BAYESIAN CLUSTER VALIDATION

by

HOYT ADAM KOEPKE

B.A., The University of Colorado at Boulder, 2004

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF

Master of Science

in

THE FACULTY OF GRADUATE STUDIES
(Computer Science)

The University Of British Columbia
(Vancouver)

August 15, 2008

Table of Contents

List of Tables

List of Figures

Abstract

We propose a novel framework based on Bayesian principles for validating clusterings and present efficient algorithms for use with centroid or exemplar based clustering solutions. Our framework treats the data as fixed and introduces perturbations into the clustering procedure. In our algorithms, we scale the distances between points by a random variable whose distribution is tuned against a baseline null dataset. The random variable is integrated out, yielding a soft assignment matrix that gives the behavior under perturbation of the points relative to each of the clusters. From this soft assignment matrix, we are able to visualize inter-cluster behavior, rank clusters, and give a scalar index of the the clustering stability. In a large test on synthetic data, our method matches or outperforms other leading methods at predicting the correct number of clusters. We also present a theoretical analysis of our approach, which suggests that it is useful for high dimensional data.

Chapter 1

Cluster Analysis and Validation

Clustering, or cluster analysis, is the process of identifying groups of similar points in data with no associated class labels. It occurs commonly as a subproblem in statistics, machine learning, data mining, biostatistics, pattern recognition, image analysis, information retrieval and numerous other disciplines. It is usually used to discover previously unknown structure, but other uses, such as increasing efficiency in accessing data, are also common. Because the problem is ubiquitous, numerous clustering algorithms have been proposed and studied.

While many clustering algorithms produce a candidate partitioning, relatively few attach a measure of confidence to the proposed clustering. An ideal clustering algorithm would do both; however, such a procedure is not always practical. As a result, the field of cluster validation attempts to remedy this by proposing methods to assess how well a proposed clustering of a dataset reflects its intrinsic structure.

The purpose of this thesis is to introduce a novel technique for validating clusterings. We begin by broadly outlining, as much as possible, the field of cluster analysis and cluster validation. Then, in chapter 2 we introduce our method as a general framework for cluster validation that can apply generally to most clustering procedures and types of data. Then, in chapter 3,

we apply it specifically to distance based clusterings – clusterings where the final partitioning depends on the distance between the point and a centroid or exemplar point. To test our method, we then describe a procedure in chapter 4 for generating synthetic data with the correct number of clusters. Chapter 5 gives the results of a massive simulation on this data, compared against several other leading methods, showing that our method performs favorably.

Furthermore, our method has several desirable properties in high dimensions, which we outline in chapter 6. In addition, we also present results describing the breakdown of certain types of clusterings in high dimensions, which is itself of interest. Finally, in chapter 7, we introduce a Monte Carlo algorithm developed for the case of reclustering a distribution, but which may apply more broadly as well.

The focus of this chapter will be on clustering analysis and then validation methods. We will briefly describe several important clustering algorithms in section 1.1, but refer the reader to other resources for more in-depth treatments of the subject. In section 1.2.1, we discuss the concept of stability as a cluster validation tool. This motivates our discussion of cluster similarity indices in section 1.2.2, as stability based methods rely heavily on these.

1.1 The Clustering Function

There are numerous ways to classify and describe the plethora of proposed clustering algorithms, and our treatment of the subject here is far from complete. Because the algorithms we propose in our framework apply to a large class of clustering algorithms – those dealing with centroids or exemplar points – we summarize several clustering algorithms relevant to our method and refer the interested reader to one of [XW05, JTZ04, Ber02, JMF99, BB99] for more detailed surveys.

In this section, we first look at two aspects that many clustering algorithms, or clustering functions, share in common. The first common aspect is the minimization of a cost function, which we discuss in detail in section

$$\textit{Data} \rightarrow \mathscr{C}_S \rightarrow \textit{Cluster Statistics} \rightarrow \mathscr{C}_P \rightarrow \textit{Assignment}$$

Figure 1.1: The two stages of the clustering function.

1.1.3. This cost function returns a value that indicates how well partitioned the data is; a lower value of the cost function usually indicates a better clustering. The second aspect, and one directly relevant to our proposed framework, is that many clustering functions operate in two stages; the first stage creates some model describing the clusters and the second partitions the data. We visit this in section 1.1.1. Then, in section 1.1.2, we summarize several different classes of clustering algorithms and present details of some of the most common ones.

1.1.1 Stages of the Clustering Function

Anticipating the method we introduce in the next chapter, we note that many clustering algorithms can be described as a two stage process, as shown in Figure 1.1. The first stage is generating a set of statistics describing the data, and the second stage uses those statistics to partition the data.[1] Formally, suppose $\mathscr{C}(K, \mathcal{X})$ is a clustering function. In most cases, we can express this as a nested function, so

$$\mathscr{C}(K, \mathcal{X}) = \mathscr{C}_P(\mathscr{C}_S(K, \mathcal{X}), \mathcal{X}) \tag{1.1}$$

The first stage, $\mathscr{C}_S$, processes the data and outputs information or statistics about the clustering. The second stage, $\mathscr{C}_P$, uses this information to partition the data points into clusters. We represent this graphically in Figure 1.1.

For example, k-means generates a set of centroids and points are assigned to the closest centroid. In this case, either the centroids or the point-to-centroid distance matrix could be treated as summary statistics produced

[1] Technically, all clustering functions can be divided this way because the statistics could be a labeling of the data. While this makes the latter partitioning step trivial, we put no constraints on the form of the statistics used to generate the information to keep this description sufficiently general.

by $\mathscr{C}_S$. Additionally, many model-based clustering methods, such as EM with Gaussian mixture models, produce a mixture model representing the density of the data, and the mixture model parameters completely describe the clustering. Hierarchical algorithms often use such statistics of the data to perform the merging or dividing steps.

This idea is used explicitly in the information bottleneck approach originally proposed by Tishby [TPB00, Slo02, ST00a]. The main idea is to find a limited set of code words that preserves the most information about the data. While this approach applies more generally than to just clustering, it has inspired some effective clustering algorithms [TS00], particularly for document clustering [ST00b] and image clustering [GGG02].

1.1.2 Common Clustering Algorithms

There are several categorizations that are useful in distinguishing the numerous existing clustering algorithms. The distinguishing criteria we use here are hierarchical versus partitional. Hierarchical algorithms build a dendogram, or tree, representing a hierarchy of nested partitions, by either repeatedly merging smaller clusters or dividing larger clusters into parts. Agglomerative-type algorithms start with smaller clusters and successively merge them until some stopping criteria is met. Partitional algorithms, on the other hand, separate the data into the clusters in one pass.

Hierarchical Algorithms

The most basic hierarchical clustering algorithm is agglomerative; not surprisingly, it is called agglomerative clustering. It can apply to any type of data where the distances between points can be defined and is used not only for clustering points in Euclidean space but for clustering other types of data such as text [ST00a, ZK02].

It starts with each data point being a single cluster. At each stage, the closest pair of points or clusters is merged, ending with a single cluster and a tree shaped representation of the structure in the data. Closest is usually defined according to one of three linkage measures. Single linkage is the

smallest distance between any point in one cluster and any point in the other. This is fast, but tends to be highly sensitive to random effects in the data and can often produce clusters that look more like strands of spaghetti than clusters. Average linkage is the average distance between all pairs of points where one point is in the first cluster and the other is in the second cluster; this often produces better clusters but is a slower algorithm. Finally, complete linkage is the farthest distance between points in one cluster and points in the other; this is algorithmically as fast as single linkage, but often produces tighter, more compact clusters.

Another well-studied class of agglomerative clustering algorithms uses probabilistic models of the clustering and merges based on how well the resulting posterior distribution would reflect the data in the resulting cluster. The vanilla version [IT95] simply uses the likelihood of the resulting model; a more developed version proposed by Heller and Ghahramani [HG05] uses hypothesis testing. While these methods work for general models and data types, they have found the most use in text and document clustering as textual data is more easily represented by probabilistic models than by points in Euclidean space. For a more detailed discussion of these algorithms, and other algorithms for textual data, we refer the reader to [ZK02], [HMS02] or [SKK00].

Divisive hierarchical clustering algorithms recursively divide the data into two or more partitions, thus building the dendogram from the root node and ending with the leaves. There are several algorithms designed specifically for this purpose [Bol98]. Another simple method, though, is to apply a partitional clustering algorithm with only a handful of clusters to divide the data, then recursively apply it to each of the partitions.

Partitional Clustering Algorithms

By far the most popular partitional clustering algorithm is k-means [Ste06]. This is not surprising, as the algorithm has several nice properties. It is quite simple; it can be both described and coded in a few lines of code. The k-means algorithm is also computationally quite fast and is one of the few

clustering algorithms that scales well to millions or even billions of points with parallel implementations [SB99]. The algorithm works remarkably well on a variety of data (though more specialized algorithms can readily beat it). Theoretically, the algorithm attempts to minimize the squared error cost function, discussed in more detail below. This formulation is conducive to theoretical analysis, so the algorithm is quite well studied.

K-means works by iteratively assigning points to the nearest centroid, then repositioning those centroids to the mean of the points. Centroids are usually initialized randomly, though there are more sophisticated methods based on density estimate that yield better results and/or faster convergence times [AV07].

Another popular class of partitional clustering algorithms are spectral clustering algorithms, which also operate on a graph structure of points [KVV04, VM01, NJW02]. The central idea is that the values of the principle eigenvector of the inter-point distance matrix can be used to group the points – groups of similar points tend to have similar values in the principle eigenvector.

This idea has also been extended to directed graphs [MP07]. The min-flow-max-cut algorithm from optimization theory, can be used to cluster the points by partitioning the inter-point distance matrix so as to maximize the distances cut, thereby minimizing the distances within partitions [DHZ$^+$01, FTT04]. Additionally, statistics such as the sum of edge lengths within clusters also reflects the connectivity of the graph, and the quality of the resulting partitions, in a way that distance matrices do not.

1.1.3 The Squared Error Cost Functions

Many clustering algorithms, e.g. k-means, attempt to find a partition that minimizes a given cost function. K-means uses the squared error cost function, which we define formally below. Because we use this extensively later on, particularly in chapter 6, we formally show some of its general properties here. We reserve the full analysis, however, to chapter 6.

Definition 1.1.1. Squared Error Cost Function

Let $\mathscr{X}^{(p)} = (\mathbf{x}_1^{(p)}, \mathbf{x}_2^{(p)}, ..., \mathbf{x}_n^{(p)})$ be a list of n p-dimensional points. Then the squared error cost of $\mathscr{X}^{(p)}$ is

$$\text{cost}\Big(\mathscr{X}^{(p)}\Big) = \sum_{i=1}^{n} \|\mathbf{x}_i - \mathbf{m}\|_2^2 \tag{1.2}$$

$$= \sum_{i=1}^{n} \sum_{q=1}^{p} (x_{iq} - \mu_q)^2 \tag{1.3}$$

where

$$\mu_q = \frac{1}{n} \sum_{i=1}^{n} x_{iq} \tag{1.4}$$

is the mean of the qth component of all the points in $\mathscr{X}^{(p)}$. If $p = 1$, this simply becomes

$$\text{cost}\Big(\mathscr{X}^{(1)}\Big) = \sum_{i=1}^{n} (x_i - m)^2 \tag{1.5}$$

Definition 1.1.2. Squared Error Cost Function for Partitions

Let $\mathscr{X}^{(p)} = \mathbf{x}_1^{(p)}, \mathbf{x}_2^{(p)}, ..., \mathbf{x}_n^{(p)}$ be a set of n p-dimensional points, and let $\mathcal{P} = (P_1, P_2, ..., P_K)$ be a partitioning of those points into K clusters. The squared error cost function is then just the sum of the costs of each partition:

$$\text{cost}\Big(\mathscr{X}^{(p)}, \mathcal{P}\Big) = \sum_{k} \text{cost}\Big(\Big\{\mathbf{x}_i \in \mathscr{X}^{(p)} : i \in P_k\Big\}\Big) \tag{1.6}$$

$$= \sum_{k} \sum_{i \in P_k} \|\mathbf{x}_i - \boldsymbol{\mu}_k\|_2^2 \tag{1.7}$$

where $\boldsymbol{\mu}_k$ is the mean of the points in the kth partition.

Theorem 1.1.3. Separability of the Squared Error Cost Function into Components.

The squared error cost function is separable into dimension components. Specifically, suppose $\mathscr{X}^{(p)} = \mathbf{x}_1^{(p)}, \mathbf{x}_2^{(p)}, ..., \mathbf{x}_n^{(p)}$ *is a set of p-dimensional*

points, and let

$$\mathscr{X}_q^{(p)} = \left\{ x_q : \mathbf{x} \in \mathscr{X}^{(p)} \right\}. \tag{1.8}$$

Then

$$\text{cost}\left(\mathscr{X}^{(p)}\right) = \sum_{q=1}^{p} \text{cost}\left(\mathscr{X}_q^{(p)}\right) \tag{1.9}$$

Proof.

$$\text{cost}\left(\mathscr{X}^{(p)}\right) = \sum_{q=1}^{p} \sum_{i=1}^{n} (x_{iq} - \mu_q)^2 \tag{1.10}$$

$$= \sum_{q=1}^{p} \text{cost}\left(\mathscr{X}_q^{(p)}\right) \tag{1.11}$$

□

Theorem 1.1.4. Invariance of Cost Function Under Unitary Linear Transformations

Let $\mathbf{A}$ *be a* $p \times p$ *unitary matrix, and let* $\mathscr{X}^{(p)} = \mathbf{x}_1^{(p)}, \mathbf{x}_2^{(p)}, ..., \mathbf{x}_n^{(p)}$ *be a set of* n p*-dimensional points. Then*

$$\text{cost}\left(\left\{ \mathbf{Ax} : \mathbf{x} \in \mathscr{X}^{(p)} \right\}\right) = \text{cost}\left(\mathscr{X}^{(p)}\right) \tag{1.12}$$

Proof.

$$\begin{aligned}
\text{cost}\Big(\Big\{\mathbf{Ax} : \mathbf{x} \in \mathscr{X}^{(p)}\Big\}\Big) &= \sum_{i=1}^{n} \|\mathbf{Ax}_i - \mathbf{Am}\|_2^2 && (1.13)\\
&= \sum_{i=1}^{n} (\mathbf{A}(\mathbf{x}_i - \mathbf{m}))^T (\mathbf{A}(\mathbf{x}_i - \mathbf{m})) && (1.14)\\
&= \sum_{i=1}^{n} (\mathbf{x}_i - \mathbf{m})^T \mathbf{A}^T \mathbf{A} (\mathbf{x}_i - \mathbf{m}) && (1.15)\\
&= \sum_{i=1}^{n} (\mathbf{x}_i - \mathbf{m})^T (\mathbf{x}_i - \mathbf{m}) && (1.16)\\
&= \text{cost}\Big(\mathscr{X}^{(p)}\Big) && (1.17)
\end{aligned}$$

□

Another nice property of the squared error cost function is that it can be expressed as a linear function of the total distance between all the pairs of points. We present this in the following theorem.

Theorem 1.1.5. *The squared error cost function can be express in terms of the distances between every pair of points. Specifically,*

$$\text{cost}\Big(\mathscr{X}^{(p)}\Big) = \sum_i \|x_i - \bar{x}\|_2^2 = \frac{1}{2n} \sum_{i,j} \|x_i - x_j\|_2^2 \tag{1.18}$$

Proof. It is easier to start with the distances between pairs of points and show that it equals the original form. By definition, we have that

$$\frac{1}{2n}\sum_{i,j}(x_i - x_j)^2 = \frac{1}{2n}\sum_{i,j}((x_i - \bar{x}) - (x_j - \bar{x}))^2 \tag{1.19}$$

$$= \frac{1}{2n}\sum_{i,j}(x_i - \bar{x})^2 + (x_j - \bar{x})^2 - 2(x_i - \bar{x})(x_j - \bar{x}) \tag{1.20}$$

$$= \frac{1}{2n}\left[\sum_i 2n(x_i - \bar{x}) - \left(\sum_i 2n(x_i - \bar{x})\right)\left(\sum_j 2n(x_j - \bar{x})\right)\right]. \tag{1.21}$$

However,

$$\sum_i (x_i - \bar{x}) = \left[\sum_i x_i\right] - n\left[\frac{1}{n}\sum_i x_i\right] = 0 \tag{1.22}$$

so

$$\frac{1}{2n}\sum_{i,j}(x_i - x_j)^2 = \sum_i (x_i - \bar{x})^2. \tag{1.23}$$

Thus the theorem is proved. □

Corollary 1.1.6. *The cost function for multiple partitions can be expressed in terms of the difference between all pairs of points within each of the partitions. Specifically,*

$$\text{cost}\Big(\mathscr{X}^{(p)}, \mathcal{P}\Big) = \sum_k \sum_{i \in P_k} \|\mathbf{x}_i - \boldsymbol{\mu}_k\|_2^2 = \frac{1}{2}\sum_k \frac{1}{|P_k|}\sum_{i,j \in P_k} \|\mathbf{x}_i - \mathbf{x}_j\|_2^2 \tag{1.24}$$

Proof. This follows immediately from theorem 1.1.5 and the definition of the cost function for multiple partitions. □

Corollary 1.1.7. Invariance of the cost function to constant shifts

Let $\mathbf{u}$ *be any p-dimensional point. Let* $\mathscr{X}^{(p)} = \mathbf{x}_1^{(p)}, \mathbf{x}_2^{(p)}, ..., \mathbf{x}_n^{(p)}$ *be a set of n p-dimensional points, and let* $\mathscr{Y}^{(p)} = \mathbf{x}_1 + \mathbf{u}, \mathbf{x}_2 + \mathbf{u}, ..., \mathbf{x}_n + \mathbf{u}$. *Let* $\mathcal{P}$ *be a partitioning of those points into K partitions. Then*

$$\text{cost}\Big(\mathscr{X}^{(p)}, \mathcal{P}\Big) = \text{cost}\Big(\mathscr{Y}^{(p)}, \mathcal{P}\Big) \tag{1.25}$$

Proof. This follows directly from corrolary 1.1.6 by observing that the cost function is expressed entirely as the difference between points. Thus we have that $\text{cost}\big(\mathscr{Y}^{(p)}, \mathcal{P}\big)$ reduces trivially to $\text{cost}\big(\mathscr{X}^{(p)}, \mathcal{P}\big)$. □

1.2 Cluster Validation

Once a clustering is found, one should ask how well it reflects the natural structure of the data before trying to interpret it in the context of the experiment. It is critical to detect if the clustering is spurious – i.e. if the structure implied by the clustering is a random artifact of the data and not part of the true underlying distribution.

Ultimately, techniques for cluster validation need to aid in answering two overlapping questions regarding quantity and quality. First, does the number of proposed clusters accurately reflect the data [MB02]? Second, how representative of the modes of the underlying distribution is the clustering? For example, one might want to know how well the data supports two clusters being distinct or whether all the significant modes in the data are represented by a cluster.

Model based clustering methods fit a mixture model to the data, representing each "cluster" by one component of the mixture model. These methods produce, for each point, a distribution over cluster memberships formed by normalizing the probabilities that the point was drawn from each of the components. This so called soft assignment matrix provides a natural way to assess how well the mixture model reflects the data; the ideal fit would result in all points having a much higher probability of belonging to one component than to any others.

Other techniques involve assessing the clustering in terms of stability, or the resistance of the clusters to reasonable perturbations to the data.

The idea is that a clustering that reflects the true underlying distribution of the data will be resistant to such perturbations, but clusters that simply reflect spurious and random effects would not. The general process, then is to cluster the data multiple times, each time with different perturbations. If the resulting set of clusterings is consistent, then the original clustering is presumed to be stable. We discuss this more in the next section.

One aspect of cluster validation is ensuring that a proposed clustering correctly identifies the number of clusters. In simulations, this can be tested by checking how often a cluster validation technique shows that clustering the dataset into the correct number of clusters yields a better clustering than clustering it into an incorrect number of clusters. The problem of predicting the correct number of clusters is often phrased as a model selection problem [TW05], and is generally accepted in the literature as a method for comparing various cluster validation techniques. In chapter 5, we compare the accuracy of our approach against that of other clustering validation techniques on this problem.

1.2.1 Clustering Stability

Many stability based techniques for cluster validation have been proposed. The key idea of stability based cluster validation is to perturb the data set, usually by sub-sampling the data or adding noise, then clustering the perturbed data [BHEG02, Hen04, GT04]. The primary idea behind cluster stability is that a clustering solution should be resistant to changes, or perturbations, in the data that one would expect to occur in a real process.

The stability of the clustering is determined by analyzing the similarity of the clustering across data perturbation runs [BHEG02] or between the original data and the perturbed data, usually using a type of index designed for this purpose (see section 1.2.2). Typically, this is simply the average of the stability indices produced by the sequence of data perturbation runs, but other summaries are possible.

There are numerous papers on the best way to construct this procedure. Ben-Hur et. al. [BHEG02] randomly draws a certain percentage of all the

data points, clustering each, and compares each resulting clustering to all the others. They then form a histogram of the resulting stabilities and use that to chose the cluster. Other variants of this idea have also been proposed [ALA$^+$03, SG03].

As a slightly different variation, Breckenridge proposes measuring replication and consistency using a type of cross-validation on the original data [Bre89]. He first tries to identify consistently corresponding clusters between multiple runs, then measure the prediction strength on these consistently recurring clusters.

This idea is extended by Lange, Roth, Braun, and Buhmann in a series of papers [LBRB02, LRBB04, RLBB02]. Moller and Radke propose a variant based on using nearest neighbors to guide the resampling [MR06]. Their idea is to split the dataset into two different parts by splitting the nearest neighbor pairs, then ask how well a clustering on one part predicts a clustering on the other. Fridlyand and Dudoit also build on this idea, instead using a hierarchical procedure based on repeatedly splitting a dataset [FD01, DF02].

Despite all these techniques to perturb the data, it's unclear how often these stability indices are actually used in practice [Hen04]. Perhaps this is because a single scalar index is a fairly limited assessment of the properties of the clustering. Ben-Hur et. al. [BHEG02] uses histograms of the stability indices to present more information, but it is unclear if this is significantly more useful. An ideal summary would also include statistics indicating the stability of each of the clusters as well as the entire clustering, but few, if any, data perturbation techniques are able provide this information.

Critiques of Cluster Stability Analysis

Cluster stability analysis, while conceptually straightforward and commonly used in practice, does have some major flaws. In particular, Ben-David [BDvLP06] points out that a highly stable clustering implies that clustering is good. For example, k-means with 2 means will usually converge to a consistent solution when applied to three clusters in which two are suffi-

ciently well separated from the other or when applied to a single elliptical cluster. Though the latter case could be handled by a parsimony or model complexity argument, the former example would counter that.

Shamir and Tishby, however, counter this [ST07] by relating stability to measures of generalization in model selection, arguing that the rate of convergence of these measures is the important criteria. This, they argue, justifies the use of stability for cluster validation.

Another problem, however, has been pointed out by Tibshirani and Walther in [TW05]. In particular, if the true number of clusters is large, the method does not have sufficient resolution to distinguish between k and $k+1$ possible clusters. This problem, it seems, is systemic in these techniques; however, to counter it, may argue that knowing, for example, that predicting that a data set has 15 clusters when it actually has 16 is less grievous a mistake than predicting a clustering has 2 clusters when it actually has 3. Nevertheless, Tibshirani's observation lends more weight to our previous claim that summarizing an entire clustering by a single scalar stability index can easily hide important properties of the clustering.

