially in situations where each

Algorithm 4: Multilinear Subspace Clustering (2-way data examples)

```
Given data examples $\mathbf{A}_n$ with columns of dimension $D_c$ and rows of dimension $D_r$:
    Fill $D_c \times N$ matrix $\mathbf{X}_{cols}$ by choosing a column from each datum $\mathbf{A}_n$ and inserting it as column $n$ of $\mathbf{X}_{cols}$.
    Generate $\mathbf{C}_{cols} \in \mathbb{R}^{N \times N}$ by running a subspace clustering algorithm on $\mathbf{X}_{cols}$.
    Fill $D_r \times N$ matrix $\mathbf{X}_{rows}$ by choosing a row from each datum $\mathbf{A}_n$, transposing it, and inserting it as column $n$ of $\mathbf{X}_{rows}$.
    Generate $\mathbf{C}_{rows} \in \mathbb{R}^{N \times N}$ by running a subspace clustering algorithm on $\mathbf{X}_{rows}$.
Repeat the procedure $T$ times, saving output, and consult Part II for various methods of combining the list of $\mathbf{C}_{cols}$ and $\mathbf{C}_{rows}$ into a single $\mathbf{C}$.
Run spectral clustering on $\mathbf{C}$ and apply the results to the original $N$ data examples.
```

datum is large. The number of trials is addressed in Section 4.2 and Figure 3. Meanwhile, the post-processing step scales the same way as TSC's: using a simple method that works well in experiments in Part II, we will ultimately need only the leading K eigenvectors of an N by N matrix. (This will let us detect up to K clusters.)

Some results, such as Theorem 3 of [22], show that clustering is more difficult when the clusters have high latent dimension. With that in mind, one advantage of MSC is that it works in the column or row spaces, where the latent dimension is only d_c or d_r rather than the product $d_r d_c$.

4.2 Experiments on Synthetic Data

Synthetic data were generated lying along three separate tensor products of subspaces. Latent dimensionality was 10×10 and ambient was 100×100, with ten data points per subspace. (Including more points may change the outcomes, but it would be computationally expensive for experiments that we repeat many times). Isotropic Gaussian noise of standard deviation 0.2 was added in the ambient space. The randomized version of MSC was run on 100 data instantiations, along with TSC as a baseline. Total clustering error (number of nodes misclassified) is shown at each number of trials. The same data instantiation was used to run every method in a given trial. Results are shown in Figure 3.

Of the graph-merging options, which are detailed in Part II, simple options in which all the graphs are added together perform best. After 4 trials, MSC outperforms TSC, with performance plateauing at about 1/3 the clustering error of TSC after 8 trials. This means that for data conforming to our model, MSC will have a definite advantage for sizes greater than 8×8.

We also compared the methods under various amounts of noise (Figure 4). Performance deteriorate slowly as noise amplitude increases. The same two graph-merging options perform well, and TSC performs worse than them across the board. This is surprising, given that the synthetic data fit the underlying model that TSC assumes. We hypothesize that TSC performs poorly because there are relatively few points per cluster. MSC compensates for this in two ways. First, MSC explicitly operates in lower-dimensional space, where fewer points are needed to fill out a subspace. Second, by acting on individual rows and columns, MSC effectively amplifies the number of points.

Part II: Clustering Using Multiple Random Graphs

5 Other Work in Graph Clustering

Given this scheme, one question raised in our paper is in finding a suitable way to aggregate information when condensing the graph. Work such as the chapter "Bayesian Methods for Graph Clustering" from [35] and [36] takes a Bayesian approach, and the latter is even designed to handle multiple graph realizations. However, we plan to use spectral clustering, and we believe there is merit in this line of investigation. As pointed out in [31], people do not necessarily use spectral clustering to get the best results. They use it partly because it is easy to understand, relatively fast, and already popular. Furthermore, the area is developing rapidly,