Predicting the Number of Clusters

To predict the number of clusters in a dataset, one would would calculate a scalar assessment of the clusterings for a range of k, then choose the one that is "best" according to some specified criteria. In the data stability case, the scalar assessment is usually the stability of that clustering. If additional information is available, such as the standard deviation of the assessment, more sophisticated methods of choosing k can also be used, e.g. the smallest one within one standard deviation of the best.

In the literature, it seems that the canonical method for comparing techniques is in terms of the accuracy of how well they predict the number of clusters in a dataset where the true value of k is known. While one could argue that good performance on this type of problem may not be a good predictor of its performance in many real-life cluster validation scenarios, which is likely true, it is perhaps the best method of comparison out there

given the discussed limitations of existing techniques. Thus it is also the method we use in chapter 5 to compare our proposed cluster validation approach against current methods. Furthermore, choosing the correct number of clusters does come up often, and there are techniques developed specifically for that, e.g. the gap statistic, described in section 1.3.

1.2.2 Clustering Similarity Indices

Because cluster stability analysis depends on how measuring the consistency of a set of clusterings, a key subproblem that has also received widescale attention is how to accurately compare clusterings. Numerous methods have been proposed in the literature.

While there are literally dozens of indices, we focus on two here, the Hubert-Arabie Adjusted Rand Index [HA85], which we denote by $\mathcal{AR}$, and the Variation of Information, $\mathcal{VI}$ [Mei07, Mei03]. The $\mathcal{AR}$ index has a relatively long history in cluster analysis, while $\mathcal{VI}$ is more recent. [MC85] compare many different indices for measuring the difference between two partitions in a set of data and end up recommending the Hubert-Arabie adjusted Rand index. However, while it is arguably the most popular, we actually found the Variation of Information to perform better in our simulations.

We first discuss the matching matrix, something used commonly by similarity indices and these in particular. We then discuss the details of each index, and we include connections to the probabilistic formulations which can be readily incorporated into our index.

The Matching Matrix

One of the common representations of the overlap between two clusterings is a so-called matching matrix or confusion table. If $\boldsymbol{\mathcal{C}}_A = \left\{C_1^A, C_2^A, ..., C_{K^A}^A\right\}$ and $\boldsymbol{\mathcal{C}}_B = \left\{C_1^B, C_2^B, ..., C_{K^B}^B\right\}$, let n_{jk} be $\left|C_j^A \cap C_k^B\right|$, that is, the number of points in common between clusters C_j^A and C_k^B. Then this matching matrix is just

$$\mathbf{M}_{AB} = [n_{jk}]_{j=1,2,\ldots,K^A,k=1,2,\ldots,K^B} \tag{1.26}$$

If the two clusterings are represented by assignment matrices $\mathbf{A}_A = [a^A_{jk}]$ and $\mathbf{A}_B = [a^B_{jk}]$, where $a_{jk} = 1$ if point j is a member of cluster j and 0 otherwise, than this matching matrix is just

$$\mathbf{M}_{AB} = \mathbf{A}_A^T \mathbf{A}_B \tag{1.27}$$

Let $n^A_j = \left|C^A_j\right|$ and let $n^B_k = \left|C^B_k\right|$. Note that these statistics match up to the total counts in rows and columns of the matching matrix:

$$n^A_j = \sum_k n_{jk} \tag{1.28}$$

$$n^B_k = \sum_j n_{jk} \tag{1.29}$$

The majority of the cluster stability indices rely on this matching matrix and particularly these summaries. We go on to present two such indices in the next two sections.

Hubert-Arabie Adjusted Rand Index

The Hubert-Arabie Adjusted Rand Index is based on the Rand index proposed in 1971 [HA85] and measures stability by looking at pairwise assignments between clusterings. Given two clusterings $\boldsymbol{\mathcal{C}}_A$ and $\boldsymbol{\mathcal{C}}_B$ on a dataset $\boldsymbol{\mathcal{X}}$ with n points, we can separate the $n(n-1)$ possible pairs of points into four categories and count the number of pairs in each:

N_A	Number of pairs whose points in the same clusters (ignoring cluster labels) in $\boldsymbol{\mathcal{C}}_A$ and $\boldsymbol{\mathcal{C}}_B$.
N_B	Number of pairs whose points are in the same cluster in $\boldsymbol{\mathcal{C}}_A$ but different clusters in $\boldsymbol{\mathcal{C}}_B$.
N_C	Number of pairs whose points are in different clusters in $\boldsymbol{\mathcal{C}}_A$ but the same cluster in $\boldsymbol{\mathcal{C}}_B$.
N_D	Number of pairs whose points are in different clusters in both $\boldsymbol{\mathcal{C}}_A$ and $\boldsymbol{\mathcal{C}}_B$.

Using these definitions, the Rand index is defined as

$$\mathcal{R}(\boldsymbol{\mathcal{C}}_A,\boldsymbol{\mathcal{C}}_B) = \frac{N_A + N_D}{N_A + N_B + N_C + N_D} \tag{1.30}$$

it is easy to see that $\mathcal{R}(\boldsymbol{\mathcal{C}}_A,\boldsymbol{\mathcal{C}}_B)$ is 1 if both clusterings are identical.

Computing N_A, N_B, N_C, and N_D can be done naively in $\mathcal{O}\big(n^2\big)$ time. However, by using alternate statistics of the clusters, we can calculate it much more efficiently. Using the notation used to define the matching matrix in the previous section, we have that

$$N_A = \textstyle\sum_{j,k}\binom{n_{jk}}{2} \tag{1.31}$$

$$N_B = \textstyle\sum_{j}\binom{n_j^A}{2} - N_A \tag{1.32}$$

$$N_C = \textstyle\sum_{k}\binom{n_k^B}{2} - N_A \tag{1.33}$$

$$N_D = \binom{n}{2} - N_A + N_B + N_C \tag{1.34}$$

which can be calculated easily.

However, the main issue with the Rand index is that its expected value depends on n and tends toward 1 as n increases [Ste04, HA85]. This makes results on different datasets more difficult to compare and limits the power of the method for large n. Thus in 1986, Hubert and Arabie proposed a modified index that solved this problem:

$$\mathcal{AR}(\boldsymbol{\mathcal{C}}_A,\boldsymbol{\mathcal{C}}_B) = \frac{\mathcal{R}(\boldsymbol{\mathcal{C}}_A,\boldsymbol{\mathcal{C}}_B) - \mathbb{E}\left[\mathcal{R}(\boldsymbol{\mathcal{C}}_A,\boldsymbol{\mathcal{C}}_B)\right]}{\max \mathcal{R}(\boldsymbol{\mathcal{C}}_A,\boldsymbol{\mathcal{C}}_B) - \mathbb{E}\left[\mathcal{R}(\boldsymbol{\mathcal{C}}_A,\boldsymbol{\mathcal{C}}_B)\right]} \tag{1.35}$$

To calculate the expectations, they assumed the partitionings came from a generalized hypergeometric distribution. Under this assumption, it can be proved [HA85] that

$$\mathbb{E}\left[\mathcal{R}\boldsymbol{\mathcal{C}}_A,\boldsymbol{\mathcal{C}}_B\right] = \left[\sum_j \binom{n_j^A}{2}\right]\left[\sum_k \binom{n_k^B}{2}\right]\binom{n}{2}^{-1}. \tag{1.36}$$

The proof, however, is lengthy and tedious and we omit it here. With some simple algebra, $\mathcal{AR}$ can then be expressed as

$$\mathcal{AR}(\boldsymbol{\mathcal{C}}_A,\boldsymbol{\mathcal{C}}_B) = \frac{\sum_{j,k}\binom{n_{jk}}{2} - \left[\sum_j \binom{n_j^A}{2}\right]\left[\sum_k \binom{n_k^B}{2}\right]\binom{n}{2}^{-1}}{\frac{1}{2}\left[\sum_j \binom{n_j^A}{2} + \sum_k \binom{n_k^B}{2}\right] - \left[\sum_j \binom{n_j^A}{2}\right]\left[\sum_k \binom{n_k^B}{2}\right]\binom{n}{2}^{-1}} \tag{1.37}$$

This expression can be calculated efficiently given the partitionings $\boldsymbol{\mathcal{C}}_A$ and $\boldsymbol{\mathcal{C}}_B$.

Because the Hubert-Arabie Adjusted Rand Index depends on partition counts, we cannot apply the index directly when our input is in terms of probabilities or distributions over point assignments. In this case, we can use an n-invariant form of the index as given by [YR01] and [Mei07]. The idea is to represent each row of the soft assignment matrix by m draws from a multinomial distribution with weights given by that row. The similarity index between the two soft partitionings of n_1, n_2 points is then approximated by the similarity between mn_1, mn_2 points with a hard partitioning, an the approximation becomes exact as $m \to \infty$.

Let p_{jk} be the probability of a point belonging to cluster j in $\boldsymbol{\mathcal{C}}_A$ and cluster k in $\boldsymbol{\mathcal{C}}_B$, and let p_j^A be the probability of a point belonging to cluster j in $\boldsymbol{\mathcal{C}}_A$ and similarly for p_k^B. Then the n-invariant form is given by [Mei07]

$$\mathcal{PAR} = \frac{\sum_j \sum_k p_{jk}^2 - \left(\sum_j \left(p_j^A\right)^2\right)\left(\sum_k \left(p_k^B\right)^2\right)}{\frac{1}{2}\left[\left(\sum_j \left(p_j^A\right)^2\right) + \left(\sum_k \left(p_k^B\right)^2\right)\right] - \left(\sum_j \left(p_j^A\right)^2\right)\left(\sum_k \left(p_k^B\right)^2\right)} \tag{1.38}$$

Again, the derivations is tedious and we omit it here.

Variation of Information

Some scalar similarity indices rely on probabilistic summaries of the clustering, e.g. the empirical probability of a point being in C_j^A in clustering $\boldsymbol{\mathcal{C}}_A$ and C_k^B in clustering $\boldsymbol{\mathcal{C}}_B$. The Variation of Information is one such index.

The Variation of Information measures the sum of information lost and information gained between the two clusterings:

$$\mathcal{VI}(\boldsymbol{\mathcal{C}}_A, \boldsymbol{\mathcal{C}}_B) = H(A) + H(B) - 2MI(A, B) \tag{1.39}$$

where $H(A)$ is the entropy of A, and $MI(A, B)$ is the mutual information between A and B.

When using a hard clustering, we can get the probabilities needed empirically using statistics from the matching matrix. Now $p_j^A = n_j^A/n^A$ denotes the probability that a point is assigned to cluster j in $\boldsymbol{\mathcal{C}}_A$, and likewise $p_k^B = n_k^B/n^B$ denotes the probability that a point is assigned to cluster k in $\boldsymbol{\mathcal{C}}_B$. Likewise, $p_{jk} = n_{jk}/n$ is the probability that a point is assigned to cluster C_j^A in clustering $\boldsymbol{\mathcal{C}}_A$ and cluster C_k^B in $\boldsymbol{\mathcal{C}}_B$. Then Meilă's Variation of Information is:

$$\mathcal{VI}(\boldsymbol{\mathcal{C}}_A, \boldsymbol{\mathcal{C}}_B) = -\sum_j p_j^A \log p_j^A - \sum_k p_k^B \log p_k^B - 2\sum_j \sum_k p_{jk} \log \frac{p_{jk}}{p_j^A p_k^B} \tag{1.40}$$

This can easily be calculated.

1.3 Gap Statistic

One popular way of choosing number of clusters is using the so-called gap statistic proposed by Tibshirani [TWH01]. The gap statistic uses the difference in total cluster spread, defined in terms of the sum of all pairwise distances in a cluster, between the true dataset and the clusterings of several reference distributions with no true clusters. For example, in Euclidean space and with the squared error distance metric, the spread is the total empirical variance of all the clusters.

The reference datasets are used primarily to adjust for the dependence on k in the measure, but also to guard against spurious clusterings as random structure would presumably be present in the reference datasets. However, generating a null dataset is not necessarily easy, as the final value can still be highly dependent on the distribution of points. While much of this is not well understood, Tibshirani [TWH01] proposes using the uniform distribution within the PCA rotated bounding box of the original dataset, arguing that a uniform distribution is most likely to contain spurious structure.

Formally, the within-cluster spread of points is the sum of the distance between each point within a cluster to every other point in the same cluster divided by the number of points:

$$\text{spread}\left(\mathbf{C}\right) = \frac{1}{2n_j} \sum_{i,i' \in C_j} \text{dist}\left(\mathbf{x}_i, \mathbf{x}_i'\right) \tag{1.41}$$

The most common choice for the distance measure is the Euclidean sum of squares distance:

$$\text{dist}\left(\mathbf{x}_i, \mathbf{x}_i'\right) = \|\mathbf{x}_i - \mathbf{x}_i'\|_2^2, \tag{1.42}$$

but other distance measures can also be used. Note that in this case, theorem 1.1.5 tells us that the spread is the same as the empirical variance of the cluster around the mean of the points.

We present the procedure more formally as a three step process:

1. For each k in a given range of k's, cluster the data into k clusters

$\mathbf{C} = \{C_1, C_2, ..., C_k\}$.

2. Generate N_b reference datasets and cluster each to form N_b sets of clusters $\mathbf{C}_1, \mathbf{C}_2, ..., \mathbf{C}_{N_b}$. The gap statistic is then

$$\text{Gap}\,(k) = \left[\frac{1}{N_b}\sum_{b=1}^{N_b}\log\left(\text{spread}\,(\mathbf{C}_b)\right)\right] - \log\left(\text{spread}\,(\mathbf{C})\right) \tag{1.43}$$

The purpose of the logs is to essentially use the fractional difference between the given data and the reference instead of the absolute difference, as $\log(a/b) = \log a - \log b$. Because there is more dependence on k in $\text{spread}\,(\mathbf{C}) - \text{spread}\,(\mathbf{C}_b)$ than in $\text{spread}\,(\mathbf{C})/\,\text{spread}\,(\mathbf{C}_b)$, the logs are used here.

3. If $\bar{b} = \frac{1}{N_b}\sum_{b=1}^{N_b}\log\left(\text{spread}\,(\mathbf{C}_b)\right)$, compute the unbiased standard deviation of the baseline clusterings:

$$s_k = \sqrt{\frac{1}{N_b - 1}\sum_{b=1}^{N_b}\left(\log\left(\text{spread}\,(\mathbf{C}_b)\right) - \bar{b}\right)^2}, \tag{1.44}$$

then estimate the number of clusters $\hat{K}$ using one of two criteria:

A. $\hat{K}$ = smallest k such that $\text{Gap}\,(k) \geq \text{Gap}\,(k+1) - s_{k+1}$.

B. $\hat{K}$ = smallest k such that $\text{Gap}\,(k) \geq \text{Gap}\,(\ell) - s_\ell$, where $\ell = \text{argmax}_{\ell'}\,\text{Gap}\,(\ell')$. Note that $\hat{K}$ may equal ℓ.

The intuition behind A is that at some point $\text{Gap}\,(k)$ levels off and stops improving with increasing k. The intuition behind the second one invokes a parsimony principle – the simplest model which adequately explains the data, as determined by how well the best model explains it, is probably correct.

While [TWH01] proposes using A, we actually found B to work better on some of the simpler problems so we include it here. We study when each holds in detail in chapter 5.

In our results section, we compare our method against several flavors of data perturbation methods and the gap statistic. The gap statistic performs surprisingly well, though it begins to break down when the clusters do not have Gaussian shapes. In this case, it seems that our methods, described in the following chapters, perform the best.

Chapter 2

A Bayesian Framework for Clustering Stability

In this chapter, we propose an abstract framework that can, at a high level, be applied universally to almost any clustering function. In our approach, we modify the clustering function to take a hyperparameter that quantitatively introduces some type of perturbation, then integrate over a prior on the hyperparameter to obtain a probabilistic soft assignment matrix of points to clusters. This matrix gives the probability that a point i is assigned to a cluster j under perturbation.[1] To our knowledge, this approach is unique.

Given this matrix, we propose a set of summarizing statistics. These allow us to determine the behavior of specific points and clusters, rank clusters in terms of stability, quantify interactions between clusters, and compare clusterings with an overall stability index. Many of these tools – particularly those giving information about individual clusters within the clustering – are unique to our method.

We begin by formally describing our approach and introducing the perturbation induced soft assignment matrix, which we denote here and in subsequent chapters using $\mathbf{\Phi} = [\phi_{ij}]$. What we propose here is a framework that abstracts away many of the implementation details, namely the aspects

[1] This is in contrast to the soft assignment matrix that comes from model based clustering, which gives the probability that point i originated from jth mixture component.

of generating $\mathbf{\Phi}$ that depend on particular clustering algorithms. The rest of the chapter assumes that those issues are worked out (which we do in chapter 3). We then introduce a set of tools and summary statistics based on $\mathbf{\Phi}$ that aid in analyzing and validating a clustering. We end this chapter by discussing possible extensions that incorporate previously studied validation indices.

While this chapter focuses on the abstract framework, in the next chapter we use this framework to develop specific algorithms for clustering validation. The most critical aspects of this are choosing the type of perturbation and selecting a good prior, and we discuss these issues only superficially here. In chapter 5, we show, in a massive test on synthetic data, that the proposed validation algorithms proposed in this framework are as reliable or better than several other leading methods in terms of determining the number of clusters, demonstrating the potential usefulness of the proposed framework. Finally, in chapter 6, we discuss the asymptotic behavior of our method as the dimension increases, suggesting that it is appropriate for use in high dimensions.

2.1 The Abstract Framework

Suppose $\mathscr{C}(K, \mathcal{X})$ is a clustering function that partitions a set of n data $\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n\}$ into a set $\{C_1, C_2, ..., C_K\}$ of K clusters based on some structural feature of the data. Our approach is to create a new clustering function $\mathscr{C}^\star(K, \mathcal{X}, \boldsymbol{\lambda})$ by modifying $\mathscr{C}(K, \mathcal{X})$ to take a hyperparameter $\boldsymbol{\lambda}$, where the role of $\boldsymbol{\lambda}$ is (informally) to perturb some aspect of the clustering.[2] The definitions in our framework make no assumptions about how the parameter affects the clustering function; the details will vary based on the clustering algorithm and what type of stability one wishes to assess.

We then define a prior distribution $\pi(\boldsymbol{\lambda}|\theta)$ over the perturbation parameter indexed by a hyperparameter θ. While the full prior selection issue depends on the role of $\boldsymbol{\lambda}$ in $\mathscr{C}^\star$, we describe a procedure in chapter 3 to

[2]Notationally, we use a superscript $\star$ to denote a part of the procedure that has been modified to accommodate the perturbation parameter $\boldsymbol{\lambda}$

choose an optimal θ using various aspects of the clustering distribution.

Without loss of generality, suppose the clustering function $\mathscr{C}(K, \mathcal{X})$ returns an $n \times K$ assignment matrix giving the assignment of points to clusters. Thus

$$\mathbf{A} = [a_{ij}] = \mathscr{C}(K, \mathcal{X}) \tag{2.1}$$

In the hard clustering case, we assume that a_{ij} equals 1 if $\mathbf{x}_i$ is in cluster C and 0 otherwise.[3] For soft clustering algorithms, each row of $\mathbf{A}$ is a distribution over the clusters giving the partial membership of each point. Note that these definitions imply that $\sum_j a_{ij} = 1$.

Likewise, suppose that, for a given instance of the random variable $\boldsymbol{\lambda}$, the perturbed clustering function also $\mathscr{C}^\star(K, \mathcal{X}, \boldsymbol{\lambda})$ returns a similarly defined assignment matrix. Explicitly denoting the dependence on $\boldsymbol{\lambda}$, we have

$$\mathbf{A}^\star(\boldsymbol{\lambda}) = [a^\star_{ij}(\boldsymbol{\lambda})] = \mathscr{C}^\star(K, \mathcal{X}, \boldsymbol{\lambda}) \tag{2.2}$$

For our purposes, we require $\mathscr{C}^\star(K, \mathcal{X}, \boldsymbol{\lambda})$ to be a deterministic function of $\boldsymbol{\lambda}$. However, this does not necessarily exclude algorithms such as k-means which inherently incorporate some degree of randomness as we show later. Also, we assume $\mathcal{X}$ and K are constants throughout our procedure. Again, this does not exclude clustering algorithms that return a variable number of centroids, as we do allow empty clusters.

2.1.1 The Averaged Assignment Matrix

Recall that the modified clustering algorithm $\mathscr{C}^\star(K, \mathcal{X}, \boldsymbol{\lambda})$ is a deterministic function of $\boldsymbol{\lambda}$. This allows us to integrate out the parameter $\boldsymbol{\lambda}$ from $a^\star_{ij}(\boldsymbol{\lambda})$ with respect to $\pi(\mathrm{d}\boldsymbol{\lambda}|\theta)$. This gives us an $n \times K$ matrix $\boldsymbol{\Phi}$ such that ϕ_{ij} expresses the average membership of $\mathbf{x}_i$ in cluster C_j under perturbation.

More formally, suppose $\mathscr{C}^\star(K, \mathcal{X}, \boldsymbol{\lambda})$ returns, for each point of $\boldsymbol{\lambda}$, an $n \times K$ assignment matrix, and let $\pi(\boldsymbol{\lambda}|\theta)$ be a prior distribution over $\boldsymbol{\lambda}$. Then the averaged assignment matrix $\boldsymbol{\Phi}$ is given by

[3] Notationally, we consistently use i to index the data point and j, k, and ℓ to index clusters.

$$\mathbf{\Phi} = [\phi_{ij}] = \int \mathbf{A}^{\star}(\boldsymbol{\lambda})\pi(\boldsymbol{\lambda};\theta)\mathrm{d}\boldsymbol{\lambda}. \tag{2.3}$$

Note that if one interprets the assignment matrix $\mathbf{A}^{\star}(\boldsymbol{\lambda})$ as a probability distribution over the labels, i.e.

$$a^{\star}_{ij}(\boldsymbol{\lambda}) = p(a_{ij}|\boldsymbol{\lambda}), \tag{2.4}$$

then equation (2.3) has the form of a Bayesian posterior distribution.

This equation formalizes the core concept of our framework. The integration spreads the binary membership matrix $\mathbf{A}^{\star}(\boldsymbol{\lambda})$ across the clusters based on the behavior of those points under perturbation. Each row of $\mathbf{\Phi}$ can be interpreted as a probability vector such that ϕ_{ij} indicates the probability that datum $\mathbf{x}_i$ belongs to cluster j when perturbed. If the perturbation type and corresponding prior are well chosen, this probability matrix provides significant information about the behavior and stability of the clustering.

2.1.2 The Matching Matrix

We can define the averaged matching matrix $\mathbf{M} = [m_{jj'}]$ in terms of $\mathbf{\Phi}$:

$$\mathbf{M} = \mathbf{A}^T\mathbf{\Phi} \Leftrightarrow m_{jk} = \sum\nolimits_i a_{ij}\phi_{ik} = \sum\nolimits_{i:a_{ij}=1} \phi_{ik} \tag{2.5}$$

In this matrix, m_{jk} represents the total point-mass (each point having a mass of 1) in the unperturbed cluster C_j that moves to cluster C_k under perturbation. In an analogous way to the matching matrix for comparing two clusterings (see section 1.2.2), $m_{jk}/\left|C_j\right|$ (normalizing $\mathbf{M}$ across rows) is the probability of a point in the unperturbed cluster C_j belonging to cluster C_k under perturbation. Likewise, m_{jk}/n is the probability that a randomly-selected point belongs to cluster C_j in the unperturbed clustering and to cluster C_k under perturbation.

2.1.3 Perturbations and Label Matching

As we mentioned above, one assumption required in our framework is that $\mathscr{C}^*(K, \mathcal{X}, \boldsymbol{\lambda})$ is a deterministic function of the parameter $\boldsymbol{\lambda}$. However, this is not necessarily enough to guarantee that the integral in equation (2.3) gives a sensible answer. We discuss here one reason that is often referred to as the label matching problem.

The label matching problem refers to the fact that many algorithms may give similar answers across different runs but only modulo a permutation of the label indices. As a simple example, two runs of random-start k-means may give identical final locations for the centroids, but the centroid given the index 1 in the algorithm on the first one may have the index 5 on the second. This means that the points associated with one centroid in the first run may be associated with a differently labeled centroid in the second. Even without randomness in the clustering algorithm, many ways of introducing perturbations into the clustering function will cause the cluster labels to switch given different values of $\boldsymbol{\lambda}$.

The label matching problem is inherent in the data perturbation methods and has received some attention. There are generally two ways of dealing with it. The first is to rely on scalar stability indices such as those described in section 1.2.2; these are invariant to permutations of the labels and thus provide a way to compare clusterings when the label matching problem is an issue. The other way is to permute the labels to maximize their similarity [Bre00, LRBB04]. This corresponds to solving a bipartite graph matching problem between the two sets of clusters labels, where the edge weights are proportional to the number of shared points. Finding the optimal matching can be done in $O(K^3)$ time using the Hungarian method [Kuh55] or network flow simplex [Chv83].

Avoiding Label Matching Completely

To handle the label matching problem in our framework, we have a third option in addition to these two. Introducing the perturbation and hyper-parameter into the $\mathscr{C}_P$ step, so $\mathscr{C}^*(K, \mathcal{X}, \boldsymbol{\lambda}) = \mathscr{C}_P^*(\mathscr{C}_S(K, \mathcal{X}), \boldsymbol{\lambda})$, provides

an alternate way around this problem. Because the cluster statistics are already calculated, perturbations will not mix up the labels. This avoids the label matching problem completely, something not possible with data-perturbation methods.

Should we want to perturb the clustering function in a way that requires matching up the labels, we can introduce a $K \times K$ permutation matrix $\mathbf{P}(\boldsymbol{\lambda})$ to formally express this. This permutation matrix matches up the labels at each point of $\boldsymbol{\lambda}$. Equation (2.5) then becomes:

$$\boldsymbol{\Phi} = [\phi_{ij}] = \int_{\boldsymbol{\Lambda}} \mathbf{A}^{\star}(\boldsymbol{\lambda})\mathbf{P}(\boldsymbol{\lambda})\mathrm{d}\Pi(\boldsymbol{\lambda}) \tag{2.6}$$

where

$$\mathbf{P}(\boldsymbol{\lambda}) = \underset{\mathbf{Q}\in\boldsymbol{\mathcal{P}}}{\operatorname{argmax}}\ \operatorname{trace}\left[\mathbf{A}^{T}\mathbf{A}^{\star}(\boldsymbol{\lambda})\mathbf{Q}\right] \tag{2.7}$$

The definition for equation (2.5) which follows remains unchanged. While this is tractable when testing a small set of $\boldsymbol{\lambda}$, however, it can still pose a significant problem.

2.2 Visualizations and Statistical Summaries of the Averaged Assignment Matrix

In this section, we discuss ways of extracting useful information about the clustering from the averaged assignment matrix. The first technique we present is a picture of how the clusters give and take points under perturbation. We then expand this to include various statistical summaries of these properties.

2.2.1 Heatmap Plot of Φ

We present here a simple way to intuitively visualize the behavior of a clustering, by plotting a rearranged form of $\boldsymbol{\Phi}$ as a heat map. This heat map requires the cluster labels to be matched up; thus it is computationally

difficult using data perturbation methods and, while it is a simple procedure to display the clusterings, to our knowledge it has not been previously introduced.

Given the averaged assignment matrix and the unperturbed assignment, we construct the heat map that separates out the rows by their assignment in the unperturbed clustering and the probability they are assigned to that same cluster under perturbation. An example of how we do this is shown in Figure 2.1. As can be seen, it gives us a visually appealing and informative picture of the behavior of the clustering under perturbation.

To be precise, we build an index mapping that rearranges the rows of $\mathbf{A}$ so that the nonzero elements are all in contiguous blocks along the columns and these blocks are arranged in descending order along the rows. Within the blocks, we permute the rows so the elements in the block $\mathbf{\Phi}$ in descending order. We then apply this index mapping to $\mathbf{\Phi}$ and further refine it so that the rows within each block are sorted by the entry corresponding to the cluster to which the point is assigned in the baseline clustering. Algorithmically, this can all be expressed in two lines:

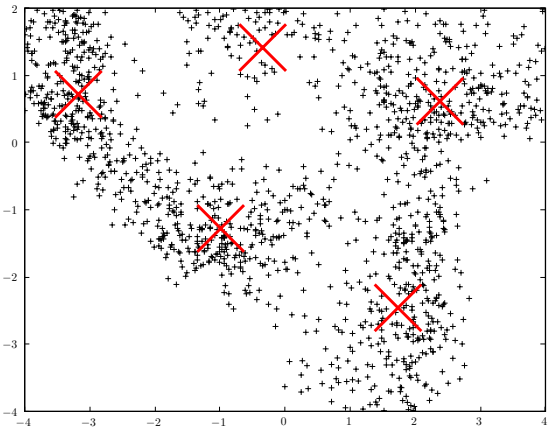
$$h_i = \underset{k}{\operatorname{argmax}}\, a_{ik} \tag{2.8}$$

$$\psi = \operatorname{argsort}\left(h_i + \phi_{ih_i}, i = 1, 2, ..., n\right) \tag{2.9}$$

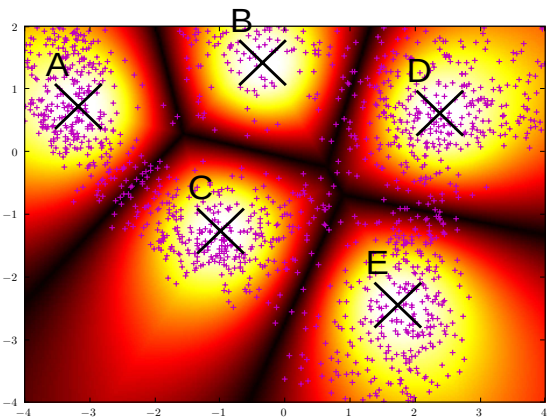
where ψ is the resulting mapping index.

To illustrate the properties of the averaged assignment matrix and the the corresponding heatmap, we show a 2d toy example in Figure 2.1. The specific perturbations (scaled distance) and prior (location exponential) used to generate the plots are introduced in chapter 3; we focus here on the intuitive understanding.

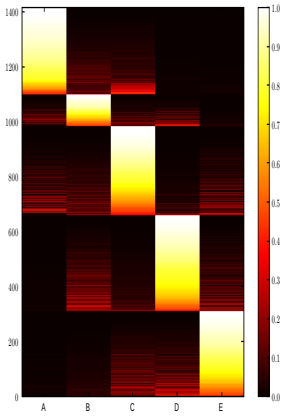
In the heatmap plot, the blocks along the "diagonal" denote the stability of points relative to their assigned clusters under perturbation, and the "off-diagonal" blocks represent membership in other clusters under perturbation. Each row shows how the corresponding point behaves when the cluster is



(a) Clustered Sample



(b) Pointwise Stability matrix



(c) Heatmap Plot

Figure 2.1: A 2d toy example illustrating the use of the heatmap plot for analyzing how clusters exchange point mass under perturbation. (a) shows the clustered 2d sample. (b) shows the sample with the pointwise stability (equation (2.11)) of each (x,y) coordinate plotted in the background, with white representing a pointwise stability of 1 and black a pointwise stability of 0. (c) shows the corresponding heatmap plot of the clustering.

perturbed; the point mass in unstable points will show up as red and orange colors to the side of such blocks. Dark red or black indicates separation between clusters. This allows us to visualize clearly how clusters exchange points when perturbed.[4]

As can be seen, cluster separation or closeness can be easily noted in the heatmap. For example, the 2d plot indicates that cluster E is well separated from cluster A; the heatmap plot shows this as A shares no point mass with E and vis-versa. Clusters A and B however, are not well separated, and exchange significant point mass as illustrated by the red and orange (gray) in blocks (A,B) and (B,A) in the heatmap.

2.2.2 Scalar Stability Indices

While the heatmap presents the clearest picture of the properties of the clustering, statistics summarizing the behavior of the clustering are often useful. We here present a set of summary statistics that condenses the relevant information in $\mathbf{\Phi}$. We here provide indices that serve several purposes. The pointwise stability of a point, $\mathcal{PW}_i$, is a per-point stability index that indicates how well point i fits into the clustering. The cluster-wise stability, $\mathcal{CW}_k$, summarizes the stability of each clustering. An inter-cluster index $\mathcal{IC}_{jk}$ indexes the separation of two clusters j and k. Finally, the average pointwise $\mathcal{APW}$ summarizes the stability of the entire clustering.

The primary motivating idea behind the way we chose to condense the information is competitive loss. Essentially, we measure the stability of a point according to the "closeness" of the competition. If the probability that a point belongs to one cluster under perturbation is nearly equal to the probability that it belongs to a different cluster, it lends evidence that the clustering is unstable. Likewise, if the probability that a point belongs to a given cluster is close to one, implying that it is significantly greater than the probability of it belonging to any other cluster, it lends evidence that the clustering is good and fits the data well.

[4]Note that all comparisons must be done keeping in mind that only the rows are normalized; comparisons within columns are not on the same scale and, though suggestive, are not formally justified.

Pointwise Stability

We define the pointwise stability of point i in terms of the difference in assignment probability between the cluster it is assigned to in the unperturbed clustering and the maximum probability of all the others. This, ultimately, gives us a scalar measure of confidence in the point's assignment. Formally,

$$h_i = \operatorname*{argmax}_k a_{ik} \tag{2.10}$$

$$\mathcal{PW}_i = \phi_{ih_i} - \max_{\ell \neq h_i} \phi_{i\ell} \tag{2.11}$$

For a dataset with n points, this produces a vector of length n. While other summary statistics might be possible – e.g. the entropy of $\phi_{i\cdot}$ (see below) – we find this definition works quite well in practice.

As can be seen in Figure 2.1, the pointwise stability imposes a type of partitioning on the space around the cluster centers. In this plot, we fix the location of the cluster centers and plot the pointwise stability of each (x,y) coordinate given the location of the cluster centers, i.e. what the pointwise stability of a point at (x,y) would be. Note that near the boundary regions, the pointwise stability is near zero, whereas near the cluster centers the pointwise stability is near one.

As mentioned, note that other summary statistics are also possible. In particular, one could use the entropy of each row of $\mathbf{\Phi}$ instead of the competitive loss as the resulting measure of stability. However, the pointwise stability has the advantage that it detects when the given clustering is wrong, i.e. the probability of a point belonging to the cluster it is assigned to in the unperturbed clustering is less than the probability of it belonging to a different cluster. To see this, suppose that $\phi_{i\ell} = 1$ for $\ell \neq j$ (recall that j is the cluster it is assigned to in the unperturbed clustering). In such a case, $\mathcal{PW}_i$ would be -1 but the entropy would be the same as the correct case where $\phi_{ij} = 1$.

Stability Measures for Clusters

We define the cluster-wise stability as the average pointwise stability of the points assigned to it in the unperturbed clustering. Specifically,

$$\mathcal{CW}_k = \underset{i:\mathbf{x}_i \in C_k}{\text{mean}} \mathcal{PW}_i \tag{2.12}$$

This statistic allows for useful operations such as ranking clusters by their relevance to the structure of the data. This type of information is very difficult to attain using other methods.

We can also measure the separation of clusters based on the probability that the points assigned to each of the clusters in the unperturbed clustering stay there under perturbation minus the probability that they move to the other cluster. Specifically,

$$\mathcal{IC}_{jk} = \frac{1}{|C_j| + |C_k|} \left(\sum_{i:\mathbf{x}_i \in C_j} \phi_{ij} - \phi_{ik} + \sum_{i:\mathbf{x}_i \in C_k} \phi_{ik} - \phi_{ij} \right). \tag{2.13}$$

A value of $\mathcal{IC}_{jk}$ near 1 indicates that clusters j and k are well separated as they exchange almost no point mass under perturbation. A value of 0 (or less) means that they are practically indistinguishable (assuming the method used to generate $\mathbf{\Phi}$ is sensible).

Note that it is also possible to construct an asymmetric measure of inter-cluster stability by measuring the average point mass that moves from cluster j to cluster k under perturbation. Formally,

$$\mathcal{A}\text{ssymmetric}\mathcal{IC}_{jk} = \underset{i:\mathbf{x}_i \in C_j}{\text{mean}} \phi_{ij} - \phi_{ik}. \tag{2.14}$$

This could be useful if the clusters had significantly different properties that, for instance, caused some clusters to be unstable and give point mass to their much more stable neighbors. While practically such a behavior can be seen in the heatmap plot, an index that allows for the detection of such behavior could be useful.

Stability Indices for the Clustering

Finally, the average of the pointwise stability vector gives us an overall measure of the stability of the clustering:

$$\mathcal{APW} = \underset{i}{\text{mean}}\, \mathcal{PW}_i. \tag{2.15}$$

We refer to this as the average pointwise stability of the clustering. In chapter 5, where we present our results, we use the difference between the $\mathcal{APW}$ of a clustering of the given data and the $\mathcal{APW}$ on a baseline null clustering, defined in the next chapter. We show that this method can be as accurate, in terms of predicting the number of clusters, as the other leading methods.

2.2.3 Extensions to Other Stability Indices

Because many of the scalar scalar indices work by comparing two partitionings, it is natural to compare the unperturbed assignment matrix and $\mathbf{\Phi}$ using these indices as an alternative method of obtaining a statistical summary of the clustering. In this section, we discuss using two such indices, the Hubert-Arabie Adjusted Rand Index and the Variation of Information. We introduced both of these in the context of comparing two clusterings in section 1.2.2.

Recall from section 1.2.2 that the n-invariant form of the Hubert-Arabie Adjusted Rand Index is given by

$$\mathcal{AR}^{\star} = \frac{\sum_j \sum_k p_{jk}^2 - \left(\sum_j \left(p_j^A\right)^2\right)\left(\sum_k \left(p_k^B\right)^2\right)}{\frac{1}{2}\left[\left(\sum_j \left(p_j^A\right)^2\right) + \left(\sum_k \left(p_k^B\right)^2\right)\right] - \left(\sum_j \left(p_j^A\right)^2\right)\left(\sum_k \left(p_k^B\right)^2\right)} \tag{2.16}$$

and the Variation of Information (section 1.2.2) is given by

$$\mathcal{VI}^\star = -\sum_j p_j^A \log p_j^A - \sum_k p_k^B \log p_k^B - 2\sum_j \sum_k p_{jk} \log \frac{p_{jk}}{p_j^A p_k^B} \tag{2.17}$$

We have the probabilities needed to calculate these from $\mathbf{A}$ and $\mathbf{\Phi}$ as follows:

p_j^A The probability of a point belonging to cluster j in the unperturbed clustering is simply the total number of points in that cluster divided by the total number of points, so

$$p_j^A = \frac{1}{n}\sum_i a_{ij}. \tag{2.18}$$

p_k^B Likewise, The probability of a point belonging to cluster k in the averaged assignment matrix is simply the total point mass in that cluster divided by the total number of points, so

$$p_k^B = \frac{1}{n}\sum_i \phi_{ik}. \tag{2.19}$$

p_{jk} The probability that a point belongs to cluster j in the unperturbed clustering and to cluster k under perturbation. Recall that the matching matrix $\mathbf{M}$ gives the total point mass shared between clusters the unperturbed clustering and clusters in the averaged assignment, so

$$p_{jk} = \frac{m_{jk}}{n} \tag{2.20}$$

We now have everything needed to calculate $\mathcal{VI}^\star$ and $\mathcal{AR}^\star$. However, we observed that these indices do not work as well as the average pointwise stability, so we present them here as possible extensions and instead recommend using the averaged pointwise stability for general use. This hints that in bypassing the label matching problem – as these indices do – they discard

important information about the dataset and the clustering.

2.3 Conclusion

In this chapter, we have presented a general framework for clustering validity based on introducing perturbations into the clustering function. This method may be seen as a natural expansion of clustering stability analysis to include more general perturbations of the clustering function. This expansion is consistent with the Bayesian tenet of conditioning on the data but expressing uncertainty in the modeling procedure.

Now that we have presented the framework at a high, abstract level, we go through the details of how we introduce the perturbation into the clustering function and how we select a prior.

Chapter 3

Bayesian Cluster Stability Algorithms

While the previous chapter proposed a general framework to use in developing cluster validation techniques, this chapter uses this framework to develop algorithms for cluster validation. Recall that the key idea in this framework is to alter the clustering function to take a parameter $\boldsymbol{\lambda}$ which indexes some sort of perturbation, then integrate out $\boldsymbol{\lambda}$ over a prior $\pi(\boldsymbol{\lambda}|\theta)$ to get an average assignment matrix $\boldsymbol{\Phi}$. For reference, recall here the principle equation as

$$\boldsymbol{\Phi} = [\phi_{ij}] = \int \mathbf{A}^\star(\boldsymbol{\lambda})\pi(\boldsymbol{\lambda};\theta)\mathrm{d}\boldsymbol{\lambda} \tag{3.1}$$

where $\mathbf{A}^\star(\boldsymbol{\lambda})$ is the perturbed assignment matrix returned by the modified clustering function,

$$\mathbf{A}^\star(\boldsymbol{\lambda}) = [a^\star_{ij}(\boldsymbol{\lambda})] = \mathscr{C}^\star(K, \mathcal{X}, \boldsymbol{\lambda}). \tag{3.2}$$

At this point, there are two outstanding issues to address.

First, what part of the clustering function should be perturbed? Certainly, there are numerous possibilities. Showing that a particular perturbation technique is "optimal" would be extremely difficult, and the proof

would likely be highly dependent on a particular application, clustering algorithm, or type of data. However, it is far less difficult to demonstrate that a particular technique can work well on several types of data and a variety of clustering algorithms. In this chapter and later, we advocate scaling the distance metric used to partition the data points by the random variable indexing the perturbation. We demonstrate in section 3.1 that this method has some desirable geometric qualities. In chapter 5, we show that it can perform quite well, matching or outperforming other leading methods. On the theoretical side, we show in chapter 6 we show that it has some nice properties as the dimension increases.

Second, given the perturbation method, which prior should be used? Again, there are numerous possibilities to consider, and proving that a prior is somehow optimal is quite difficult and would likely depend significantly on the type of data and other factors; thus it is beyond our scope. Our aim here is to find a prior that works well on a wide variety of data and can be calculated quickly enough to be used in a wide variety of applications.

Ultimately, we leave the full prior selection question for future research, and instead focus on choosing the best prior out of a class of priors. We choose the hyperparameters of the prior to maximize the difference in overall stability against a baseline null clustering and the overall stability of the clustered data. For a class of perturbations based on distance scaling, this technique turns out to work quite well, as we demonstrate in chapter 5. We propose several classes of priors for the distance scaling perturbations, a location exponential and shifted Gamma distribution. In section 3.3.1, we derive an efficient algorithm to calculate the averaged assignment matrix $\mathbf{\Phi}$ with a location exponential prior in $\mathcal{O}(nK \log K)$ time. In section 3.4.1, we do the same for the shifted Gamma prior, though this algorithm requires $\mathcal{O}\big(nK^3\big)$ time to calculate. However, we show in section 3.6 that these prior classes have several nice properties, the most significant being provable bounds on the error if some of the centroids are excluded from the calculation; this can vastly improve the running time. Finally, in section 3.5, we propose a general and reasonably fast Monte Carlo algorithm, which allows the partial assignment matrix to be calculated provided the prior can

be sampled from.

3.1 Perturbing the distance metric

In the algorithms we discuss here, we restrict ourselves to clustering functions where the final partitioning depends on the distances to a set of centroids or exemplar points, and data points are assigned to the nearest centroid or exemplar. Thus we assume the first stage of the clustering algorithm returns an $n \times K$ matrix $\mathbf{D} = [d_{ij}]$, where d_{ij} is the distance between point i and centroid (or exemplar) j. Then the assignment matrix $\mathbf{A}$ in the unperturbed clustering would be

$$a_{ij} = \begin{cases} 1 & d_{ij} \leq d_{i\ell} \ \forall \, \ell \neq j \\ 0 & \text{otherwise} \end{cases} . \tag{3.3}$$

We propose that scaling the distance metrics used to partition the points is a reasonable way to introduce the perturbation. Specifically, given $\mathbf{D} = [d_{ij}]$, the perturbed assignment matrix $\mathbf{A}^\star(\boldsymbol{\lambda})$ would be

$$a^\star_{ij}(\boldsymbol{\lambda}) = \begin{cases} 1 & d_{ij}\lambda_j \leq d_{i\ell}\lambda_\ell \ \forall \, \ell \neq j \\ 0 & \text{otherwise} \end{cases} , \tag{3.4}$$

and the elements of $\boldsymbol{\Phi} = [\phi_{ij}]$ can be expressed as

$$\phi_{ij} = \int \mathbb{I}[d_{ij}\lambda_j \leq d_{i\ell}\lambda_\ell \ \forall \, \ell \neq j] \pi(\boldsymbol{\lambda}; \theta) \mathrm{d}\boldsymbol{\lambda}. \tag{3.5}$$

For convenience, we here introduce an alternative functional formulation of equation (3.5) that explicitly denotes the dependence on the distances. Let $\mathbf{d}_i = (d_{i1}, d_{i2}, ..., d_{iK})$ be the vector of points to centroids. Then, define the function ψ as

$$\psi_j(\mathbf{d}_i, \theta) = \phi_{ij} = \int \mathbb{I}[d_{ij}\lambda_j \leq d_{i\ell}\lambda_\ell \ \forall \, \ell \neq j] \pi(\boldsymbol{\lambda}; \theta) \mathrm{d}\boldsymbol{\lambda}. \tag{3.6}$$

We use this formulation often in subsequent proofs.

For several reasons, we believe this is a reasonable way to introduce the perturbations. First, because the distance measures are the statistics used to

partition the data points, changing them sufficiently will result in an altered configuration. Second, it gives us an excellent geometrical interpretation which we elaborate on next. Third, we are able to derive computationally efficient and accurate algorithms to calculate ϕ_{ij} – for a location exponential prior, the most useful prior we've found, it can be done in amortized $\mathcal{O}(\log K)$ time. Finally, we demonstrate in chapter 5 that it can yield excellent results that match or exceed all other methods we've compared against.