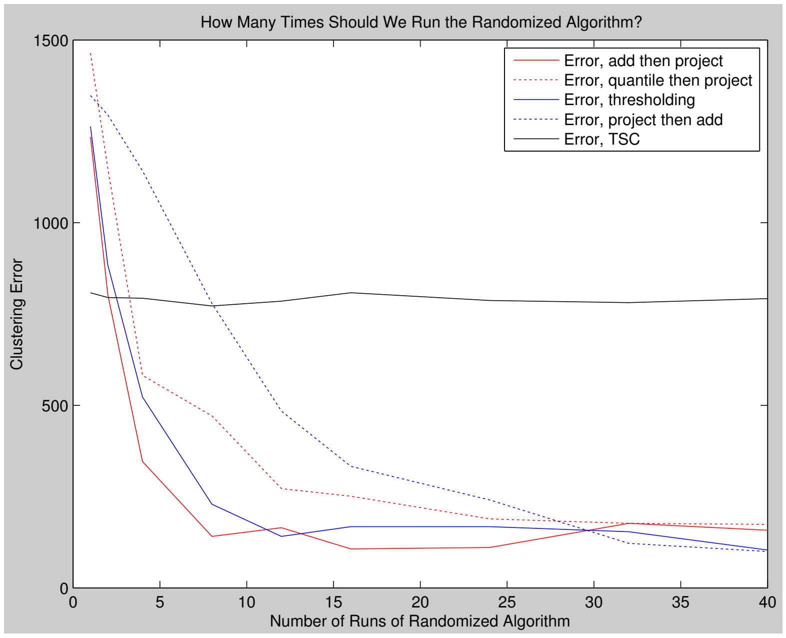

Figure 3: Total clustering error when running the randomized method different numbers of times. Considering how TSC and MSC scale, MSC will have a definite speed advantage for sizes greater than 8×8. Performance of MSC is shown under several different ways of merging the resulting graphs (see Part II for details). Section 4.2 describes experiments in detail.

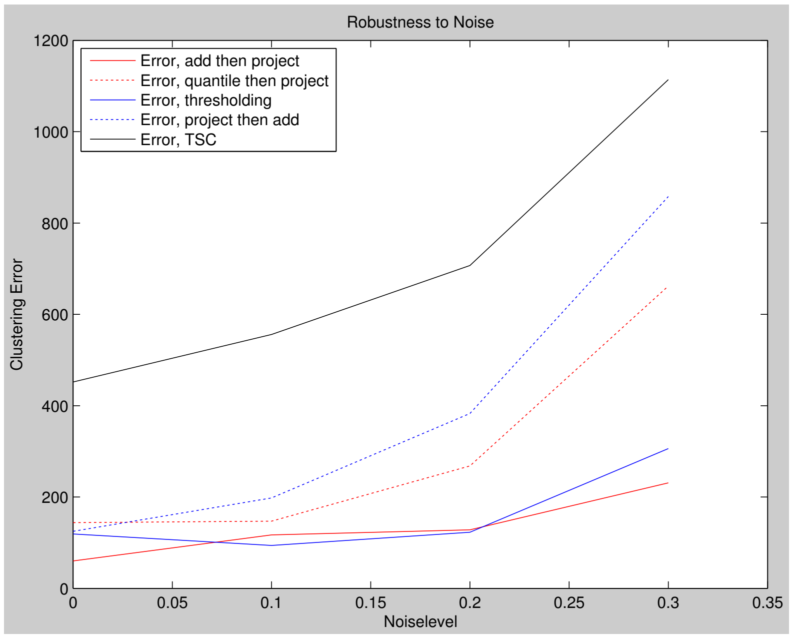

Figure 4: Performance of MSC under several different ways of merging the resulting graphs (see Part II for details). The plot shows total clustering error when running the randomized method 12 times with noise as indicated on the horizontal axis. MSC performs better than TSC, with simple post-processing methods based on adding together edge weights performing best. Section 4.2 describes experiments in detail.

and it carries potential for continuing research. In light of these facts, characterizing variations on spectral clustering is important despite the existence of sophisticated models that may have better performance.

Although there is extensive literature giving theoretical guarantees on spectral clustering, none of the work thus far does exactly what we are trying to do. Much existing work such as [37] deals with consistency in a model where increasingly large samples are drawn from a manifold, the result being a matrix that increases in size (ours is constant). In another work [38], an ideal matrix is corrupted by subgaussian noise with zero expectation and some dependency among the entries (our matrix may not have zero-expectation corruptions). McSherry [39] deals with a simple random graph model including latent "communities" (connected components) and false positive and false negative probabilities. However, he uses a different spectral algorithm from those typically paired with subspace clustering. Finally, in the setting most similar to our model, the paper [40] studies performance of spectral clustering on graphs generated via a specific parametric model (the stochastic blockmodel). This model generalizes the one from [39], and it captures our situation well but for the fact that it produces unweighted edges. In [40], Rohe et al. give results on clustering of large, finite networks.