3.1.1 Intuitive Understanding

Restating equation (3.5) in terms of probabilities gives us

$$\phi_{ij} = \mathbb{P}(d_{ij}\lambda_j \leq d_{i\ell}\lambda_\ell \ \forall \ell) \tag{3.7}$$

ϕ_{ij} is, intuitively, a measure of how "competitive" the distances to the clusters are. If cluster j is a clear winner – meaning it is significantly closer than the others – then $\mathbb{P}(d_{ij}\lambda_j \leq d_{i\ell}\lambda_\ell \ \forall \ell)$ would be higher as the condition would be true for a larger set of $\boldsymbol{\lambda}$. Conversely, if the distances to another cluster is similar to the distance to cluster j, ϕ_{ij} would be less as the event $\{d_{ij}\lambda_j \leq d_{i\ell}\lambda_\ell \ \forall \ell\}$ would occur for a smaller set of $\boldsymbol{\lambda}$. Essentially, then the values in $\boldsymbol{\Phi}$ provide information about the relative density of points close to the boundary regions versus the density around the cluster centers.

3.1.2 Example: Exponential Prior

At this point, it is helpful to consider an example prior for scaled distance based perturbations that has an intuitive analytic evaluation for ϕ. The priors that we ultimately propose for use in actual cluster validation work yield a very unintuitive equation for ϕ_{ij} (see section 3.3.1, but work much better than this one in practice. We first present the result as a proof and then discuss the final equation.

Theorem 3.1.1. ϕ_{ij} **with an exponential prior.**

Let ϕ_{ij} be defined as in equation (3.5), and let

$$\pi(\boldsymbol{\lambda}) = \prod_{\ell} \mathcal{Exp}(\lambda_\ell|\theta) = \prod_{\ell} \theta e^{-\theta\lambda_\ell}. \tag{3.8}$$

Then

$$\phi_{ij} = \frac{d_{ij}^{-1}}{\sum_\ell d_{i\ell}^{-1}} \tag{3.9}$$

Proof. For convenience, we drop the i's as they are constant throughout the proof. By definition, we have that

$$\begin{aligned}
\phi_j &= \int \mathbb{I}[d_j\lambda_j \leq d_\ell\lambda_\ell \ \forall\, \ell \neq j] \prod_{\ell} \mathcal{Exp}(\lambda_\ell|\theta)\mathrm{d}\boldsymbol{\lambda} && (3.10)\\
&= \int \theta e^{-\theta\lambda_\ell} \prod_{\ell\neq j} \left[\mathbb{I}[d_j\lambda_j \leq d_\ell\lambda_\ell]\theta e^{-\theta\lambda_\ell}\right] \mathrm{d}\boldsymbol{\lambda} && (3.11)\\
&= \int_0^\infty \theta e^{-\theta\lambda_\ell} \left[\prod_{\ell\neq j} \int_{d_j\lambda_j/d_\ell}^\infty \theta e^{-\theta\lambda_\ell}\mathrm{d}\lambda_\ell\right] \mathrm{d}\lambda_j && (3.12)\\
&= \int_0^\infty \theta e^{-\theta\lambda_\ell} \left[\prod_{\ell\neq j} \exp\left[-\frac{\theta d_j}{d_\ell}\lambda_j\right]\right] \mathrm{d}\lambda_j && (3.13)\\
&= \int_0^\infty \theta \exp\left[-\left(1+\sum_{\ell\neq j}\frac{d_j}{d_\ell}\right)\theta\lambda_j\right] \mathrm{d}\lambda_j && (3.14)\\
&= \frac{\theta}{\theta\left(1+\sum_{\ell\neq j}\frac{d_j}{d_\ell}\right)} \int_0^\infty \mathcal{Exp}\left(\lambda_j|\theta\left(1+\sum_{\ell\neq j}\frac{d_j}{d_\ell}\right)\right)\mathrm{d}\lambda_j && (3.15)\\
&= \frac{1}{1+\sum_{\ell\neq j}\frac{d_j}{d_\ell}} && (3.16)\\
&= \frac{d_j^{-1}}{\sum_\ell d_\ell^{-1}} && (3.17)
\end{aligned}$$

□

This result – that the stability is indicated by the normalized inverse distance metrics – is a sensible. If one of the distances is quite small relative

to the others, the corresponding entry in ϕ_{ij} will be close to 1. Similarly, if all of them are close to the same value, ϕ_{ij} becomes close to $1/K$.

3.2 Prior Selection Using a Baseline Null Distribution

The full question of prior selection is historically an extremely difficult problem, and the present case is no different. As a result, finding a good prior, or class of priors, is an effort/reward trade-off – the goal is to find some guiding principles that are relatively easy to work with but greatly improve the technique. While there are surely additional undiscovered principles beyond the one we outline here, the following principle has proved very useful.

3.2.1 Observations from High Dimensions

In higher dimensions, the distances between points become similar and many methods break down. Much literature has been published on this effect. In fact, if each dimension component $\mathbf{x}_i$ in a set of random data is i.i.d. from a continuous distribution,

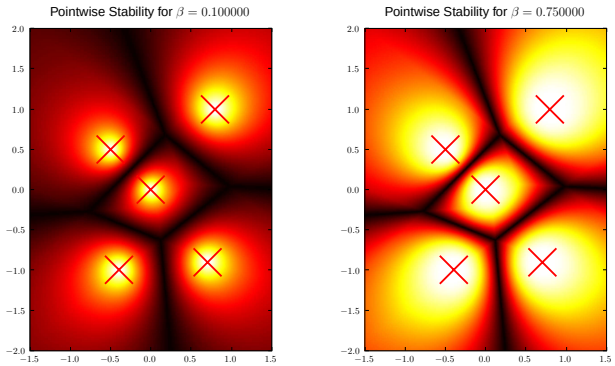
$$\frac{\max_{i \neq j} \|\mathbf{x}_i - \mathbf{x}_j\|_2}{\min_{i \neq j} \|\mathbf{x}_i - \mathbf{x}_j\|_2} \xrightarrow{\mathbb{P}} 1 \tag{3.18}$$

as the dimension goes to infinity. This is proved by Hinneburg et. al. [HAK00] (their version of the proof has more general conditions than i.i.d., but i.i.d. and continuous is sufficient.)

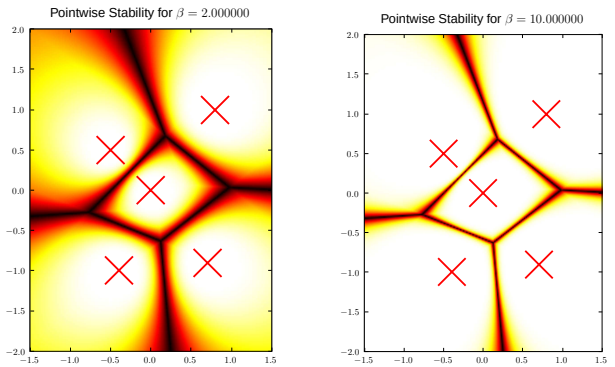
The implication of this for us is that clustering algorithms in high dimensions often produce a clustering based on very small relative differences in dimension. This does not necessarily imply the clustering is unstable; rather, it reflects the fact such small differences carry significant influence in determining the outcome. This observation is something that cluster validation schemes must take into considerations.

3.2.2 Tunable Priors

In our case – scaled distance perturbations, where the scaling is determined by a prior – this means that we must be able to adjust the severity of the scaled distance perturbations to accommodate this effect. In other words, we must be able to adjust $\pi(\boldsymbol{\lambda}|\theta)$ so that $\mathbb{P}(d_{ij}\lambda_j \leq d_{i\ell}\lambda_\ell \ \ \forall\, \ell)$ has the proper sensitivity to the magnitude of the relative differences between the d_{ij}'s. We suspect that being able to readily tune our method is one thing that gives it an edge over the other methods for testing stability in higher dimensions, as no other methods that we know of explicitly account for it.



(a) $\beta = 0.1$ (b) $\beta = 0.75$



(c) $\beta = 2$ (d) $\beta = 10$

Figure 3.1: Plots of the pointwise stability at each point of a 2d plain given the cluster centers.

Both of the prior classes we introduce in sections 3.3 and 3.4 can be tuned to be arbitrarily sensitive, a result we prove formally in chapter 6. Figure Figure 3.1 illustrates this by fixing the cluster centers and plotting the pointwise stability of the coordinates (x, y) around these centers. This is done for several values of the prior hyperparameter θ. As can be observed, as the value of θ increases, points must be much closer to the boundary regions before they are unstable. As discussed above, this is highly advantageous in higher dimensions.

In our case, we use a baseline null clustering – a clustered distribution of points in which any clusters found are spurious, random effects rather than true clusters – to tune the prior to account for the effects of high dimensions. The baseline data needed for our procedure is a point-to-centroid distance matrix that represents a clustered dataset that has no actual clusters. To tune the prior, $\pi(\boldsymbol{\lambda}|\theta)$, we choose the θ that maximizes the difference between the average pointwise stability of the clustering we are examining and the baseline clustering. Using this prior in our method, in principle, differentiates between a bad clustering, which has more points in the unstable boundary regions, and a good clustering, which does not.

Another way of saying this is that we choose the parameters for the prior as those which maximize the difference between the stability of the true data and the stability of the baseline. This causes the distance metric to "fit" the regions of stability to the true clusters. In the baseline, the density of the dataset around the boundary regions is roughly the same as that closer to the cluster centers. Thus maximizing the difference penalizes priors that force points to be very close to the boundary regions before they can be considered stable. Because the baseline can be made arbitrarily stable (assuming the priors we propose later in this chapter) by choosing the prior parameters so that only points very close to the boundary regions are unstable, this penalizes priors that cause too much of the space to be considered stable. Conversely, if the density around the cluster centers of the true data really is higher than that close to the boundaries, then making too little of the space will decrease the difference. The optimum balances the two, and in practice this works very well.

This can be formulated as an optimization problem. Let $\mathcal{APWD}(\theta)$ denote the difference in average pointwise stability, i.e.

$$\mathcal{APWD}(\theta') = \mathcal{APW}(\mathbf{D}, \theta') - \frac{1}{N_B} \sum_i \mathcal{APW}(\mathbf{D}_i^B, \theta') \tag{3.19}$$

$$\Rightarrow \theta = \operatorname*{argmax}_{\theta'} \mathcal{APW}(\mathbf{D}, \theta') - \frac{1}{N_B} \sum_i \mathcal{APW}(\mathbf{D}_i^B, \theta') \tag{3.20}$$

$$= \operatorname*{argmax}_{\theta'} \mathcal{APWD}(\theta') \tag{3.21}$$

where we express the averaged pointwise stability, $\mathcal{APW}$, as a function of the point to centroid distance matrix $\mathbf{D}$ and the prior hyperparameter θ. We use the average over N_B baseline distance matrices as the baseline $\mathcal{APW}$. Because it is an optimization problem, the algorithms are much more efficient if they can make use of the first derivative of $\mathcal{APWD}(\theta)$ with respect to θ; thus we also derive $\frac{\partial \phi_{ij}}{\partial \theta}$.

3.2.3 Types of Baseline Distance Matrices

Measuring the stability of a baseline null clustering is often used to account for dependence of the measure on non-clustering effects of the data. The gap statistic uses a clustering on a uniform distribution to adjust for the dependence in the measure on the number of clusters. However, the gap statistic is quite sensitive to the distribution of the null data since it depends on only the closest cluster. In comparison, calculating ϕ_{ij} depends on the distribution of distances, so it is far less sensitive to the null data. This allows us significantly more flexibility in how we generate the d_{ij}'s for the baseline.

Ultimately, we consider three types of baseline distance matrices to use to train the prior. The first comes from the baseline distribution proposed previously as part of the gap statistic [TWH01], namely reclustering a set of points distributed uniformly in the bounding box of the dataset along its PCA components. The second is also uses a uniform dataset in the same bounding box, but recycles the same cluster centers. The third uses an idea

from nonparametrics and simple uses a random permutation of the elements in the distance matrix. In chapter 5 we show that in higher dimensions and on more difficult datasets, all three baseline distributions produce comparable results, but the cheapest computationally, by far, is the third.

Reclustered, Uniformly Distributed Data

Tibshirani [TWH01] proposes running the clustering algorithm on a dataset uniformly distributed in the bounding box of the original data along its PCA components. They justify this empirically and by noting that in one dimension, a uniform distribution is provably the most likely to contain spurious clusterings and provide an intuitive argument that this generalizes into higher dimensions. One downside to this is that it does require running the clustering algorithm multiple times. In using this baseline for our method, we find that, in lower dimensions, it works best with data less easily described by simple, symmetric disstributions such as 2 dimensional ANOVA data. However, in higher dimensions and with less structured data, the next methods that we propose yield comparable or even superior results without requiring multiple runs of the clustering function.

Uniformly Distributed Data

In the gap statistic, the purpose of reclustering the data is to capture spurious clusterings in the dataset and thus account for those in the clustering on real data. However, since our purpose is to tune the prior, we care more about the estimated density of points between the cluster regions than about spurious clusterings. This provides some justification for simply recycling the centroids from the original dataset (we provide some empirical justification in chapter 5). This both saves the computational expense of rerunning the clustering algorithm and yields the natural geometric interpretation that tuning the prior is adjusting the sharpness of the geometric partitioning of the space.

Permutation of the Distance Matrix

This method, inspired by permutation tests from nonparametrics, simply uses a random permutation of all the elements in the distance matrix. Permutation tests [RW79] are used in non-parametric statistics as a way of operationalizing null hypothesis testing without having to specify a null distribution precisely. Here, we tune the prior to maximally distinguish between the null distribution and the true distribution. Furthermore, it has the advantage of being distribution free in the sense that only the original d_{ij}'s are needed; this can be a huge advantage for data types where it is not clear how to generate a null distribution.

3.3 Scaled-distance Perturbations with a Location Exponential Prior

The most useful prior we've found thus far on scaled distance perturbations is the exponential distribution with the location shifted by 1:

$$\pi(\lambda|\theta) = \mathcal{L}oc\mathcal{E}\chi p(\lambda|\theta) = \theta e^{-\theta(\lambda-1)}\mathbf{1}_{\lambda\geq 1}\,. \tag{3.22}$$

Note that since factors that scale the entire distribution drop out (section 3.1.2), it would be redundant to have parameters controling both the shift and the slope; thus we set the slope to 1. We denote the corresponding averaged assignment matrix with a superscript LE:

$$\mathbf{\Phi}^{\mathbf{LE}} = [\phi_{ij}^{\mathrm{LE}}] = \int \mathbb{I}[d_{ij}\lambda_j \leq d_{i\ell}\lambda_\ell \;\; \forall\, \ell] \left[\prod_\ell \mathcal{L}oc\mathcal{E}\chi p(\lambda_\ell|\theta_\ell)\right] \mathrm{d}\boldsymbol{\lambda} \tag{3.23}$$

Although we derive an analytical way to calculate this for the case when the parameter θ can vary between priors, in practice and in our tests we tied them all to the same value, so $\theta_\ell = \theta$. The equations for this case are a straightforward simplification of the more general result.

By shifting the exponential, $\phi_{ij} = \mathbb{P}(d_{ij}\lambda_j \leq d_{i\ell}\lambda_\ell \;\; \forall\, \ell)$ is no longer independent of the scaling parameter θ. This gives us a tuning parameter to

adjust the degree of perturbation – note that as θ increases, the prior puts its mass closer 1, causing ϕ_{ij} to resemble the indicator function $\mathbb{I}[d_{ij} \leq d_{i\ell} \ \forall \, \ell]$ unless the differences in $d_{i\cdot}$'s are sufficiently small. As we discuss in section 3.2.2, this allows us to optimally fit the perturbation to the clustering.

While it does make the derivation substantially more difficult than the more intuitive non-shifted exponential described in section 3.1.2, the calculation is still quite efficient. We derive an algorithm in section 3.3.1 that calculates $\mathbf{\Phi}^{\mathbf{LE}}$ in $\mathcal{O}(nK \log K)$. Additionally, in section 3.6, we use the lack of support in the prior distribution for $\lambda < 1$ to prove a rigorous and useful lower bound on the accuracy when we ignore sets of distances larger than an easily computed threshold.

3.3.1 Analytic Calculation of $\boldsymbol{\phi}^{\mathbf{LE}}$

In this section, we derive a computationally simple algorithm to compute $\boldsymbol{\phi}^{\mathbf{LE}}$, scaled distance perturbations with a location exponential prior. We do the calculation independently for each row of ϕ, so for notational convenience we drop the row index i.

Proposition 3.3.1. Calculation of $\boldsymbol{\phi}^{\mathbf{LE}}$

Suppose $\mathbf{d}$ *is a list of* K *distances and* $\boldsymbol{\theta}$ *is a list of* K *slope parameters for a location exponential prior. Let* ψ *be a bijective mapping of the indices,* $\{1, ..., K\} \mapsto \{1, ..., K\}$*, that puts* $\mathbf{d}$ *in sorted order, i.e.*

$$d_{\psi(1)} \leq d_{\psi(2)} \leq \cdots \leq d_{\psi(K)}. \tag{3.24}$$

Let

$$B_j = \sum_{k=1}^{j} \frac{\theta_{\psi(k)}}{d_{\psi(k)}} \tag{3.25}$$

$$C_j = \begin{cases} 1 & j = 1 \\ \exp\Big[-\sum_{k=1}^{j-1} \theta_{\psi(k)} \Big(\frac{d_{\psi(j)}}{d_{\psi(k)}} - 1 \Big) \Big] & j \in \{2, ..., K\} \end{cases} \tag{3.26}$$

$$D_j = \sum_{k=j+1}^{K} C_k \left[B_{k-1} \left(B_{k-1} \frac{d_{\psi(k)}}{\theta_{\psi(k)}} + 1 \right) \right]^{-1} \tag{3.27}$$

Then

$$\phi_j^{\mathrm{LE}} = \frac{\theta_j}{d_j} \left[\frac{C_{\psi^{-1}(j)}}{B_{\psi^{-1}(j)}} - D_{\psi^{-1}(j)} \right] \tag{3.28}$$

Proof. From equation (3.23), we have that

$$\phi_j^{\mathrm{LE}} = \int \pi_j(\lambda_j, \theta_j) \prod_{\ell \neq j} \mathbb{I}[d_j \lambda_j \leq d_\ell \lambda_\ell] \pi_\ell(\lambda_\ell, \theta_\ell) \mathrm{d}\lambda_\ell \mathrm{d}\lambda_j \tag{3.29}$$

$$= \int_1^\infty \theta_j e^{-\theta_j(\lambda_j - 1)} \prod_{\ell \neq j} \int_1^\infty \theta_\ell e^{-\theta_\ell(\lambda_\ell - 1)} \int_0^\infty \delta((d_\ell \lambda_\ell - d_j \lambda_j) - t_\ell) \mathrm{d}t_\ell \mathrm{d}\lambda_\ell \, \mathrm{d}\lambda_j \tag{3.30}$$

where $\delta(\cdot)$ is the Dirac delta function, defined such that $\int_a^b \delta(x-t)\mathrm{d}t$ equals one if $a \leq x \leq b$ and zero otherwise, so $\mathbf{1}_{a \leq b} = \int_0^\infty \delta(b-a-t)\mathrm{d}t$, and $f(x) = \int f(t)\delta(x-t)\mathrm{d}t$ [AWR96]. We can then switch the order of integration and integrate over λ_ℓ first:

$$\phi_j^{\text{LE}} = \int_1^\infty \theta_j e^{-\theta_j(\lambda_j - 1)} \prod_{\ell \neq j} \int_0^\infty \frac{\theta_\ell}{d_\ell} e^{-\theta_\ell\left(\frac{t_\ell + d_j\lambda_j}{d_\ell} - 1\right)} \mathbf{1}_{\left[\frac{t_\ell + d_j\lambda_j}{d_\ell} \geq 1\right]} \, \mathrm{d}t_\ell \, \mathrm{d}\lambda_j \tag{3.31}$$

$$= \int_1^\infty \theta_j e^{-\theta_j(\lambda_j - 1)} \prod_{\ell \neq j} \left[\exp\left(-\theta_\ell \left(\frac{d_j}{d_\ell}\lambda_j - 1 \right) - \frac{\theta_\ell}{d_\ell} t_\ell \right) \Big|_{t_\ell = \max(d_\ell - d_j\lambda_j, 0)} \right. \mathrm{d}\lambda_j \tag{3.32}$$

$$= \int_1^\infty \theta_j e^{-\theta_j(\lambda_j - 1)} \prod_{\ell \neq j} \left\{ \begin{array}{ll} \exp\left[-\theta_\ell \left(\frac{d_j}{d_\ell}\lambda_j - 1 \right) \right] & \lambda_j \geq \frac{d_\ell}{d_j} \\ 1 & \text{otherwise} \end{array} \right\} \mathrm{d}\lambda_j \tag{3.33}$$

At this point, we can use the fact that many of the terms in the product are one for certain regions of the integration to break the region of integration into sections. If we sort the terms by increasing d_ℓ, we can handle the conditional terms in the product by breaking the integration up into $K+1$ possibly empty intervals, doing the integration separately on each interval.