Most of this work deals with a model where only one sample of the random matrix is available. By contrast, we face the following question: given multiple samples of a fixed-size matrix, each corrupted by perturbations that may have dependencies between the entries and non-zero expectations, how can we best combine the samples to improve results? We answer this question via empirical study of some candidate procedures.

6 List of Candidate Methods

We begin by listing some clustering options below.

- Addition of edges: we could add the adjacency matrices together.
- Quantiles: to filter out false positives, we could choose the second (in general, lth) highest value from the T trials at each position in the adjacency matrix.
- Thresholding: after adding together the adjacency matrices, we could take the q highest weight values in each row in order to filter out false positives.
- Optimization: we could define an objective function and attempt to minimize it.
- Spectral filtering: before adding together the graph realizations, we could "filter" each individual data matrix *à la* [39] by projecting onto the K leading eigenvectors.

In order to better understand these options, we refer the reader to section 3.2, which introduces one of the derivations that motivates spectral clustering. To summarize, we can view spectral clustering as a relaxation of a hard optimization problem, and the original problem can be interpreted simply in terms of quantities such as the number of between-cluster edges and the size of clusters.

7 Discussion of Candidate Methods

7.1 Addition of Edges, Quantiles, and Thresholding

One strategy involves choosing the lth highest weight at each edge. This strategy offers the opportunity to filter out false edges: by choosing the lth highest weight, at least l graphs must have the edge before it will register. The choice of l, however, represents a significant obstacle. This type of parameter ought to be chosen adaptively, ideally reflecting the number of trials, the probability of false negatives, and the probability of false positives. That would require complicated probabilistic modeling, which is not the objective here. This method also does not take context into account: ideally, an algorithm would be smart enough to filter out false edges based on surrounding network structure, rather than relying exclusively on same-edge information. We also note that the "quantile" method fails to take note of nonzero elements' values, making decisions only based on presence or absence of an edge between a given pair of rows or columns.

By contrast, addition of adjacency matrices is straightforward and easy to interpret: all edges from every pair of rows are preserved in the result. This has the disadvantage that it fails to filter out false edges. One potential adaptation, thresholding, consists of adding together all of the adjacency matrices and then choosing the q highest edges to keep in each row. It makes use of nonzeroes' values, as opposed to the lth-highest-edge tactic, which discards all but ordering information. It takes notice of network structure when filtering out spurious edges, although there is more than one way to do this.

Choosing the q highest edges dictates the level of connectedness in the graph, avoiding the inherent variability of quantile selection but biasing the method towards clusters larger than q. Depending on the quantity and reliability of the starting data, this bias may or may not improve results. It would be interesting to see, given a generative model for the graphs to be condensed, exactly what type of bias this strategy entails, and whether q can be selected adaptively.

7.2 Projection Before Merging

In this strategy, we take each data matrix and project its columns onto its K left singular vectors with largest singular value. We then merge the graphs and perform spectral clustering using L_{rw} and k-means. Depending on how we merge the graphs after projecting, this approach is able to filter out false edges both in terms of their context within a single graph and their recurrence between different graph realizations. However, the method does not integrate information in the principled way that a Bayesian model would. There is also a concern that failure to share information between matrices while constructing the projectors would lead to loss of quality.

8 Experiments on Synthetic Data

Synthetic data were generated consisting of graphs on 30 nodes. Each graph had five clusters; cluster memberships for the 30 nodes were chosen to be 1 ... 5 for the first five and thereafter randomly such that P(node n belongs to class k) is 1/5. Once memberships were chosen, edges were created with probability p of having an edge within a cluster (Figure 5, vertical axis) and probability q (horizontal axis) of having edges between clusters. Ten edge set realizations were generated to form a 30 by 30 by 10 array for each graph.

For each array, multiple algorithms were run. The plot shows maps with total clustering error (brighter indicates more error) in 100 trials and at various (p, q) pairs. Our calculation of clustering error accounts for labeling ambiguity: if the ground truth is $(1, 1, 2, 3)$, then the clustering error of $(2, 2, 3, 1)$ is zero.

The two best-performing methods both add together all of the graph realizations as a first step, with one moving straight to spectral clustering and another first thresholding after the q highest entries in each row ($q = 10$ here).

9 Conclusions

9.1 Recap

In Part I, we defined a new model, MSC, where data are drawn from a union of tensor products of low-dimensional subspaces. Then, we proposed a method of inference involving subspace clustering within row and column spaces of the data matrices. We generated simulated data and showed that, under the simulated conditions, our algorithm out-performs the subspace clustering algorithm TSC, which is a logical baseline. In Part II, we examined different ways of merging multiple graph realizations such as those that emerge in MSC. Using another simulation experiment, we showed that under the simulated conditions, simply adding together the graph realizations was the best-performing strategy.

9.2 Peripheral Questions

Some of the questions prompted by this project lead towards interesting projects relate only tangentially to subspace clustering. For example, implicit in the subspace clustering literature is the belief that spectral

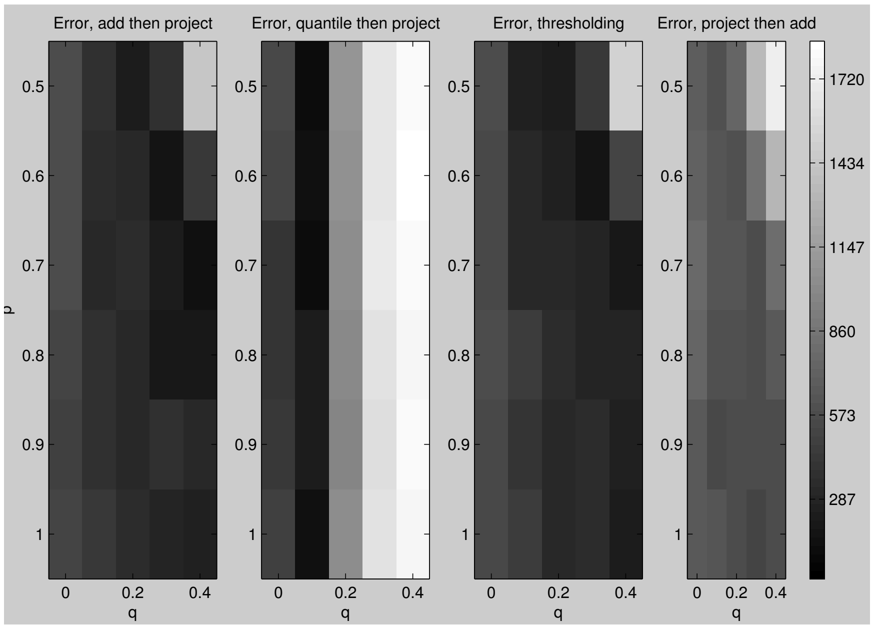

Figure 5: Performance of some candidate solutions to the graph condensing problem. The plot shows maps with total clustering error (brighter indicates more error) in 100 trials and at various false negative $(1 - p)$ and false positive (q) probabilities. Section 8 describes experiments in detail. See Section 7.2 for details on the algorithm labeled "Project then add" and Section 7.1 for details on the others.

clustering better tolerates missing edges than extra ones. To give examples, [29] makes guarantees only on avoidance of false positives, while [20] and [34] guarantee (with high probability) no false positives and some or many true discoveries. However, to our knowledge, to the sensitivity of spectral clustering methods to false negatives and false positives has not been explored mathematically. In reference to [40], this would require relating the hypotheses of Theorem 3.1, which involve the spectrum of a matrix parameter B, to the contents of the matrix B. In [39], Corollary 1 provides a result dealing directly in probabilities, but the procedure employed in that paper differs from the version of spectral clustering used in the subspace clustering literature (e.g. [29, 20, 34]).

There is also a gap between existing results on spectral clustering, which often use models with unweighted edges, and these algorithms, which apply spectral clustering to weighted graphs.

9.3 Additional Question and Future Work

Ultimately, we hope that MSC will prove its usefulness by exploiting structure that other algorithms ignore, but this depends on the existence of real data with the right type of structure. To convince the machine learning community these methods are valuable, we need to try this method on a number of data types and demonstrate that for some natural, non-synthetic data, it has an advantage in either speed or accuracy (preferably both). Unfortunately, its advantage in speed comes with fairly high dimensional data (at least 12×12), which seem like an unlikely fit for the model. A good place to start would be in the types of data mentioned in the introduction: EEG, auditory modulation features, or image patches.

The code written as part of this project works on data tensors of any order. It also has more of a speed advantage with higher order tensors. With this in mind, one potential " sweet spot" could be found in segmentation of color images, hyperspectral images, FMRI, or video. In order to speed up the algorithm so that it will work on data numbering in the ten thousands, which could happen when you split a 400×400 hyperspectral image into 4×4 superpixels, a parallel implementation of TSC would be useful. Likewise, economical methods for detecting the top few eigenvectors of a large Laplacian would be helpful. Another possibility is to take advantage of [41] and only compute some of the inner products.