Recall that ψ is a bijective mapping such that $d_{\psi(1)} \leq d_{\psi(2)} \leq \cdots \leq d_{\psi(K)}$. Then the boundaries of these regions are given by

$$A_m = \left\{ \begin{array}{ll} d_{\psi(m)}/d_j & m \in \{1, ..., K\} \\ \infty & m = K+1 \end{array} \right. \tag{3.34}$$

The intervals to integrate over are then

$$(0, A_1] \cap [1, \infty), \ (A_1, A_2] \cap [1, \infty), \ ..., \ (A_K, A_{K+1} = \infty) \cap [1, \infty).$$

As $A_m = d_{\psi(m)}/d_j \leq 1$ for $m < \psi^{-1}(j)$, the first $\psi^{-1}(j)$ of these intervals are empty. Thus the integration becomes

$$\phi_j^{\text{LE}} = \sum_{k=\psi^{-1}(j)}^{K} \int_{A_k}^{A_{k+1}} \theta_j e^{-\theta_j(\lambda_j - 1)} \prod_{\substack{m=1 \\ \psi(m)\neq j}}^{k} e^{-\theta_{\psi(m)}(A_m^{-1}\lambda_j - 1)} \mathrm{d}\lambda_j \tag{3.35}$$

$$= \sum_{k=\psi^{-1}(j)}^{K} \int_{A_k}^{A_{k+1}} \theta_j \exp\left[-\sum_{m=1}^{k} \theta_{\psi(m)}(A_m^{-1}\lambda_j - 1) \right] \mathrm{d}\lambda_j \tag{3.36}$$

where we've used the fact that $A_{\psi^{-1}(j)} = 1$ to simplify the expression. The integral can now be easily evaluated:

$$\phi_j^{\text{LE}} = \sum_{k=\psi^{-1}(j)}^{K} \theta_j \exp\left[\sum_{m=1}^{k} \theta_{\psi(m)} \right] \int_{A_k}^{A_{k+1}} \exp\left[-\left(\sum_{m=1}^{k} \theta_{\psi(m)} A_m^{-1} \right) \lambda_j \right] \mathrm{d}\lambda_j \tag{3.37}$$

$$= \sum_{k=\psi^{-1}(j)}^{K} \left[\theta_j \frac{\exp\left[\sum_{m=1}^{k} \theta_{\psi(m)} \right]}{\sum_{m=1}^{k} \theta_{\psi(m)} A_m^{-1}} \exp\left[\sum_{m=1}^{k} \theta_{\psi(m)} A_m^{-1} \lambda_j \right] \right]_{\lambda_j = A_{k+1}}^{\lambda_j = A_k} \tag{3.38}$$

If we collect similar terms and use the given definitions of C and D, this can be expressed as:

$$\phi_j^{\text{LE}} = \frac{\theta_j}{d_j} \sum_{k=\psi^{-1}(j)}^{K} \frac{1}{B_k} (C_k - C_{k+1}) \tag{3.39}$$

Now the C terms may be quite close, so for better numerical stability we can re-express equation (3.39) to avoid working with the difference of two similar numbers:

$$\phi_j^{\text{LE}} = \frac{\theta_j}{d_j} \left[\frac{C_{\psi^{-1}(j)}}{B_{\psi^{-1}(j)}} - \frac{C_{\psi^{-1}(j)+1}}{B_{\psi^{-1}(j)}} + \frac{C_{\psi^{-1}(j)+1}}{B_{\psi^{-1}(j)+1}} - \frac{C_{\psi^{-1}(j)+2}}{B_{\psi^{-1}(j)+1}} + \cdots + \frac{C_K}{B_K} \right] \tag{3.40}$$

where the arguments to ϕ_j^{LE}, B, and C are implied. Now for better numerical stability – needed here as B_{k-1} and B_k are likely to be close together –

$B_{k-1}^{-1} - B_k^{-1}$ can be expressed as

$$\frac{1}{B_{k-1}} - \frac{1}{B_k} = \frac{1}{B_{k-1}} - \left[B_{k-1} + \frac{\theta_{\psi(k)}}{d_{\psi(k)}}\right]^{-1} \tag{3.41}$$

$$= \left[B_{k-1}\left(B_{k-1}\frac{d_{\psi(k)}}{\theta_{\psi(k)}} + 1\right)\right]^{-1} \tag{3.42}$$

Putting this back into equation (3.40) gives:

$$\phi_j^{\text{LE}} = \frac{\theta_j}{d_j}\left[\frac{C_{\psi^{-1}(j)}}{B_{\psi^{-1}(j)}} - \sum_{k=\psi^{-1}(j)+1}^{K} C_k\left[B_{k-1}\left(B_{k-1}\frac{d_{\psi(k)}}{\theta_{\psi(k)}} + 1\right)\right]^{-1}\right] \tag{3.43}$$

$$= \frac{\theta_j}{d_j}\left[\frac{C_{\psi^{-1}(j)}}{B_{\psi^{-1}(j)}} - D_{\psi^{-1}(j)}\right] \tag{3.44}$$

which completes the theorem. □

We can precompute common terms to create an efficient and accurate algorithm, which we present in algorithm (1). The running time of this algorithm is linear in K save for the call to argsort ($\mathbf{d}$) to sort the distances, which runs in $\mathcal{O}(K \log K)$ time. Thus the overall running time is $\mathcal{O}(K \log K)$.

3.3.2 Analytic Calculation of $\partial\phi_j^{\text{LE}}/\partial\theta_\ell$

As we discussed previously, we need to find the optimum value for θ by maximizing the difference between the averaged pointwise stability of the clustering and that of the null distance matrix. While there are algorithms that do not require the gradient or first derivative of the function, these can take a very long time, especially in higher dimensions. This relates directly to the $\mathcal{APW}$, as

Algorithm 1: Calculation of scaled distance perturbations with a location exponential prior.

Input: A vector $\mathbf{d}$ of distances and a vector $\boldsymbol{\theta}$ of prior parameters.
Output: A probability vector $\boldsymbol{\phi}^{\text{LE}}$ of partial memberships.

$K \leftarrow \text{length}\mathbf{d}$, $\boldsymbol{\psi} \leftarrow \text{argsort}\,(\mathbf{d})$

Initialize $d^{\mathbf{s}}$, $\theta^{\mathbf{s}}$, $\mathbf{B}$, $\mathbf{C}$, $\mathbf{D}$, $\mathbf{r}$ as vectors of length K.

for $j = 1$ **to** K **do** $d_j^{\mathbf{s}} \leftarrow d_{\psi(j)}$, $\theta_j^{\mathbf{s}} \leftarrow \theta_{\psi(j)}$, $r_j \leftarrow \theta_j^{\mathbf{s}}/d_j^{\mathbf{s}}$

$B_1 \leftarrow r_1$, $t \leftarrow 0$, $C_1 \leftarrow 1$
for $j = 2$ **to** K **do**
  $B_j \leftarrow B_{j-1} + rj$
  $t \leftarrow t + \theta_{i-1}^{\mathbf{s}}$
  $C_j \leftarrow \exp\big(t - B_{j-1} d_j^{\mathbf{s}}\big)$
end

$D_K \leftarrow 0$
for $j = K - 1$ **to** 1 **step** -1 **do**
  $D_j \leftarrow D_{j+1} + C_{j+1}/\big[B_j\big(B_j r_{j+1} + 1\big)\big]$
end

for $j = 1$ **to** K **do** $\phi_{\psi(j)}^{\text{LE}} \leftarrow r_j\big(C_j/B_j - D_j\big)$

return $\boldsymbol{\phi}^{\mathbf{LE}}$

$$\frac{\partial}{\partial\theta_\ell}\mathcal{APW} = \operatorname*{mean}_i \frac{\partial}{\partial\theta_\ell}\mathcal{PW}_i \tag{3.45}$$

$$\frac{\partial}{\partial\theta_\ell}\mathcal{PW}_i = \frac{\partial\phi_{ih_i}^{\text{LE}}}{\partial\theta_\ell} - \frac{\partial\phi_{ig_i}^{\text{LE}}}{\partial\theta_\ell} \tag{3.46}$$

$$\tag{3.47}$$

where

$$h_i = \operatorname*{argmax}_k a_{ik} \tag{3.48}$$

$$g_i = \operatorname*{argmax}_{\ell \neq h_i} \phi_{ik} \tag{3.49}$$

To make the optimization process more efficient, we present here the derivation for $\partial\phi_j^{\text{LE}}/\partial\theta_\ell$.

As $\nabla_{\boldsymbol{\theta}}\,\mathcal{AR}^\star$ and $\nabla_{\boldsymbol{\theta}}\,\mathcal{VI}^\star$ depend on the partial derivatives of ϕ^{LE} with respect to $\boldsymbol{\theta}$, we here derive an algorithm to calculate $\left[\partial\phi_j^{\text{LE}}/\partial\theta_\ell\right]$ in amortized constant time. For simplicity, we assume in our derivations that the d's are already in sorted order; when we give the final algorithm we return to using the sorted index map. Then, using ϕ^{LE} from equation (3.28), we have

$$\frac{\partial\phi_j^{\text{LE}}}{\partial\theta_\ell} = \frac{\partial}{\partial\theta_\ell}\frac{\theta_j}{d_j}\left[\frac{C_j}{B_j} - D_j\right] \tag{3.50}$$

$$= \frac{\theta_j}{d_j}\left(\frac{\partial}{\partial\theta_\ell}\frac{C_j}{B_j} - \frac{\partial D_j}{\partial\theta_\ell}\right) + \frac{1}{d_j}\left(\frac{C_j}{B_j} - D_j\right)\mathbf{1}_{\ell=j} \tag{3.51}$$

Now as everything builds on previous derivatives, let's start with the derivatives of B_j, C_j:

$$\frac{\partial B_j}{\partial\theta_\ell} = \frac{\partial}{\partial\theta_\ell}\sum_{m=1}^{j}\frac{\theta_{\psi(m)}}{d_{\psi(m)}} = \frac{1}{d_\ell}\mathbf{1}_{\ell\le j} \tag{3.52}$$

$$\frac{\partial C_j}{\partial\theta_\ell} = \frac{\partial}{\partial\theta_\ell}\exp\left[-\left(B_{j-1}d_j + \sum_{m=1}^{j-1}\theta_m\right)\mathbf{1}_{j\ge 2}\right] \tag{3.53}$$

$$= C_j\left(1 - \frac{d_j}{d_\ell}\right)\mathbf{1}_{\ell<j} \tag{3.54}$$

$$\Rightarrow \frac{\partial}{\partial\theta_\ell}\frac{C_j}{B_j} = \frac{C_j}{B_j}\left(1 - \frac{d_j}{d_\ell}\right)\mathbf{1}_{\ell<j} - \frac{C_j}{B_j^2 d_\ell}\mathbf{1}_{\ell\le j} \tag{3.55}$$

$$\frac{\partial}{\partial\theta_\ell}D_j = \frac{\partial}{\partial\theta_\ell}\sum_{k=j+1}^{K}\frac{C_k}{E_k} = \sum_{k=j+1}^{K}\frac{1}{E_k}\frac{\partial C_k}{\partial\theta_\ell} - \frac{C_k}{E_k^2}\frac{\partial E_k}{\partial\theta_\ell} \tag{3.56}$$

Algorithm 2: Calculation of $\frac{\partial}{\partial\phi_j}\theta_\ell$ for scaled distance perturbations with a location exponential prior and different prior parameters for each K.

Input: A vector $\mathbf{d}$ of distances and a vector $\boldsymbol{\theta}$ of prior parameters.

Output: A $K \times K$ matrix $\mathbf{R}$ giving $\frac{\partial\phi_j^{\mathrm{IE}}}{\partial\theta_\ell}$.

$K \leftarrow \text{length}(\mathbf{d})$, $\boldsymbol{\psi} \leftarrow \text{makeSortingIndexMap}(d)$

Initialize d^{s}, θ^{s}, $\mathbf{B}$, $\mathbf{C}$, $\mathbf{D}$, $\mathbf{E}$, $\mathbf{F}$, $\mathbf{G}$, $\mathbf{H}$, $\mathbf{L}$, $\mathbf{r}$ as vectors of length K.
Initialize $\mathbf{R}$ as a $2d$ array of size $K \times K$.

for $j = 1$ **to** K **do** $\quad d_j^{\mathsf{s}} \leftarrow d_{\psi(j)}$, $\theta_j^{\mathsf{s}} \leftarrow \theta_{\psi(j)}$, $r_j \leftarrow \theta_j^{\mathsf{s}}/d_j^{\mathsf{s}}$

`// Some common terms to simplify the computation.`
$B_1 \leftarrow r_1$, $t \leftarrow 0$, $C_1 \leftarrow 1$, $H_1 \leftarrow 1/r_1$
for $j = 2$ **to** K **do**
  $B_j \leftarrow B_{j-1} + r_j$
  $t \leftarrow t + \theta_{j-1}^{\mathsf{s}}$
  $C_j \leftarrow \exp\big(t - B_{j-1}d_j^{\mathsf{s}}\big)$
  $E_j \leftarrow B_{j-1}(B_{j-1}/r_j + 1)$
  $L_j \leftarrow C_j/E_j$
  $G_j \leftarrow L_j B_{j-1}^2/E_j r_j \theta_j^{\mathsf{s}}$
  $H_j \leftarrow C_j/B_j$
end

`// Now the terms that build on each other.`
$D_K \leftarrow 0$, $F_K \leftarrow 0$
for $j = K - 1$ **to** 1 **step** -1 **do**
  $D_j \leftarrow D_{j+1} + L_{j+1}$
  $F_j \leftarrow F_{j+1} + L_{j+1}\big[d_{j+1}^{\mathsf{s}} + (2B_j/r_{j+1} + 1)/E_{j+1}\big]$
end

`// Finally, calculate` $\mathbf{R}$.
for $m = 1$ **to** K **do**
  $R_{\psi(m),\psi(m)} \leftarrow r_m[F_m/d_m^{\mathsf{s}} - D_m - H_m/B_m d_m^{\mathsf{s}}] + (H_m - D_m)/d_m^{\mathsf{s}}$
  for $n = m + 1$ **to** K **do**
    $R_{\psi(m),\psi(n)} \leftarrow r_m(F_n/d_n^{\mathsf{s}} - D_n - G_n)$
    $R_{\psi(n),\psi(m)} \leftarrow r_n\big[F_n/d_m^{\mathsf{s}} - D_n + \big(H_n/d_m^{\mathsf{s}}\big)\big(d_m^{\mathsf{s}} - d_n^{\mathsf{s}} - 1/B_n\big)\big]$
  end
end
return $\mathbf{R}$

where $E_k = \left(B_{k-1}^2\, d_k/\theta_k + B_{k-1}\right)\mathbf{1}_{k\geq 2}$. Continuing:

$$\frac{\partial E_k}{\partial \theta_k} = \frac{\partial}{\partial \theta_\ell}\left[B_{k-1}\left(B_{k-1}\frac{d_k}{\theta_k}+1\right)\right] \tag{3.57}$$

$$= \left(\frac{2B_{k-1}d_k}{\theta_k}+1\right)\frac{1}{d_\ell}\mathbf{1}_{2\leq \ell<k} - \frac{B_{k-1}^2 d_k}{\theta_k^2}\mathbf{1}_{2\leq \ell=k} \tag{3.58}$$

$$\Rightarrow \frac{\partial D_j}{\partial \theta_\ell} = \sum_{k=j+1}^{K}\frac{C_k}{E_k}\left[1-\frac{1}{d_\ell}\left(d_k+\frac{2B_{k-1}d_k}{E_k\theta_k}+\frac{1}{E_k}\right)\right]\mathbf{1}_{\ell<k} + \frac{B_{k-1}^2 C_k d_k}{\theta_k^2 E_k^2}\mathbf{1}_{\ell=k} \tag{3.59}$$

$$= \sum_{k=j+1}^{K}\begin{cases} \frac{C_k}{E_k}\left[1-\frac{d_k}{d_\ell}-\frac{1}{E_k d_\ell}\left(\frac{2B_{k-1}d_k}{\theta_k}+1\right)\right] & \ell<k \\ \frac{C_k B_{k-1}^2 d_k}{\theta_k^2 E_k^2} & \ell=k \\ 0 & \ell>k \end{cases} \tag{3.60}$$

Grouping terms dependent only on k together allows us to express this in a form that can be calculated in constant time per j, ℓ:

$$\frac{\partial}{\partial \theta_\ell} D_j = \begin{cases} D_j - \frac{F_j}{d_\ell} & \ell \leq j \\ \frac{C_\ell B_{\ell-1}^2 d_\ell}{\theta_\ell^2} + D_\ell - \frac{F_\ell}{d_\ell} & \ell > j \end{cases} \tag{3.61}$$

where

$$F_j = \sum_{k=j+1}^{K}\frac{C_k}{E_k}\left[\left(1+\frac{2B_{k-1}}{E_k\theta_k}\right)d_k + \frac{1}{E_k}\right] \tag{3.62}$$

and can be computed beforehand. The final algorithm is then given in algorithm (2). Calculating all the intermediate arrays takes only $\mathcal{O}(K)$ time, and any sorting is only $\mathcal{O}(K \log K)$, so the algorithm takes $\mathcal{O}\left(K^2\right)$ time overall. Obviously, much of the calculation overlaps with algorithm (1) and in practice the two should be calculated together.

Algorithm 3: Calculation of scaled distance perturbations with a tied location exponential prior.

Input: A vector $\mathbf{d}$ of distances and a scalar parameter θ giving the prior.

Output: A probability vector ϕ^{LE} of partial memberships and a vector $\mathbf{R}$ giving $\partial\phi_j^{\text{LE}}/\partial\theta$.

$K \leftarrow \text{length}\mathbf{d}$, $\boldsymbol{\psi} \leftarrow \text{argsort}\,(\mathbf{d})$

Initialize d^{s}, $\mathbf{B}$, $\mathbf{C}$, $\mathbf{D}$, $\mathbf{E}$, $\mathbf{F}$, $\mathbf{G}$, $\mathbf{H}$, and $\mathbf{R}$ as vectors of length K.

for $j = 1$ **to** K **do** $d_i^{\text{s}} \leftarrow d_{\psi(i)}$,

$B_1 \leftarrow 1/d_0^{\text{s}}$, $D_1 \leftarrow 1$, $E_1 \leftarrow d_1^{\text{s}}$
for $j = 2$ **to** K **do**
  $B_j \leftarrow B_{j-1} + 1/d_j^{\text{s}}$
  $C_j \leftarrow j - d_j^{\text{s}} B_{j-1}$
  $D_j \leftarrow \exp[\theta C_j]$
  $E_j \leftarrow D_j/B_j$
  $F_j \leftarrow D_j/\Big(B_{j-1}(d_j^{\text{s}} B_{j-1} + 1)\Big)$
end

$G_K \leftarrow 0$
for $j = K$ **to** 2 **step** -1 **do** $G_{j-1} \leftarrow G_j + F_j$

for $j = 1$ **to** K **do** $\phi_{\psi(j)}^{\text{LE}} \leftarrow (E_j - G_j)/d_j^{\text{s}}$,

`// The rest of the code calculates` $\partial\phi_{\psi(j)}^{\text{LE}}/\partial\theta$.
$H_K \leftarrow 0$,
for $j = K$ **to** 1 **step** -1 **do** $H_{j-1} \leftarrow H_j + C_j F_j$

for $j = 1$ **to** K **do** $R_j \leftarrow (C_j E_j - H_j)/d_j^{\text{s}}$

3.3.3 Analytic Calculation of ϕ^{LE} and $\partial\phi_j^{\text{LE}}/\partial\theta$ for Tied θ

In many cases we want to tie the prior hyperparameters together, so $\theta_\ell = \theta$ and all K perturbation parameters have the same prior. In this case, we can simplify some aspects of the calculation. The derivations are very similar to the ones for multiple θ, thus we leave out the tedious detains. Instead presenting the final algorithm to calculate both versions in algorithm (3). As can be seen with a quick comparison – especially the part which calculates the first derivative – is quite simple compared to the part for multiple θ's.

Algorithmically, though, the running time is the same for calculating ϕ^{JE}, $\mathcal{O}(K \log K)$. This is again because of the call to argsort; everything else has linear running time. Calculating $\partial \phi_j^{\text{JE}} / \partial \theta$, however, is now linear in K instead of quadratic, so that gives us a significant speedup.

In the end, $\mathbf{\Phi}^{\text{LE}}$ with tied prior hyperparameters is our preferred method, and we illustrate in chapter 5 that its performance is indeed comparable with many other leading methods.

3.4 Scaled Distance Perturbations with a Shifted Gamma Prior

In this section, we slightly extend the results for the location exponential distribution described in section 3.3 by using a shifted version of the $\mathit{Gamma}(\alpha = 2, \beta)$ prior:

$$\pi(\lambda|\alpha, \beta) = Z(\alpha, \beta)(\lambda - 1)^{\alpha-1} e^{-\beta(\lambda-1)} \mathbf{1}_{\lambda \geq 1} \tag{3.63}$$

$$= \mathit{ShiftedGamma}(\lambda|\alpha, \beta) \tag{3.64}$$

where $Z(\alpha, \beta)$ is a normalizing constant. This simple addition significantly complicates the final equations, raising the overall time complexity to $\mathcal{O}\big(nK^3\big)$. Additionally, because of numerical instability issues, we are forced to to use arbitrary precision variables in evaluating the final algorithm. One way around this, which also has its downsides, is to use the Monte Carlo integration method outlined in section 3.5. This method is still reasonably efficient, however, for small K, so we present it here. The derivations are straightforward, albeit messy.

3.4.1 Analytic Calculation of ϕ^{SG}

As in the derivation with the location exponential prior, we drop all the i's for notational convenience; this calculation happens once per data point i. We begin the derivation by noting that

$$\phi_j^{\text{SG}} = \int \mathbb{I}[d_j\lambda_j \leq d_\ell\lambda_\ell \ \forall\, \ell \neq j] \prod_\ell \mathit{ShiftedGamma}(\lambda_\ell; \alpha_\ell, \beta_\ell)\mathrm{d}\boldsymbol{\lambda} \tag{3.65}$$

$$= \int \mathbb{I}[d_j\lambda_j + d_j \leq d_\ell\lambda_\ell + d_\ell \ \forall\, \ell \neq j] \prod_\ell \mathit{Gamma}(\lambda_\ell; \alpha_\ell, \beta_\ell)\mathrm{d}\boldsymbol{\lambda} \tag{3.66}$$

$$= \int \mathbb{I}[\beta d_j\lambda_j + d_j \leq \beta d_\ell\lambda_\ell + d_\ell \ \forall\, \ell \neq j] \prod_\ell \mathit{Gamma}(\lambda_\ell; \alpha_\ell, 1)\mathrm{d}\boldsymbol{\lambda}. \tag{3.67}$$

We eventually derive the case for $\alpha_\ell = \alpha = 2$ (recall that the $\alpha = 1$ case is the location exponential prior); other values would be better handled by using Monte Carlo integration.

Now assume that the d's are in sorted order, so

$$d_1 \leq d_2 \leq \cdots \leq d_K. \tag{3.68}$$

This can be accomplished by mapping the indices before and remapping them afterwords. For conciseness, we here omit the explicit index mapping used in the derivations of the location exponential prior, but the principle is the same.