As far as the theoretical end, experiments are a useful way to make conjectures, but a mathematical characterization is often both more thorough and more trustworthy. If MSC catches on, it could be valuable to investigate MSC and attempt to make theoretical guarantees on it. Is there a concrete structure behind the circumstances in which MSC outperforms TSC, and the reasons why?

References

[1] Griffith, O.L., Gao, B.J., Bilenky, M., Prichyna, Y., Ester, M., Jones, S.J.M.: KiWi: A Scalable Subspace Clustering Algorithm for Gene Expression Analysis. ArXiv e-prints (April 2009)

[2] Dutta, H., Waltz, D., Ramasamy, K., Gross, P., Salleb-Aouissi, A., Diab, H., Pooleery, M., Schevon, C., Emerson, R.: Patient-specific seizure detection from intra-cranial eeg using high dimensional clustering. In: Machine Learning and Applications (ICMLA), 2010 Ninth International Conference on. (Dec 2010) 782–787

[3] Yang, A.Y., Wright, J., Sastry, S.S., Ma, Y.: Unsupervised segmentation of natural images via lossy data compression. Technical Report UCB/EECS-2006-195, EECS Department, University of California, Berkeley (Dec 2006)

[4] Panagakis, Y., Kotropoulos, C.: Elastic net subspace clustering applied to pop/rock music structure analysis. Pattern Recognition Letters **38**(0) (3 2014) 46–53

[5] Hoff, P.D.: Separable covariance arrays via the tucker product, with applications to multivariate relational data. Bayesian Analysis **6**(2) (06 2011) 179–196

[6] Srivastava, M.S., von Rosen, T., von Rosen, D.: Estimation and testing in general multivariate linear models with kronecker product covariance structure. Sankhyā: The Indian Journal of Statistics, Series A (2008-) **71**(2) (08 2009) 137–163

[7] Werner, K., Jansson, M., Stoica, P.: Kronecker structured covariance matrix estimation. In: Acoustics, Speech and Signal Processing, 2007. ICASSP 2007. IEEE International Conference on. Volume 3. (April 2007) III–825–III–828

[8] Naik, D., Rao, S.: Analysis of multivariate repeated measures data with a kronecker product structured covariance matrix. Journal of Applied Statistics **28**(1) (2001) 91–105

[9] Huizenga, H., De Munck, J., Waldorp, L., Grasman, R.: Spatiotemporal eeg/meg source analysis based on a parametric noise covariance model. Biomedical Engineering, IEEE Transactions on **49**(6) (June 2002) 533–539

[10] Tipping, M.E., Bishop, C.M.: Probabilistic principal component analysis. Journal of the Royal Statistical Society, Series B **61** (1999) 611–622

[11] Tipping, M.E., Bishop, C.M.: Mixtures of probabilistic principal component analyzers. Neural Comput. **11**(2) (February 1999) 443–482

[12] Yang, J., Zhang, D., Frangi, A., Yang, J.Y.: Two-dimensional pca: a new approach to appearance-based face representation and recognition. Pattern Analysis and Machine Intelligence, IEEE Transactions on **26**(1) (2004) 131–137

[13] Ishteva, M., Absil, P.A., Van Huffel, S., De Lathauwer, L.: Best low multilinear rank approximation of higher-order tensors, based on the riemannian trust-region scheme. SIAM J. Matrix Anal. Appl. **32**(1) (February 2011) 115–135

[14] Eldén, L., Savas, B.: A newton–grassmann method for computing the best multilinear rank-$(r_1,$ $r_2,$ $r_3)$ approximation of a tensor. SIAM Journal on Matrix Analysis and Applications **31**(2) (2013/06/30 2009) 248–271

[15] Lathauwer, L.D., Vandewalle, J.: Dimensionality reduction in higher-order signal processing and rank-(r1,r2,...,rn) reduction in multilinear algebra. Linear Algebra and its Applications **391**(0) (2004) 31 – 55 ¡ce:title¿Special Issue on Linear Algebra in Signal and Image Processing¡/ce:title¿.