Recall from the derivation of ϕ^{LE} that we can rewrite an indicator function using a Dirac delta function. This function is defined such that $\int_a^b \delta(x-t)\mathrm{d}t$ equals one if $a \leq x \leq b$ and zero otherwise, so $\mathbf{1}_{a\leq b} = \int_0^\infty \delta(b-a-t)\mathrm{d}t$, and $f(x) = \int f(t)\delta(x-t)\mathrm{d}t$ [AWR96]. Thus equation (3.65) becomes:

$$\phi_j^{\text{SG}} = \int \prod_\ell \pi(\lambda_\ell|\alpha,\beta)\delta((\lambda_\ell\beta_\ell d_\ell + d_\ell) - (\lambda_j\beta_j d_j + d_j) - t_\ell)\mathrm{d}t_\ell\mathrm{d}\boldsymbol{\lambda}. \tag{3.69}$$

Using the Dirac delta function again,

$$\phi_j^{\text{SG}} = \int \pi(\lambda_j) \prod_{\ell\neq j} \int \pi\left(\frac{1}{\beta_\ell d_\ell}(t_\ell + \beta_j d_j\lambda_j + d_j - d_\ell)\right)\frac{1}{\beta_\ell d_\ell}\mathrm{d}t_\ell\mathrm{d}\lambda_j \tag{3.70}$$

We set the shape parameter α to 2 to make the derivation tractable while

still preserving the desired shape. At this point, we set $\alpha = 2$, so $\pi(\lambda_\ell)$ is

$$\pi(\lambda_\ell) = \lambda_\ell e^{-\lambda_\ell}. \tag{3.71}$$

Defining

$$B_{j\ell}(\lambda) = \frac{\beta_j d_j \lambda_j + d_j - d_\ell}{\beta_\ell d_\ell} \tag{3.72}$$

for convenience, we have

$$\phi_j^{\text{SG}} = \int \lambda_j e^{-\lambda_j} \prod_{\ell \neq j} \left[t_\ell + B_{j\ell}(\lambda)\right] e^{-t_\ell - B_{j\ell}(\lambda)} \mathbb{I}[t_\ell + B_{j\ell}(\lambda) \geq 0] \, \mathrm{d}t_\ell \mathrm{d}\lambda_j \tag{3.73}$$

$$= \int \lambda_j e^{-\lambda_j} \prod_{\ell \neq j} \left[(1 + B_{j\ell}(\lambda)) e^{-B_{j\ell}(\lambda)}\right]^{\mathbb{I}[B_{j\ell}(\lambda) \geq 0]} \mathrm{d}\lambda_j \tag{3.74}$$

To evaluate the integral, we can use the division points at

$$B_{j\ell}(\lambda) = 0 \Leftrightarrow \lambda_j = \frac{d_\ell - d_j}{\beta_j d_j} \tag{3.75}$$

to break it up into at most $K + 1$ discrete regions and integrate each separately. Let

$$A_j^\ell = \frac{d_\ell - d_j}{\beta_j d_j} \tag{3.76}$$

denote the division points (recall that $d_\cdot$ is in sorted order), and for convenience, let $A_j^{K+1} = \infty$. Note that $A_j^j = 0$, so we only need to integrate between the division points $A_j^j, A_j^{j+1}, ..., A_j^{K+1}$ to calculate ϕ_j^{SG}:

$$\phi_j^{\text{SG}} = \sum_{m=j}^{K} \int_{A_j^m}^{A_j^{m+1}} \lambda_j e^{-\lambda_j} \prod_{\substack{\ell=1 \\ \ell \neq j}}^{m=K} (1 + B_{j\ell}(\lambda)) e^{-B_{j\ell}(\lambda)} \mathrm{d}\lambda_j \tag{3.77}$$

We use

$$C_j = \beta_j d_j \lambda_j + d_j \tag{3.78}$$

to transform the variable of integration, yielding, after some straightforward but tedious algebra:

$$\phi_j^{\text{SG}} = \sum_{m=j}^{K} \int_{d_m}^{d_{m+1}} \prod_{\substack{\ell=1 \\ \ell \neq j}}^{m=K} \left[1 - \frac{1}{\beta_\ell} + \frac{C_j}{\beta_\ell d_\ell}\right] \left[\frac{C_j - d_j}{\beta_j d_j}\right] e^{-\sum_{\ell=1}^{m} \frac{C_j - d_\ell}{\beta_\ell d_\ell}} \frac{1}{\beta_j d_j} \mathrm{d}C_j \tag{3.79}$$

$$= \frac{1}{\beta_j d_j} \sum_{m=j}^{K} \int_{d_m}^{d_{m+1}} \left[P_m(C_j) - P_m(C_j) \frac{\beta_j d_j}{\beta_j d_j - d_j + C_j}\right] e^{-\sum_{\ell=1}^{m} \frac{C_j - d_\ell}{\beta_\ell d_\ell}} \mathrm{d}C_j \tag{3.80}$$

where $P_m(y)$ is a polynomial, equal to

$$P_m(y) = \prod_{\ell=1}^{m} \left[1 - \frac{1}{\beta_\ell} + \frac{y}{\beta_\ell d_\ell}\right] \tag{3.81}$$

$$= \sum_{k=0}^{m} p_k^m y^k. \tag{3.82}$$

We can evaluate the integral by first calculating the coefficients p_k^m using recursion. We then need to integrate the difference of two polynomials times an exponential, so the result is in the same form:

$$\int \sum_{k=0}^{m} \left(p_k^m - \frac{\beta_j d_j p_k^m}{\beta_j d_j - d_j + y}\right) y^k e^{-\frac{y}{E_m}} \mathrm{d}y = \sum_{k=0}^{m} \left(q_k^m - r_k^{jm}\right) y^k e^{-\frac{y}{E_m}} \tag{3.83}$$

where

$$E_m = \left[\sum_{\ell=1}^{m} \frac{1}{\beta_\ell d_\ell}\right]^{-1}. \tag{3.84}$$

We can calculate q_k^m and integrating successively lower powers of C_j, resulting in another recurrence. Likewise, the division and integration on r_k^{jm} can be expressed as a recurrence. Finally, we have

$$\phi_j^{\text{SG}} = \frac{1}{\beta_j d_j}\left[G_{j,\ell=j}(d_j)e^{-F_j} + \sum_{\ell=j+1}^{K}\left(G_{j\ell}(d_\ell) - G_{j,\ell-1}(d_\ell)\right)e^{-F_\ell}\right] \tag{3.85}$$

where

$$G_{jm}(y) = \sum_{k=0}^{m}\left(q_k^m - r_k^{jm}\right)y^k \tag{3.86}$$

$$F_\ell = \sum_{m=1}^{\ell}\frac{d_m - d_\ell}{\beta_\ell d_\ell} \tag{3.87}$$

and

$$p_k^m = \begin{cases} \beta_m d_m^{-1}\left[p_{k-1}^{m-1} + d_m(\beta_m - 1)p_k^{m-1}\right] & 0 \le k \le m, 1 \le m \le K \\ 1 & m = k = 0 \\ 0 & \text{otherwise} \end{cases} \tag{3.88}$$

$$q_k^m = \begin{cases} E_m\left[(k+1)q_{k+1}^m + p_k^m\right] & 0 \le k \le m, 1 \le m \le K \\ 0 & \text{otherwise} \end{cases} \tag{3.89}$$

$$r_k^{jm} = \begin{cases} \left[E_m(k+1) - d_j(\beta_j - 1)\right]r_{k+1}^{jm} + E_m d_j\left[(\beta_j - 1)(k+2)r_{k+2}^{jm} + \beta_j p_{k+1}^m\right] & \\ & 0 \le k \le m-1, 1 \le m \le K \\ 0 & \text{otherwise} \end{cases} \tag{3.90}$$

For the general case, evaluating $G_{jim}(\cdot)$ takes $\mathcal{O}(K)$ time per m for $\mathcal{O}\big(K^2\big)$ time per i, j pair and $\mathcal{O}\big(nK^3\big)$ overall. Note that calculating p_k^m

once per i is sufficient.

In practice, this recursion is subject to several points of numerical failure due to taking differences between similar terms. For our calculations, we wrapped the result in a routine that verifies that the sum of the entries was sufficiently close to 1, and that all of the individual entries were between 0 and 1. All the numerical errors we detected would cause these tests to fail. If they did, we reran the algorithm again, doubling the precision. This approach was quick and accurate enough for all the experiments we ran using it.

3.5 General Monte Carlo Algorithm

Algorithm 4: Monte Carlo Calculation of $\boldsymbol{\phi}$.

Input: A vector $\mathbf{d}$ of distances and a vector $\boldsymbol{\theta}$ of prior parameters and sample size N.

Output: A probability vector $\boldsymbol{\phi}$ of partial memberships and matrix $\mathbf{R}$ giving $\partial\phi_j/\partial\theta_\ell$.

$K \leftarrow \text{length}(\mathbf{d})$, $N_p \leftarrow \text{length}(\boldsymbol{\theta})$
$\boldsymbol{\phi} \leftarrow \mathbf{0}_K$, $\mathbf{R} \leftarrow \mathbf{0}_{K\times N_p}$
for $n = 1$ **to** N **do**
  draw $\boldsymbol{\lambda}$ **from** $\pi(\theta)$
  for $j = 1$ **to** K **do**
    $\phi_j \leftarrow \phi_j + f_j(\boldsymbol{\lambda}, \mathbf{d})$
    for $\ell = 1$ **to** N_p **do**

$$R_{j\ell} \leftarrow R_{j\ell} + f_j(\boldsymbol{\lambda}, \mathbf{d}) \left[\frac{\partial \log \pi(\boldsymbol{\lambda}', \boldsymbol{\theta})}{\partial \theta_\ell} \right|_{\boldsymbol{\lambda}'=\boldsymbol{\lambda}}$$

    end
  end
end
$\boldsymbol{\phi} \leftarrow \boldsymbol{\phi}/N$, $\mathbf{R} \leftarrow \mathbf{R}/N$
return $\boldsymbol{\phi}$, $\mathbf{R}$

In this section, we develop a more general Monte Carlo algorithm to calculate ϕ_{ij} when the priors $\pi_\ell(\lambda_\ell)$, $\ell = 1, 2, ..., K$, can be sampled from, but when analytically evaluating equation (2.3) is impossible or computationally

Algorithm 5: Monte Carlo Calculation of $\boldsymbol{\phi}$ for Scaled Distance Perturbations.

Input: A vector $\mathbf{d}$ of distances and a vector $\boldsymbol{\theta}$ of prior parameters and sample size N.

Output: A probability vector $\boldsymbol{\phi}$ of partial memberships and matrix $\mathbf{R}$ giving $\partial\phi_j/\partial\theta_\ell$.

$K \leftarrow \text{length}(\mathbf{d}),\ \ N_p \leftarrow \text{length}(\boldsymbol{\theta})$
$\boldsymbol{\phi} \leftarrow \mathbf{0}_K,\ \ \mathbf{R} \leftarrow \mathbf{0}_{K\times N_p}$
for $n = 1$ **to** N **do**
  draw $\boldsymbol{\lambda}$ **from** $\pi(\theta)$
  $k \leftarrow \text{argmin}_j\, d_j\lambda_j$
  $\phi_k \leftarrow \phi_k + 1$
  for $\ell = 1$ **to** N_p **do**

$$R_{k\ell} \leftarrow\ R_{k\ell} + \left[\frac{\partial \log \pi(\boldsymbol{\lambda}', \boldsymbol{\theta})}{\partial \theta_\ell}\right|_{\boldsymbol{\lambda}'=\boldsymbol{\lambda}}$$

  end
end
$\boldsymbol{\phi} \leftarrow \boldsymbol{\phi}/N,\ \ \mathbf{R} \leftarrow \mathbf{R}/N$
return $\boldsymbol{\phi}$, $\mathbf{R}$

prohibitive.

The simple Monte Carlo algorithm we propose can be applied to any equations for $\boldsymbol{\phi}$ of the form:

$$\phi_j = \int f_j(\boldsymbol{\lambda}, \mathbf{d})\pi(\boldsymbol{\lambda}|\theta)\mathrm{d}\boldsymbol{\lambda}. \tag{3.91}$$

In the case where we perturb the distance metric as described in section 3.1, we would have

$$f_j(\boldsymbol{\lambda}, \mathbf{d}) = \mathbb{I}[\lambda_j d_j \leq \lambda_\ell d_\ell\ \ \forall\, \ell \neq j] \tag{3.92}$$

Fortunately, evaluating equation (3.91) using Monte Carlo is a simple application of importance sampling. Because $\sum_j \phi_j = 1$, we sidestep the issue that importance sampling can only yield unnormalized answers simply by normalizing $\boldsymbol{\phi}$ at the end. Thus, to approximate ϕ, we repeatedly:

1. Sample a vector of K parameters $\tilde{\boldsymbol{\lambda}}$ from the prior.
2. Evaluate $\left[f_1\left(\tilde{\boldsymbol{\lambda}}\right), f_2\left(\tilde{\boldsymbol{\lambda}}\right), ..., f_K\left(\tilde{\boldsymbol{\lambda}}\right)\right]$.
3. Add it to the vector $\boldsymbol{\phi}$.
4. Repeat steps 1 to 3 N times.
5. Normalize $\boldsymbol{\phi}$.

Once we have repeated this N times, we can normalize $\boldsymbol{\phi}$ to get the final answer. Alternately, we can use the assumption that $\sum_j f_j(\boldsymbol{\lambda}, \mathbf{d}) = 1$ and simply divide the entries of $\boldsymbol{\phi}$ by N.

For the optimization step, we can calculate the derivative of $\boldsymbol{\phi}$ with respect to one of the hyperparameters of the prior – something advantageous when optimizing for the best hyperparameters – using the same technique on a similar expression:

$$\frac{\partial \phi_j(\mathbf{d}, \boldsymbol{\theta})}{\partial \theta_\ell} = \int f(\boldsymbol{\lambda}, \mathbf{d}) \frac{\partial}{\partial \theta_\ell} \pi(\boldsymbol{\lambda}, \boldsymbol{\theta}) \mathrm{d}\boldsymbol{\lambda} \tag{3.93}$$

$$= \int f(\boldsymbol{\lambda}, \mathbf{d}) \left[\frac{\partial \log \pi(\boldsymbol{\lambda}, \boldsymbol{\theta})}{\partial \theta_\ell}\right] \pi(\boldsymbol{\lambda}, \boldsymbol{\theta}) \mathrm{d}\boldsymbol{\lambda} \tag{3.94}$$

We can thus sample from $\pi(\boldsymbol{\lambda})$ and weight the sample by $f(\boldsymbol{\lambda}, \mathbf{d}) \left[\frac{\partial \log \pi(\boldsymbol{\lambda}, \boldsymbol{\theta})}{\partial \theta_\ell}\right]$, again taking the average of N samples to get our estimate of the derivative. This approach motivates the general Monte Carlo algorithm for calculating the distance metrics outlined in algorithm (4). In algorithm (5), we apply this using equation (3.92) for f, giving a slightly more efficient algorithm.

3.6 Approximation Algorithms

In many practical clustering validation problems, especially in dealing with a large number of clusters, a number of terms in the partial membership vector will probably be negligibly small. Being able to exclude these irrelevant clusters from consideration beforehand may vastly improve computation speed

and numerical accuracy, especially when using the shifted Gamma prior because of its $\mathcal{O}\big(nK^3\big)$ running time. We present here an accurate and easily computed bound that can be used exclude distances whose resulting value in $\boldsymbol{\phi}$ will negligibly small.

The central idea, presented formally in lemma 3.6.1, is to use the fact that the priors discussed above have support only for $\lambda \geq 1$ to lower bound the value of $\boldsymbol{\phi}_j$ with the integral over the region where $d_j\lambda_j$ has nonzero probability but $d_\ell\lambda_\ell$ does not. After presenting the formal idea in the following lemma, we apply it to the location exponential prior and the shifted Gamma prior.

Lemma 3.6.1. *Suppose, for $\ell = 1, 2, ..., K$, that $\pi_\ell(\lambda_\ell) = 0 \ \forall\, \lambda_\ell < 1$. Let $S \subset \{1, ..., K\}$ and $j \notin S$, and let*

$$\kappa = \underset{k \in S}{\operatorname{argmin}}\, d_k. \tag{3.95}$$

Then

$$\sum_{k \in S} \phi_k^{\mathrm{IE}} \leq \int_{d_\kappa/d_j}^{\infty} \pi_j(\lambda_j)\mathrm{d}\lambda_j \tag{3.96}$$

Proof. Now $S^c = \{1, ..., K\} - S$. Then, for each $k \in S$,

$$\phi_k^{\text{IE}} = \int \pi_k(\lambda_k) \prod_{\ell \neq k} \pi_\ell(\lambda_\ell) \mathbf{1}_{\{d_k\lambda_k < d_\ell\lambda_\ell\}} \, \mathrm{d}\lambda_\ell \mathrm{d}\lambda_k \tag{3.97}$$

$$= \int \pi_k(\lambda_k)\pi_j(\lambda_j) \mathbf{1}_{\{d_k\lambda_k < d_j\lambda_j\}} \, \mathrm{d}\lambda_j \prod_{\ell \in S\setminus k} \pi_\ell(\lambda_\ell) \mathbf{1}_{\{d_k\lambda_k < d_\ell\lambda_\ell\}} \, \mathrm{d}\lambda_\ell$$
$$\times \int \prod_{m \in S^c \setminus j} \pi_m(\lambda_m) \mathbf{1}_{\{d_k\lambda_k < d_m\lambda_m\}} \, \mathrm{d}\lambda_m \mathrm{d}\lambda_k \tag{3.98}$$

$$\leq \int \pi_k(\lambda_k)\pi_j(\lambda_j) \mathbf{1}_{\{d_k\lambda_k < d_j\lambda_j\}} \, \mathrm{d}\lambda_j \prod_{\ell \in S\setminus k} \pi_\ell(\lambda_\ell) \mathbf{1}_{\{d_k\lambda_k < d_\ell\lambda_\ell\}} \, \mathrm{d}\lambda_\ell$$
$$\times \int \prod_{m \in S^c \setminus j} \pi_m(\lambda_m) \mathrm{d}\lambda_m \mathrm{d}\lambda_k \tag{3.99}$$

$$= \int \pi_k(\lambda_k)\pi_j(\lambda_j) \mathbf{1}_{\{d_k\lambda_k < d_j\lambda_j\}} \, \mathrm{d}\lambda_j \int \prod_{\ell \in S\setminus k} \pi_\ell(\lambda_\ell) \mathbf{1}_{\{d_k\lambda_k < d_\ell\lambda_\ell\}} \, \mathrm{d}\lambda_\ell \mathrm{d}\lambda_k \tag{3.100}$$

where the inequality holds as integration over the removed indicator functions can only increase the area of integration and thus increase the area of integration. Now none of the priors has support on $(-\infty, 1)$, so $\min_{k' \in S} d_{k'} \leq \min_{k' \in S} d_{k'}\lambda_{k'} \leq d_k\lambda_k \;\; \forall\, k \in S$, so

$$\sum_{k \in S} \phi_k^{\text{IE}} \leq \sum_{k \in S} \int \pi_k(\lambda_k)\pi_j(\lambda_j) \mathbf{1}_{\{d_k\lambda_k < d_j\lambda_j\}} \, \mathrm{d}\lambda_j$$
$$\times \prod_{\ell \in S\setminus k} \mathbf{1}_{\{d_k\lambda_k < d_\ell\lambda_\ell\}} \, \pi_\ell(\lambda_\ell) \mathrm{d}\lambda_\ell \mathrm{d}\lambda_k \tag{3.101}$$

$$\leq \sum_{k \in S} \int \pi_k(\lambda_k)\pi_j(\lambda_j) \mathbb{I}[d_\kappa < d_j\lambda_j] \mathrm{d}\lambda_j$$
$$\times \int \prod_{\ell \in S\setminus k} \mathbf{1}_{\{d_k\lambda_k < d_\ell\lambda_\ell\}} \, \pi_\ell(\lambda_\ell) \mathrm{d}\lambda_\ell \mathrm{d}\lambda_k. \tag{3.102}$$

We can now separate the integral:

$$\sum_{k\in S}\phi_k^{\text{IE}} \leq \left[\int \pi_j(\lambda_j)\mathbb{I}[d_\kappa < d_j\lambda_j]\mathrm{d}\lambda_j\right] \times \left[\sum_{k\in S}\int_{\cap_{\ell\in S\setminus k}\{\boldsymbol{\lambda}_S\,:\,d_k\lambda_k<d_\ell\lambda_\ell\}}\prod_{\ell\in S}\pi_\ell(\lambda_\ell)\mathrm{d}\lambda_\ell\right] \tag{3.103}$$

where $\boldsymbol{\lambda}_S = (\lambda_{k_1}, \lambda_{k_2}, ..., \lambda_{k_{|S|}})$ for $S = \left\{k_1, k_2, ..., k_{|S|}\right\}$. The derivation is complete if the second term is ≤ 1, which we now show. We do by showing that the area of integration of each of the integrals under sum is disjoint from all the other integrals, giving us an upper bound of one for the sum. Formally $\forall\ k_1, k_2 \in S, k_1 \neq k_2$,

$$\left(\cap_{\ell\in S\setminus k_1}\left\{\boldsymbol{\lambda}_S : d_{k_1}\lambda_{k_1} < d_\ell\lambda_\ell\right\}\right) \cap \left(\cap_{\ell\in S\setminus k_2}\left\{\boldsymbol{\lambda}_S : d_{k_2}\lambda_{k_2} < d_\ell\lambda_\ell\right\}\right) \tag{3.104}$$

$$\subseteq \left\{\boldsymbol{\lambda}_S : d_{k_1}\lambda_{k_1} < d_{k_2}\lambda_{k_2}\right\} \cap \left\{\boldsymbol{\lambda}_S : d_{k_2}\lambda_{k_2} < d_{k_1}\lambda_{k_1}\right\} \tag{3.105}$$

$$= \emptyset \tag{3.106}$$

Because each area of integration is mutually disjoint, we can incorporate the sum over k directly into the integral as a union of the sets. Thus

$$\sum_{k\in S}\phi_k^{\text{IE}}(\mathbf{d}, \boldsymbol{\theta}) \leq \int \pi_j(\lambda_j)\mathbb{I}[d_\kappa < d_j\lambda_j]\mathrm{d}\lambda_j \times \int_{\cup_{k\in S}\cap_{\ell\in S\setminus k}\{\boldsymbol{\lambda}_S\,:\,d_k\lambda_k<d_\ell\lambda_\ell\}}\prod_{\ell\in S}\pi_\ell(\lambda_\ell)\mathrm{d}\lambda_\ell \tag{3.107}$$

$$\leq \int \pi_j(\lambda_j)\mathbb{I}[d_\kappa < d_j\lambda_j]\mathrm{d}\lambda_j \int \prod_{\ell\in S}\pi_\ell(\lambda_\ell)\mathrm{d}\lambda_\ell \tag{3.108}$$

$$= \int \pi_j(\lambda_j)\mathbb{I}[d_\kappa < d_j\lambda_j]\mathrm{d}\lambda_j \tag{3.109}$$

$$= \int_{d_\kappa/d_j}^{\infty} \pi_j(\lambda_j)\mathrm{d}\lambda_j, \tag{3.110}$$

which completes the proof. □

3.6.1 Error Bounds for the Location Exponential Prior

Lemma 3.6.1 allows us to easily upper bound the probability mass in a subset of the elements of ϕ when the corresponding distances are sufficiently large. In essence, we are arguing that the error from setting a sufficiently large distance to infinity, and thus ignoring it, is negligible.