[16] Kolda, T., Bader, B.: Tensor decompositions and applications. SIAM Review **51**(3) (2013/06/30 2009) 455–500

[17] Bergqvist, G., Larsson, E.G.: Overview of recent advances in numerical tensor algebra. Signals, Systems and Computers (ASILOMAR), 2010 Conference Record of the Forty Fourth Asilomar Conference on (7-10 Nov. 2010) 3–7

[18] Hao, N., Kilmer, M., Braman, K., Hoover, R.: Facial recognition using tensor-tensor decompositions. SIAM Journal on Imaging Sciences **6**(1) (2013/06/20 2013) 437–463

[19] Vidal, R.: Subspace clustering. Signal Processing Magazine, IEEE **28**(2) (March 2011) 52–68

[20] Soltanolkotabi, M., Elhamifar, E., Candes, E.: Robust Subspace Clustering. ArXiv e-prints (January 2013)

[21] Heckel, R., Bölcskei, H.: Noisy Subspace Clustering via Thresholding. ArXiv e-prints (May 2013)

[22] Heckel, R., Bölcskei, H.: Robust Subspace Clustering via Thresholding. ArXiv e-prints (July 2013)

[23] MacQueen, J.: Some methods for classification and analysis of multivariate observations (1967)

[24] Sun, W., Wang, J., Fang, Y.: Regularized k-means clustering of high-dimensional data and its asymptotic consistency. (2012) 148–167

[25] Bussche, J., Vianu, V., Aggarwal, C., Hinneburg, A., Keim, D. In: On the Surprising Behavior of Distance Metrics in High Dimensional Space. Volume 1973. Springer Berlin Heidelberg (2001) 420–434

[26] Kaufman, L., Rousseeuw, P.J.: Finding groups in data: an introduction to cluster analysis. John Wiley and Sons, New York (1990)

[27] Fern, X.Z., Brodley, C.E.: Random projection for high dimensional data clustering: A cluster ensemble approach. (2003) 186–193

[28] A.Y.Ng, M.I.Jordan, a.: On spectral clustering: Analysis and an algorithm. Number 14 in Advances in Neural Information Processing Systems. MIT Press (2002)

[29] Elhamifar, E., Vidal, R.: Sparse subspace clustering: Algorithm, theory, and applications. Pattern Analysis and Machine Intelligence, IEEE Transactions on **PP**(99) (2013)

[30] Liu, G., Lin, Z., Yan, S., Sun, J., Yu, Y., Ma, Y.: Robust Recovery of Subspace Structures by Low-Rank Representation. ArXiv e-prints (October 2010)

[31] Luxburg, U.: A tutorial on spectral clustering. **17**(4) (2007) 395–416

[32] Nadler, B., Galun, M.: Fundamental limitations of spectral clustering. In: in Advanced in Neural Information Processing Systems 19, B. Schölkopf and. (2007) 1017–1024

[33] Ng, A.Y., Jordan, M.I., Weiss, Y.: On spectral clustering: Analysis and an algorithm. In: ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS, MIT Press (2001) 849–856

[34] Heckel, R., Bölcskei, H.: Robust Subspace Clustering via Thresholding. ArXiv e-prints (July 2013)

[35] Fink, A., Lausen, B., Seidel, W., Ultsch, A., Latouche, P., Birmelé, E., Ambroise, C. In: Studies in Classification, Data Analysis, and Knowledge Organization. Springer Berlin Heidelberg (2010) 229–239

[36] Shiga, M., Mamitsuka, H.: A variational bayesian framework for clustering with multiple graphs. Knowledge and Data Engineering, IEEE Transactions on **24**(4) (April 2012) 577–590

[37] von Luxburg, U., Belkin, M., Bousquet, O.: Consistency of spectral clustering. (2008) 555–586

[38] Balakrishnan, S., Xu, M., Krishnamurthy, A., Singh, A.: Noise thresholds for spectral clustering. In Shawe-Taylor, J., Zemel, R., Bartlett, P., Pereira, F., Weinberger, K., eds.: Advances in Neural Information Processing Systems 24. (2011) 954–962

[39] McSherry, F.: Spectral partitioning of random graphs. In: Foundations of Computer Science, 2001. Proceedings. 42nd IEEE Symposium on. (Oct 2001) 529–537

[40] Rohe, K., Chatterjee, S., Yu, B.: Spectral clustering and the high-dimensional stochastic blockmodel. ArXiv e-prints (July 2010)

[41] Wauthier, F.L., Jojic, N., Jordan, M.I.: Active spectral clustering via iterative uncertainty reduction. In: Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. KDD '12, New York, NY, USA, ACM (2012) 1339–1347