Theorem 3.6.2. *Let* $\pi_\ell(\lambda_\ell) = \mathcal{LocExp}(1, \beta_\ell)$, $\ell = 1, 2, ..., K$, *and suppose* $\varepsilon > 0$. *Let* $j \in \{1, 2, ..., K\}$. *If*

$$S = \left\{ k : d_k \geq d_j \left(1 - \frac{\log \varepsilon}{\theta_j} \right) \right\}, \tag{3.111}$$

then

$$\sum_{k \in S} \phi_k^{\mathrm{LE}} \leq \varepsilon \tag{3.112}$$

Proof. Let

$$\kappa = \underset{k \in S}{\operatorname{argmin}}\, d_k. \tag{3.113}$$

From lemma 3.6.1 we have that

$$\sum_{k \in S} \phi_k^{\mathrm{LE}} \leq \int_{d_\kappa / d_j}^{\infty} \pi_j(\lambda_j) \mathrm{d}\lambda_j \tag{3.114}$$

Now

$$d_\kappa \geq d_j\left(1 - \frac{\log\varepsilon}{\theta_j}\right) \tag{3.115}$$

$$\Rightarrow \theta_j\left(\frac{d_\kappa}{d_j} - 1\right) \geq -\log\varepsilon \tag{3.116}$$

$$\Rightarrow \exp\left[-\theta_j\left(\frac{d_\kappa}{d_j} - 1\right)\right] \leq \varepsilon \tag{3.117}$$

$$\Rightarrow \int_{d_\kappa/d_j}^{\infty} \pi_j(\lambda_j)\mathrm{d}\lambda_j \leq \varepsilon \tag{3.118}$$

$$\Rightarrow \sum_{k\in S} \phi_k^{\mathrm{LE}} \leq \varepsilon \tag{3.119}$$

□

3.6.2 Error Bounds for the Shifted Gamma Prior

Theorem 3.6.3. *Let* $\pi_\ell(\lambda_\ell) = \mathcal{LocExp}(1, \beta_\ell)$, $\ell = 1, 2, ..., K$, *and suppose* $\varepsilon > 0$. *Let* $j \in \{1, 2, ..., K\}$. *If*

$$S = \left\{k : d_k \geq d_j F_{Ga}^{-1}(1-\varepsilon) + 1\right\}, \tag{3.120}$$

where F_{Ga}^{-1} *is the inverse cdf of the Gamma distribution, then*

$$\sum_{k\in S} \phi_k^{\mathrm{SG}} \leq \varepsilon \tag{3.121}$$

Proof. Let

$$\kappa = \operatorname*{argmin}_{k\in S} d_k. \tag{3.122}$$

Let F_{SG}^{-1} be the inverse cdf of the shifted Gamma distribution, defined in equation (3.64). Then

$$\frac{d_\kappa}{d_j} \geq F_{\mathrm{Ga}}^{-1}(1-\varepsilon) + 1 = F_{\mathrm{SG}}^{-1}(1-\varepsilon) \tag{3.123}$$

$$\Rightarrow 1 - F_{\mathrm{SG}}\frac{d_\kappa}{d_j} \leq \varepsilon \tag{3.124}$$

From lemma 3.6.1 we have that

$$\sum_{k\in S} \phi_k^{\mathrm{SG}} \leq \int_{d_\kappa/d_j}^{\infty} \pi_j(\lambda_j)\mathrm{d}\lambda_j \tag{3.125}$$

$$= 1 - F_{\mathrm{SG}}^{-1}\left(\frac{d_\kappa}{d_j}\right), \tag{3.126}$$

so the theorem is proved. □

These theorems allow us to maintain a guaranteed level of accuracy while, in some cases, asymptotically improving the running time. Many flavors of k-means for large scale clustering use data structures like kd-trees to avoid computing distances from points to irrelevant centroids. In such cases, the above bound allows us to worry only about the centroids within a specified radius – often called a range query – which kd-trees are easily able to handle.

3.7 Additional Properties of the Pointwise Stability

Sometimes it may be easier to directly calculate the pointwise stability instead of the entire partial membership. This can be particularly true if, for instance, the integration has to be done numerically using more expensive approximation methods such as those described in chapter 7. Using the theorems we present here, a good estimate of the pointwise stability can be obtained quite easily.

Additionally, and perhaps more importantly, this theorem can be used to prove other desirable properties of the pointwise stability measure. We

thus present an additional theorem using this one that shows the pointwise stability behaves as one would expect if one of the distance measures changes.

3.7.1 Differences Between two Pointwise Stability Terms

We derive results for the slightly more general case of the difference between any two points; the pointwise stability is then just a special case. Note that in practice it will probably easier to just evaluate this directly from the averaged assignement matrix; however, this version is useful as a tool to prove several theorems later on.

Theorem 3.7.1. *Suppose $d_j < d_k$. Then*

$$\psi_j(\mathbf{d},\theta) - \psi_k(\mathbf{d},\theta) = \int_{R_{jk}(\mathbf{d})} \pi(\boldsymbol{\lambda},\theta)\mathrm{d}\boldsymbol{\lambda} \tag{3.127}$$

where

$$R_{jk}(\mathbf{d}) = \left\{\boldsymbol{\lambda} : \frac{d_j}{d_k}\lambda_k \leq \lambda_j \leq \frac{d_\ell}{d_j}\lambda_\ell \ \forall\, \ell \neq j\right\} \tag{3.128}$$

Proof. We begin by combining the common terms in the integral:

$$\begin{aligned}\psi_j(\mathbf{d},\theta) - \psi_k(\mathbf{d},\theta) &= \int \pi(\boldsymbol{\lambda})\ \mathbb{I}\{d_j\lambda_j \leq d_\ell\lambda_\ell \ \forall\, \ell \neq j\}\mathrm{d}\boldsymbol{\lambda} \\ &\quad - \int \pi(\boldsymbol{\lambda})\ \mathbb{I}\{d_k\lambda_k \leq d_\ell\lambda_\ell \ \forall\, \ell \neq k\}\mathrm{d}\boldsymbol{\lambda} \qquad (3.129)\\ &= \int \pi(\boldsymbol{\lambda})\ \big[\mathbb{I}\{d_j\lambda_j \leq d_\ell\lambda_\ell \ \forall\, \ell \neq j\} \\ &\quad - \mathbb{I}\{d_k\lambda_k \leq d_\ell\lambda_\ell \ \forall\, \ell \neq k\}\big]\mathrm{d}\boldsymbol{\lambda} \qquad (3.130)\end{aligned}$$

To translate the indicator functions into sets, we define

$$S_{jk} = \{\boldsymbol{\lambda} : d_j\lambda_j \leq d_\ell\lambda_\ell \ \forall\, \ell \neq j,k\} \tag{3.131}$$

$$\Rightarrow S_{kj} = \{\boldsymbol{\lambda} : d_k\lambda_k \leq d_\ell\lambda_\ell \ \forall\, \ell \neq j,k\} \tag{3.132}$$

$$T_{jk} = \{\boldsymbol{\lambda} : d_j\lambda_j \leq d_k\lambda_k\} \tag{3.133}$$

$$\Rightarrow T_{kj} = \{\boldsymbol{\lambda} : d_k\lambda_k \leq d_j\lambda_j\} \tag{3.134}$$

We can express the indicator functions in equation (3.130) using these sets:

$$\psi_j(\mathbf{d},\theta) - \psi_k(\mathbf{d},\theta) = \int \pi(\boldsymbol{\lambda}) \left(\mathbf{1}_{S_{jk}\cap T_{jk}} - \mathbf{1}_{S_{kj}\cap T_{kj}}\right)\mathrm{d}\boldsymbol{\lambda} \tag{3.135}$$

Our intent now is to isolate the regions of $S_{jk} \cap T_{jk}$ and $S_{kj} \cap T_{kj}$ where the integral will return the same answer. We begin by breaking $S_{jk} \cap T_{jk}$ apart as follows. Recall that $d_k \geq d_j$, so we can split T, which upper bounds λ_j at d_ℓ/d_j, along λ_j by breaking it at d_ℓ/d_k:

$$S_{jk} \cap T_{jk} = \left\{\boldsymbol{\lambda} : \lambda_j \leq \frac{d_\ell}{d_j}\lambda_\ell \ \forall\, \ell \neq j,k\right\} \cap \left\{\boldsymbol{\lambda} : \lambda_j \leq \frac{d_k}{d_j}\lambda_k\right\} \tag{3.136}$$

$$= \left[\left\{\lambda_j \leq \frac{d_\ell}{d_k}\lambda_\ell \ \forall\, \ell \neq j,k\right\} \cup \left\{\frac{d_\ell}{d_k}\lambda_\ell \leq \lambda_j \leq \frac{d_\ell}{d_j}\lambda_\ell \ \forall\, \ell \neq j,k\right\}\right] \cap \left\{\lambda_j \leq \frac{d_k}{d_j}\lambda_k\right\} \tag{3.137}$$

$$= \left[\left\{\lambda_j \leq \frac{d_\ell}{d_k}\lambda_\ell \ \forall\, \ell \neq j,k\right\} \cap \left\{\lambda_j \leq \frac{d_k}{d_j}\lambda_k\right\}\right] \cup \left[\left\{\frac{d_\ell}{d_k}\lambda_\ell \leq \lambda_j \leq \frac{d_\ell}{d_j}\lambda_\ell \ \forall\, \ell \neq j,k\right\} \cap \left\{\lambda_j \leq \frac{d_k}{d_j}\lambda_k\right\}\right]. \tag{3.138}$$

We split the second set using the same technique:

$$S_{jk} \cap T_{jk} = \left[\left\{ \lambda_j \leq \frac{d_\ell}{d_k} \lambda_\ell \ \forall \, \ell \neq j,k \right\} \right.$$
$$\left. \cap \left(\left\{ \lambda_j \leq \frac{d_j}{d_k} \lambda_k \right\} \cup \left\{ \frac{d_j}{d_k} \lambda_k \leq \lambda_j \leq \frac{d_k}{d_j} \lambda_k \right\} \right) \right]$$
$$\cup \left[\left\{ \frac{d_\ell}{d_k} \lambda_\ell \leq \lambda_j \leq \frac{d_\ell}{d_j} \lambda_\ell \ \forall \, \ell \neq j,k \right\} \cap \left\{ \lambda_j \leq \frac{d_k}{d_j} \lambda_k \right\} \right] \tag{3.139}$$

$$= \left[\left\{ \lambda_j \leq \frac{d_\ell}{d_k} \lambda_\ell \ \forall \, \ell \neq j,k \right\} \cap \left\{ \lambda_j \leq \frac{d_j}{d_k} \lambda_k \right\} \right]$$
$$\cup \left[\left\{ \lambda_j \leq \frac{d_\ell}{d_k} \lambda_\ell \ \forall \, \ell \neq j,k \right\} \cap \left\{ \frac{d_j}{d_k} \lambda_k \leq \lambda_j \leq \frac{d_k}{d_j} \lambda_k \right\} \right]$$
$$\cup \left[\left\{ \frac{d_\ell}{d_k} \lambda_\ell \leq \lambda_j \leq \frac{d_\ell}{d_j} \lambda_\ell \ \forall \, \ell \neq j,k \right\} \cap \left\{ \lambda_j \leq \frac{d_k}{d_j} \lambda_k \right\} \right]. \tag{3.140}$$

We can simplify and combine the second and third line:

$$S_{jk} \cap T_{jk} = \left[\left\{ \lambda_j \leq \frac{d_\ell}{d_k} \lambda_\ell \ \forall \, \ell \neq j,k \right\} \cap \left\{ \lambda_j \leq \frac{d_j}{d_k} \lambda_k \right\} \right]$$
$$\cup \left[\left(\left\{ \frac{d_j}{d_k} \lambda_k \leq \lambda_j \leq \frac{d_\ell}{d_k} \lambda_\ell \ \forall \, \ell \neq j,k \right\} \right.\right.$$
$$\left.\left. \cup \left\{ \frac{d_\ell}{d_k} \lambda_\ell \leq \lambda_j \leq \frac{d_\ell}{d_j} \lambda_\ell \ \forall \, \ell \neq j,k \right\} \right) \right.$$
$$\left. \cap \left\{ \lambda_j \leq \frac{d_k}{d_j} \lambda_k \right\} \right] \tag{3.141}$$

$$= \left[\left\{ \lambda_j \leq \frac{d_\ell}{d_k} \lambda_\ell \ \forall \, \ell \neq j,k \right\} \cap \left\{ \lambda_j \leq \frac{d_j}{d_k} \lambda_k \right\} \right]$$
$$\cup \left\{ \frac{d_j}{d_k} \lambda_k \leq \lambda_j \leq \min \left(\frac{d_\ell}{d_j} \lambda_\ell, \frac{d_k}{d_j} \lambda_k \right) \ \forall \, \ell \neq j,k \right\} \tag{3.142}$$

$$= \left[\left\{ \lambda_j \leq \frac{d_\ell}{d_k} \lambda_\ell \ \forall \, \ell \neq j,k \right\} \cap \left\{ \lambda_j \leq \frac{d_j}{d_k} \lambda_k \right\} \right]$$
$$\cup \left\{ \frac{d_j}{d_k} \lambda_k \leq \lambda_j \leq \frac{d_\ell}{d_j} \lambda_\ell \ \forall \, \ell \neq j \right\}. \tag{3.143}$$

The lemma follows directly from observing that

$$\int_{\left\{\lambda_j \le \frac{d_\ell}{d_k}\lambda_\ell \ \forall \ell \neq j,k\right\} \cap \left\{\lambda_j \le \frac{d_j}{d_k}\lambda_k\right\}} \pi(\boldsymbol{\lambda}) \mathrm{d}\boldsymbol{\lambda} = \int_{\left\{\lambda_k \le \frac{d_\ell}{d_k}\lambda_\ell \ \forall \ell \neq j,k\right\} \cap \left\{\lambda_k \le \frac{d_j}{d_k}\lambda_j\right\}} \pi(\boldsymbol{\lambda}) \mathrm{d}\boldsymbol{\lambda} \quad (3.144)$$

as the variables of integration λ_j and λ_k can be switched without affecting the value of the integral. Thus we have that

$$\int \pi(\boldsymbol{\lambda}) \left(\mathbf{1}_{S_{jk} \cap T_{jk}} - \mathbf{1}_{S_{kj} \cap T_{kj}}\right) \mathrm{d}\boldsymbol{\lambda} \quad (3.145)$$

$$= \int_{R(\mathbf{d})} \pi(\boldsymbol{\lambda}) \mathrm{d}\boldsymbol{\lambda} + \int_{\{(\cdots, \lambda_j, \cdots, \lambda_k, \cdots) : (\cdots, \lambda_k, \cdots, \lambda_j, \cdots) \in S_{kj} \cap T_{kj}\}} \pi(\boldsymbol{\lambda}) \mathrm{d}\boldsymbol{\lambda} - \int_{S_{kj} \cap T_{kj}} \pi(\boldsymbol{\lambda}) \mathrm{d}\boldsymbol{\lambda} \quad (3.146)$$

$$= \int_{R(\mathbf{d})} \pi(\boldsymbol{\lambda}) \mathrm{d}\boldsymbol{\lambda}, \quad (3.147)$$

which completes the proof. □

The result of theorem 3.7.1 is most useful if it can apply to arbitrary ϕ_j and ϕ_k, without the restriction that $d_j \le d_k$. To cover this case, we present the following corollary.

Corollary 3.7.2. *Suppose $d_j \ge d_k$. Then*

$$\psi_j(\mathbf{d}, \theta) - \psi_k(\mathbf{d}, \theta) = -\int_{R_{kj}(\mathbf{d})} \pi(\boldsymbol{\lambda}, \theta) \mathrm{d}\boldsymbol{\lambda} \quad (3.148)$$

where

$$R_{kj}(\mathbf{d}) = \left\{\boldsymbol{\lambda} : \frac{d_k}{d_j}\lambda_j \le \lambda_k \le \frac{d_\ell}{d_k}\lambda_\ell \ \forall \ell \neq k\right\} \quad (3.149)$$

Proof. Trivially, note that

$$\psi_j(\mathbf{d},\theta) - \psi_k(\mathbf{d},\theta) = -\left[\psi_k(\mathbf{d},\theta) - \psi_j(\mathbf{d},\theta)\right]. \tag{3.150}$$

Thus we can apply the result of theorem 3.7.1, taking the negative of the result to get the corollary. □

3.7.2 Behavior of the Pointwise Stability

In this section, we investigate how the pointwise stability changes given changes in the distances between points and centroids. These proofs, in essence, are a sanity check that what it measures is sensible. In particular, we prove that if the distance to centroid j decreases while all the other distances remain fixed, then the difference between ϕ_j and any of the other $\phi_\cdot$'s increases or remains constant. This is what we would expect, given that such a move should only increase the stability of a point. Similarly, if the distance to another centroid k, $k \neq j$, is decreased, than $\phi_j - \phi_\ell$ decreases or stays the same for $\ell \neq j$. We present these formally in the following theorems.

Theorem 3.7.3. *Let* $\mathbf{d} = (d_1, d_2, ..., d_K)$ *be a vector of centroid-to-point distances, and let* ψ *be any stability measure based on scaled distance perturbations. Let* j, k *be any index in* $\{1, 2, ..., K\}$ *such that* $d_j < d_k$*. Suppose* $0 < \varepsilon < d_j$*, and let*

$$\mathbf{d}' = (d_1, ..., d_j - \varepsilon, ..., d_K). \tag{3.151}$$

Then

$$\psi_j(\mathbf{d}',\theta) - \psi_k(\mathbf{d}',\theta) \geq \psi_j(\mathbf{d},\theta) - \psi_k(\mathbf{d},\theta) \tag{3.152}$$

Proof. The proof follows directly from theorem 3.7.1. By theorem 3.7.1, we have that

$$\left[\psi_j(\mathbf{d}',\theta) - \psi_k(\mathbf{d}',\theta)\right] - \left[\psi_j(\mathbf{d},\theta) - \psi_k(\mathbf{d},\theta)\right]$$
$$= \int_{R_{jk}(\mathbf{d}')} \pi(\boldsymbol{\lambda}|\theta)\mathrm{d}\boldsymbol{\lambda} - \int_{R_{jk}(\mathbf{d})} \pi(\boldsymbol{\lambda}|\theta)\mathrm{d}\boldsymbol{\lambda} \quad (3.153)$$

where

$$R_{jk}(\mathbf{d}) = \left\{\boldsymbol{\lambda} : \frac{d_j}{d_k}\lambda_k \leq \lambda_j \leq \frac{d_\ell}{d_j}\lambda_\ell \ \forall\, \ell \neq j\right\} \quad (3.154)$$

$$R_{jk}(\mathbf{d}') = \left\{\boldsymbol{\lambda} : \frac{d_j - \varepsilon}{d_k}\lambda_k \leq \lambda_j \leq \frac{d_\ell}{d_j - \varepsilon}\lambda_\ell \ \forall\, \ell \neq j\right\} \quad (3.155)$$

Now $d_k > 0$, so

$$\frac{d_j - \varepsilon}{d_k} < \frac{d_j}{d_k}. \quad (3.156)$$

Similarly, $d_\ell > 0$ so

$$\frac{d_\ell}{d_j - \varepsilon} > \frac{d_\ell}{d_j} \quad (3.157)$$

Thus

$$R_{jk}(\mathbf{d}) \subset R_{jk}(\mathbf{d}') \quad (3.158)$$

This allows us to combine the two integrals in equation (3.153):

$$\left[\psi_j(\mathbf{d}',\theta) - \psi_k(\mathbf{d}',\theta)\right] - \left[\psi_j(\mathbf{d},\theta) - \psi_k(\mathbf{d},\theta)\right] = \int_{R_{jk}(\mathbf{d}') - R_{jk}(\mathbf{d})} \pi(\boldsymbol{\lambda}|\theta)\mathrm{d}\boldsymbol{\lambda}. \quad (3.159)$$

However, $\pi(\boldsymbol{\lambda}|\theta) \geq 0 \ \forall\, \boldsymbol{\lambda}$, so the value of the integral is nonnegative. Thus

$$\left[\psi_j(\mathbf{d}',\theta) - \psi_k(\mathbf{d}',\theta)\right] - \left[\psi_j(\mathbf{d},\theta) - \psi_k(\mathbf{d},\theta)\right] \geq 0, \quad (3.160)$$

proving the theorem. □

This method of proof can be repeated to get additional inequalities that cover all the possible cases where a single distance metric changes. Together, this gives a fairly solid sanity check that the pointwise stability is not going to do anything too surprising.

Note also that the converse of all of these theorems apply as well for sufficiently small ε; while we discuss specifically the cases for shrinking a distance metric by a value ε, the reverse inequality holds for the case of increasing the distance metric as well. For the above theorem, we present this as another corollary.

Corollary 3.7.4. *Let $\mathbf{d} = (d_1, d_2, ..., d_K)$ be a vector of centroid-to-point distances, and let ψ be any stability measure based on scaled distance perturbations. Let j, k be any index in $\{1, 2, ..., K\}$ such that $d_j < d_k$. Suppose $d_j < \varepsilon < d_k$, and let*

$$\mathbf{d}' = (d_1, ..., d_j + \varepsilon, ..., d_K). \tag{3.161}$$

Then

$$\psi_j(\mathbf{d}', \theta) - \psi_k(\mathbf{d}', \theta) \leq \psi_j(\mathbf{d}, \theta) - \psi_k(\mathbf{d}, \theta) \tag{3.162}$$

Proof. This theorem is simply a restatement of theorem 3.7.3. If we swap the prime notation on the distance vectors and assume we were given $d_j + \varepsilon$ instead of d_j, we have theorem 3.7.3 exactly. The proof of that theorem is sufficient. □

We can now continue with the other 2 cases, changing k and changing $\ell \neq j, k$.

Theorem 3.7.5. *Let $\mathbf{d} = (d_1, d_2, ..., d_K)$ be a vector of centroid-to-point distances, and let ψ be any stability measure based on scaled distance perturbations. Let j, k be any index in $\{1, 2, ..., K\}$ such that $d_j < d_k$. Suppose $d_j < \varepsilon < d_k$, and let*

$$\mathbf{d}' = (d_1, ..., d_k - \varepsilon, ..., d_K). \tag{3.163}$$

Then

$$\psi_j(\mathbf{d}', \theta) - \psi_k(\mathbf{d}', \theta) \leq \psi_j(\mathbf{d}, \theta) - \psi_k(\mathbf{d}, \theta) \tag{3.164}$$

Proof. Again, the proof follows directly from theorem 3.7.1. By theorem 3.7.1, we have that

$$\begin{aligned}[\psi_j(\mathbf{d}, \theta) - \psi_k(\mathbf{d}, \theta)] - \left[\psi_j(\mathbf{d}', \theta) - \psi_k(\mathbf{d}', \theta)\right] \\ = \int_{R_{jk}(\mathbf{d}')} \pi(\boldsymbol{\lambda}|\theta) \mathrm{d}\boldsymbol{\lambda} - \int_{R_{jk}(\mathbf{d})} \pi(\boldsymbol{\lambda}|\theta) \mathrm{d}\boldsymbol{\lambda}\end{aligned} \tag{3.165}$$

where

$$R_{jk}(\mathbf{d}) = \left\{\boldsymbol{\lambda} : \frac{d_j}{d_k}\lambda_k \leq \lambda_j \leq \frac{d_\ell}{d_j}\lambda_\ell \ \forall\, \ell \neq j\right\} \tag{3.166}$$

$$R_{jk}(\mathbf{d}') = \left\{\boldsymbol{\lambda} : \frac{d_j}{d_k - \varepsilon}\lambda_k \leq \lambda_j \leq \frac{d_\ell}{d_j}\lambda_\ell \ \forall\, \ell \neq j\right\} \tag{3.167}$$

Now $d_k > 0$, so

$$\frac{d_j}{d_k - \varepsilon} < \frac{d_j}{d_k}. \tag{3.168}$$

Thus

$$R_{jk}(\mathbf{d}') \subset R_{jk}(\mathbf{d}) \tag{3.169}$$

This allows us to combine the two integrals in equation (3.165):

$$[\psi_j(\mathbf{d}, \theta) - \psi_k(\mathbf{d}, \theta)] - \left[\psi_j(\mathbf{d}', \theta) - \psi_k(\mathbf{d}', \theta)\right] = \int_{R_{jk}(\mathbf{d}) - R_{jk}(\mathbf{d}')} \pi(\boldsymbol{\lambda}|\theta) \mathrm{d}\boldsymbol{\lambda}. \tag{3.170}$$

However, $\pi(\boldsymbol{\lambda}|\theta) \geq 0 \ \forall\, \boldsymbol{\lambda}$, so the value of the integral is nonnegative. Thus

$$[\psi_j(\mathbf{d}, \theta) - \psi_k(\mathbf{d}, \theta)] - \left[\psi_j(\mathbf{d}', \theta) - \psi_k(\mathbf{d}', \theta)\right] \geq 0, \tag{3.171}$$

proving the theorem. □

Corollary 3.7.6. *Let* $\mathbf{d} = (d_1, d_2, ..., d_K)$ *be a vector of centroid-to-point distances, and let* ψ *be any stability measure based on scaled distance perturbations. Let* j, k *be any index in* $\{1, 2, ..., K\}$ *such that* $d_j < d_k$*. Suppose* $0 < \varepsilon$*, and let*

$$\mathbf{d}' = (d_1, ..., d_k + \varepsilon, ..., d_K). \tag{3.172}$$

Then

$$\psi_j(\mathbf{d}', \theta) - \psi_k(\mathbf{d}', \theta) \geq \psi_j(\mathbf{d}, \theta) - \psi_k(\mathbf{d}, \theta) \tag{3.173}$$

Proof. Again, this theorem is simply a restatement of theorem 3.7.5. If we swap the prime notation on the distance vectors and assume we were given $d_k + \varepsilon$ instead of d_k, we have theorem 3.7.5 exactly. The proof of that theorem is sufficient. □

Theorem 3.7.7. *Let* $\mathbf{d} = (d_1, d_2, ..., d_K)$ *be a vector of centroid-to-point distances, and let* ψ *be any stability measure based on scaled distance perturbations. Let* j, k *be any index in* $\{1, 2, ..., K\}$ *such that* $d_j < d_k$*, and let* m *be any index other than* j *or* k*. Suppose* $0 < \varepsilon < d_m$*, and let*

$$\mathbf{d}' = (d_1, ..., d_m - \varepsilon, ..., d_K). \tag{3.174}$$

Then

$$\psi_j(\mathbf{d}', \theta) - \psi_k(\mathbf{d}', \theta) \leq \psi_j(\mathbf{d}, \theta) - \psi_k(\mathbf{d}, \theta) \tag{3.175}$$

Proof. Again, the proof follows directly from theorem 3.7.1. By theorem 3.7.1, we have that

$$\begin{aligned}[\psi_j(\mathbf{d}, \theta) - \psi_k(\mathbf{d}, \theta)] - \left[\psi_j(\mathbf{d}', \theta) - \psi_k(\mathbf{d}', \theta)\right] \\ = \int_{R_{jk}(\mathbf{d}')} \pi(\boldsymbol{\lambda}|\theta) \mathrm{d}\boldsymbol{\lambda} - \int_{R_{jk}(\mathbf{d})} \pi(\boldsymbol{\lambda}|\theta) \mathrm{d}\boldsymbol{\lambda}\end{aligned} \tag{3.176}$$

where

$$R_{jk}(\mathbf{d}) = \left\{ \boldsymbol{\lambda} : \frac{d_j}{d_k}\lambda_k \leq \lambda_j \leq \frac{d_\ell}{d_j}\lambda_\ell \ \forall\, \ell \neq j, m \text{ and } \lambda_j \leq \frac{d_m}{d_j}\lambda_m \right\} \quad (3.177)$$

$$R_{jk}(\mathbf{d}') = \left\{ \boldsymbol{\lambda} : \frac{d_j}{d_k}\lambda_k \leq \lambda_j \leq \frac{d_\ell}{d_j}\lambda_\ell \ \forall\, \ell \neq j, m \text{ and } \lambda_j \leq \frac{d_m - \varepsilon}{d_j}\lambda_m \right\} \quad (3.178)$$

Note that we have explicitly highlighted the terms with m. Now $d_j, d_m > 0$, so

$$\frac{d_m - \varepsilon}{d_j} < \frac{d_m}{d_j}. \quad (3.179)$$

Thus

$$R_{jk}(\mathbf{d}') \subseteq R_{jk}(\mathbf{d}) \quad (3.180)$$

This allows us to combine the two integrals in equation (3.176):

$$[\psi_j(\mathbf{d}, \theta) - \psi_k(\mathbf{d}, \theta)] - \left[\psi_j(\mathbf{d}', \theta) - \psi_k(\mathbf{d}', \theta)\right] = \int_{R_{jk}(\mathbf{d}) - R_{jk}(\mathbf{d}')} \pi(\boldsymbol{\lambda}|\theta) \mathrm{d}\boldsymbol{\lambda}. \quad (3.181)$$

However, $\pi(\boldsymbol{\lambda}|\theta) \geq 0 \ \forall\, \boldsymbol{\lambda}$, so the value of the integral is nonnegative. Thus

$$[\psi_j(\mathbf{d}, \theta) - \psi_k(\mathbf{d}, \theta)] - \left[\psi_j(\mathbf{d}', \theta) - \psi_k(\mathbf{d}', \theta)\right] \geq 0, \quad (3.182)$$

proving the theorem. □

Corollary 3.7.8. *Let* $\mathbf{d} = (d_1, d_2, ..., d_K)$ *be a vector of centroid-to-point distances, and let* ψ *be any stability measure based on scaled distance perturbations. Let* j, k *be any index in* $\{1, 2, ..., K\}$ *such that* $d_j < d_k$*, and let* m *be any index other than* j *or* k*. Suppose* $0 < \varepsilon$*, and let*

$$\mathbf{d}' = (d_1, ..., d_m + \varepsilon, ..., d_K). \quad (3.183)$$

Then

$$\psi_j(\mathbf{d}', \theta) - \psi_k(\mathbf{d}', \theta) \geq \psi_j(\mathbf{d}, \theta) - \psi_k(\mathbf{d}, \theta) \tag{3.184}$$

Proof. Again, this theorem is simply a restatement of theorem 3.7.7. If we swap the prime notation on the distance vectors and assume we were given $d_m + \varepsilon$ instead of d_m, we have theorem 3.7.7 exactly. The proof of that theorem is sufficient. □

3.8 Extensions to Other Indices

We wrap up this chapter by tying up some loose connections to chapter 2. As mentioned in section 2.2.3, we can naturally incorporate other stability indices into our method. We continue this extension in this section by deriving the gradient of each of these indices with respect to the prior parameters. This allows us to efficiently tune the priors using these methods as well should we desire to use them.

3.8.1 Optimizing $\mathcal{AR}^\star$ and $\mathcal{VI}^\star$ over $\boldsymbol{\theta}$

In finding the $\boldsymbol{\theta}$ that maximizes $\mathcal{AR}^\star$ (or $\mathcal{VI}^\star$), or a linear combination thereof, it speeds the computation up immensely if we know the gradient of $\mathcal{AR}^\star$ ($\mathcal{VI}^\star$) with respect to $\boldsymbol{\theta}$. Knowing this allows us to use much more sophisticated and powerful optimizers. In this section, we calculate the gradient of $\mathcal{AR}^\star$ and $\mathcal{VI}^\star$. Ultimately, both of these depend on first calculating $\partial \phi_j^{\mathrm{IE}} / \partial \theta_\ell \Big|_{\theta_\ell = \theta'_\ell}$ for each row of ϕ; we derive an algorithm in the next section to do this efficiently in amortized constant time per j, ℓ pair.

For the Hubert-Arabie Adjusted Rand Index,

$$\nabla_{\boldsymbol{\theta}}\, \mathcal{AR}^\star = \nabla_{\boldsymbol{\theta}}\, \frac{\sum_j \sum_\ell p_{j\ell}^2 - \left(\sum_j p_j^2\right)\left(\sum_\ell {p_\ell^\star}^2\right)}{\frac{1}{2}\left[\left(\sum_j p_j^2\right) + \left(\sum_\ell {p_\ell^\star}^2\right)\right] - \left(\sum_j {p_j}^2\right)\left(\sum_\ell {p_\ell^\star}^2\right)} \tag{3.185}$$

$$= \nabla_{\boldsymbol{\theta}}\, \frac{A - BB^\star}{\frac{1}{2}(B + B^\star) - BB^\star} \tag{3.186}$$

where, to make the calculation easier, we've defined the following terms:

$$A = \sum\nolimits_j \sum\nolimits_\ell p_{j\ell}^2 \tag{3.187}$$

$$B = \sum\nolimits_j p_j^2 \tag{3.188}$$

$$B^\star = \sum\nolimits_\ell {p_\ell^\star}^2 \tag{3.189}$$

Continuing:

$$\nabla_{\boldsymbol{\theta}}\ \mathcal{AR}^\star = \frac{1}{\frac{1}{2}(B + B^\star) - BB^\star}\left[\nabla_{\boldsymbol{\theta}}\ A - \left(B + \left(\frac{1}{2} - B\right)\mathcal{AR}^\star\right)\nabla_{\boldsymbol{\theta}}\ B^\star\right] \tag{3.190}$$

Now

$$\nabla_{\boldsymbol{\theta}}\ A = \sum_j \sum_\ell \nabla_{\boldsymbol{\theta}}\ p_{j\ell}^2 = \frac{1}{n}\sum_j \sum_\ell 2p_{j\ell}\nabla_{\boldsymbol{\theta}}\ p_{j\ell} \tag{3.191}$$

$$\nabla_{\boldsymbol{\theta}}\ B^\star = \sum_\ell \nabla_{\boldsymbol{\theta}}\ {p_\ell^\star}^2 = \frac{1}{n}\sum_\ell 2p_\ell^\star \nabla_{\boldsymbol{\theta}}\ p_\ell^\star \tag{3.192}$$

where

$$\nabla_{\boldsymbol{\theta}}\ p_{j\ell} = \sum_{i:a_{ij}=1} \nabla_{\boldsymbol{\theta}}\ \phi_{i\ell} \tag{3.193}$$

$$\nabla_{\boldsymbol{\theta}}\ p_\ell^\star = \sum_i \nabla_{\boldsymbol{\theta}}\ \phi_{i\ell} \tag{3.194}$$

Thus we can calculate $\nabla_{\boldsymbol{\theta}}\ \mathcal{AR}^\star$ given the partial derivatives of ϕ_j with respect to the prior parameters θ_ℓ.

The Variation of Information is similar:

$$\nabla_{\boldsymbol{\theta}}\ \mathcal{VI}^{\star} = \nabla_{\boldsymbol{\theta}}\ \left[-\sum_{j} p_j \log p_j - \sum_{\ell} p^{\star}_{\ell} \log p^{\star}_{\ell} - 2 \sum_{j} \sum_{\ell} p_{j\ell} \log \frac{p_{j\ell}}{p_j p^{\star}_{\ell}} \right] \tag{3.195}$$

$$= -\sum_{\ell} \left(\log p^{\star}_{\ell} + 1 + \frac{K}{p^{\star}_{\ell}} \right) \nabla_{\boldsymbol{\theta}}\ p^{\star}_{\ell} - 2 \sum_{j} \left(\log \frac{p_{j\ell}}{p_j p^{\star}_{\ell}} + 1 \right) \nabla_{\boldsymbol{\theta}}\ p_{j\ell} \tag{3.196}$$

Again, using equations (3.193) and (3.194), this can be calculated easily given $\partial \phi_j^{\text{IE}} / \partial \theta_\ell$.

Chapter 4

Synthetic Data for Cluster Validation Tests

Recall from section 1.2.1 that the canonical way to compare cluster validation procedures is to test how well they predict the true number of clusters in a known dataset. Such an analysis, however, has limits. One is that it depends on the type of data. Another is that it depends on the quality of data. Issues such as missing data, truncated features, and contaminations points, and the methods used to control them, can have a significant influence on the results. We thus propose a method to control these issues and abstract them, as much as possible, from the process of testing clustering validation methods.

Many of the papers proposing or describing cluster validation methods only compare them on a handful of datasets. This is largely due to two factors. First, there are few robust methods for generating "difficult" synthetic data with a known number of clusters, and many of these do not work in high dimensions. Second, most real datasets are for classification purposes; while sometimes people ignore the labels and compare methods based on how well they recreate the known partitions, most of these datasets do not have classes that separate well into clusters. We thus felt a method to generate non-Gaussian, "difficult" datasets with a known number of clusters would be a valuable and useful tool for the clustering community.

A reasonable collection of synthetic datasets used to compare the accuracy of clustering validation techniques should meet several requirements. First, each dataset should have K_{true} distinct clusters. Second, it should contain datasets with a variety of types – i.e. cluster shapes[1] that deviate non-trivially from a "nice" Gaussian distribution.

We thus propose a procedure to create a multimodal distribution from which such synthetic datasets can be drawn. The procedure takes as input the dimension of the space, the desired number of modes K_{true}, and a handful of interpretable tuning parameters that control the separation and shape of the modes. Our procedure also includes a verification step that ensures, with reasonably high probability, that there are exactly K_{true} distinct modes in the mixture model.

At a high level, our procedure consists of five steps, each of which we will outline in more detail in the following sections.

1. Choose input parameters. These include a closeness index β governing the required separation between components and parameters governing the shape of the component distributions.

2. Set the locations and average variance of the mixture components based on the specified dimension, number of components, and required separation between components.

3. Draw the individual weights and variances of each component based on the average variance and input parameters governing the spread of the weights and variances.

4. Shape each component using a sequence of randomly drawn, invertible transforms. The distribution of each component is based on a symmetric Gaussian distribution; this set of transforms scales, rotates, and translates the coordinate system the distribution is based on to reshape the component distribution. This allows significant diversity

[1] By shape, we mean a level set or contour curve of the joint distribution. A Gaussian, for example, would have a hyperspherical or hyperelliptical shape.

in component shapes while preserving the modal structure of the mixture model.

5. Adjust the components to ensure that all $K_{\text{true}}(K_{\text{true}} - 1)$ pairs of component distributions are sufficiently well separated.

In the following sections, we begin by discussing several other procedures for generating synthetic data and how our method is distinctive. We then define some of the basic notation and equations used throughout the procedure. Section 4.3 describes creating the base component distributions (steps 2-3), and section 4.4 describes shaping these components (step 4). In section 4.5 we describe the verification process (step 5). Section 4.6 describes sampling from the mixture model produced by these steps. We end with a summary of the input parameters to the procedure and a brief description of each.

4.1 Related Work

While numerous papers mention generating synthetic data from mixture models, the vast majority sample data from a mixture model of standard distributions and with visual inspection or no checking at all to ensure the components are not overlapping.

The one exception we found was a synthetic data generator implemented by Pei and Zaiane [PZ]. To generate synthetic data with a specified number of components, they start with complex shapes – e.g. letters, sqaures, etc. – and then locate them in the space, ensuring they do not overlap. These shapes are then filled with a uniform sampling of points. Additionally, clusters can come from Gaussians, and noise points can be added. They provide five levels of difficulty determined by geometric configurations and cluster shapes.

Our method has several advantages over this approach. The first is that in using a random transformation approach coupled with a verification and adjustment process, the complex shapes of our method are more random and arguably more representative of real data than those in [PZ]. Furthermore,

the pdf of our method can easily be realized, while this is more difficult in their method. Finally, the cluster shaping and difficulty are continuously tunable parameters, so the user has more control over the resulting mixture model.

4.2 Definitions

The general procedure of our algorithm is to start with a base distribution – here a standard Normal – and reshape it using a sequence of non-linear transformations. The base distribution, which we denote here as g, is common between all the components. The transformation sequence – which we compose into a single function denoted by $\mathcal{T}_j$ – varies between components. Likewise, the component variance σ_j, component location $\boldsymbol{\mu}_j$, and component weight w_j also vary between components. Putting all these together, we can define the final pdf for mixture component j as

$$h_j(\mathbf{x}) = \mathcal{D}_{\mathbf{x}}(\mathcal{T}_j) g_j\left(\mathcal{T}_j\left(\frac{\mathbf{x}-\boldsymbol{\mu}_j}{\sigma_j}\right)\right) \tag{4.1}$$

Combining all these together into the pdf of the full mixture model gives us

$$h(\mathbf{x}) = \sum_{j=1}^{K_{\text{true}}} w_j h_j(\mathbf{x}). \tag{4.2}$$

where w_j are the weights assigned to each component.

4.2.1 Component Separation and Proximity

As mentioned, one of the distinctive features of our procedures is that the user specifies the difficulty of the problem in part by specifying the minimum separation between components. We quantify this in terms of a proximity index $\mathcal{S}_{jk}$, where a proximity of 0 indicates maximal separation and a proximity of 1 indicates that the modes of the two components are indistinguishable. The user sets the separation parameter β, and our procedure guarantees that all component pairs have a proximity index of at most β.

We define the proximity $\mathcal{S}_{jk}$ between components j and k in terms of

the probability density on the line between the two components. Formally,

$$\mathcal{S}_{jk} = \int_0^1 \min\left\{1, \frac{h'_{jk}\big(u\boldsymbol{\mu}_j + (1-u)\boldsymbol{\mu}_k\big)}{\gamma_{jk}}\right\}\mathrm{d}u \tag{4.3}$$

where the mixture pdf h_{jk} of the jth and kth components is

$$h'_{jk}(\mathbf{x}) = \frac{w_j h_j(\mathbf{x}) + w_k h_k(\mathbf{x})}{w_j + w_k}. \tag{4.4}$$

γ_{jk} is the minimum between the values of the pdf at the two centroids, given by

$$\gamma_{jk} = \min\left\{h'_{jk}(\boldsymbol{\mu}_j), h'_{jk}(\boldsymbol{\mu}_k)\right\} \tag{4.5}$$

Note that $\mathcal{S}_{jk} = 0$ requires the pdf between the the j and k mixture components to be identically 0. We assume that $\boldsymbol{\mu}_j$ is the mode of component j – our procedure is designed to ensure this – so $\mathcal{S}_{jk} = 1$ indicates that there are not two distinct modes in $h'_{jk}(\mathbf{x})$.

Formally, the condition we guarantee our cluster distribution to have in the end is

$$\min_{j,k} \mathcal{S}_{jk} \leq \beta \tag{4.6}$$

We do this by first setting the variances of each of the components so this condition is met. Then, after we shape the components – a procedure which may result in component pairs violating equation (4.6) – we iteratively shrink the variances of subsets of the components until the resulting mixture model satisfies it.

4.3 Stage 1: Pretransformed Component Distributions

The base distribution of each component is a normal[2] with a randomly drawn variance and a location determined by a random process described below. The primary challenge is to choose the component locations and average cluster variance in a way that packs the components together in a geometrically reasonable way.

By geometrically reasonable, we mean avoiding a component configuration in which two components are so close relative to the rest that it is conceptually ambiguous whether data drawn from them should be regarded as one cluster or two. In this ambiguous case, while the variance of the two close components can be adjusted to ensure they are well separated, it makes the separation parameter less meaningful as a summary of the overall structure of the mixture model since it refers to an isolated case. As an analogy, one could argue that the minimum value out of a set of samples carries more information about the distribution if the sampling distribution is known not to generate outliers. Ideally, then, we want a configuration in which the many of the cluster pairs have a separation close the required separation parameter.

4.3.1 Choosing Locations for the Components

Practically, there are several options. One viable option is to generate a random dataset from a single normal distribution, then cluster it using k-means with the desired number of centroids. The resulting cluster locations would then be the new component centers. This would ensure that the clusters are positioned with reasonably good separation. However, this may also tend to position the components too evenly, a possibly undesirable effect (though one could severely limit the number of iterations, or number of points in the test dataset, to prevent it).

The current version of our software chooses the cluster centers by it-

[2]While we use the normal as our base distribution, in theory any symmetric, unimodal distribution would work.

eratively refining a set of candidate component centers. It starts with a list $\mathcal{M}_1 = \left\{\mathbf{M}_{11}, \mathbf{M}_{12}, ..., \mathbf{M}_{1N_\mathrm{b}}\right\}$ of N_b candidate sets $\mathbf{M}_{1i}$, where $\mathbf{M}_{1i} = \{\mu_{1i1}, \mu_{1i2}, ..., \mu_{1iK}\}$ with each element consisting of K centers drawn from a spherical normal distribution. The algorithm then iteratively refines this list a specified number of times using a type of max-min criteria. At each iteration, it chooses the set of centers from the list having the largest minimum distance between centers, then forming a new list of sets of centers based on that set.

Let $\mathcal{M}_t = \left\{\mathbf{M}_{t1}, \mathbf{M}_{t2}, ..., \mathbf{M}_{tN_\mathrm{b}}\right\}$ be the list of sets at iteration t. We then choose a set $\mathbf{M}_t^\star$ from this as

$$\mathbf{M}_t^\star = \underset{\mathbf{M}\in\mathcal{M}_t}{\operatorname{argmax}} \min_{\boldsymbol{\mu}_1,\boldsymbol{\mu}_2\in\mathbf{M}_t} \|\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2\|_2 \tag{4.7}$$

$$\mathcal{M}_{t+1} = \mathbf{M}_t^\star \cup \text{newCentersSetList}\left(\mathbf{M}_t^\star\right) \tag{4.8}$$

where newCentersSetList generates a new list of sets of centers by randomly replacing one of the centers in the pair of two closest points – breaking that pair – with a new point randomly drawn.

By choosing the next $\mathbf{M}_{t+1}^\star$ from the union of a new, randomly generated list and $\mathbf{M}_t^\star$, we ensure that at each iteration our locations of centers do not get any worse. Furthermore, if the smallest inter-center distance is a significant outlier, the probability that an iteration increases this statistic is quite high as it is easy to find an alternate location. We repeat this iterative step a specified number of times.

4.3.2 Setting the Mean of the Initial Cluster Variance Distribution

Recall that the mean cluster variance is not one of the user specified parameters; rather, this must be determined by the user-set minimum separation parameter β. We choose the variance of the clusters by initially setting all base components – at this stage, they are symmetric Normals – to a common variance determined by distance between the two closest points and

the minimum separation parameter given by the user.

From equation (4.6), this means we set σ_{avg} such that

$$\mathcal{S}'_{jk} = \beta \text{ where } j, k = \underset{j',k'}{\text{argmin}} \|\boldsymbol{\mu}_{j'} - \boldsymbol{\mu}_{k'}\|_2 \tag{4.9}$$

where $\mathcal{S}'_{jk}$ is the separation parameter of symmetric Normals with variances equal to σ_{avg} and locations given by $\boldsymbol{\mu}_j$ and $\boldsymbol{\mu}_k$. Formally, this translates into determining σ_{avg} by numerically solving the following equation for σ_{avg}:

$$\frac{1 - 2F\left(-\frac{d}{\sigma_{\text{avg}}}\right)}{\frac{d}{\sigma_{\text{avg}}}\left(f(0) + f\left(-\frac{d}{\sigma_{\text{avg}}}\right)\right)} = \beta \tag{4.10}$$

where d is the distance between the two components, f is the pdf and F is the cdf of the standard normal distribution. Practically, this is done numerically using an iterative bisection algorithm.

Once the mean of the cluster variance distribution is set, we sample the variance of individual components as described in the next section. Note that these variances may be further tuned to ensure that the condition given in equation (4.6) holds for the final mixture model.

4.3.3 Individual Mixture Component Settings

There are two input parameters that control the spread of the individual component weights and variances, s_w and s_σ. After the component locations and σ^2_{avg} are set, we draw the individual cluster variances from a distribution indexed by σ^2_{avg} and an input parameter, s_σ, that governs the spread around σ^2_{avg} of the individual component variances. Likewise, the component weights are drawn from a distribution indexed by an input parameter s_w that governs the standard deviation of the component weights around $1/K_{\text{true}}$.

The input parameter s_σ governs the spread of the component variances. Formally, we use the model $\sigma_j = \sigma_{\text{avg}}\sigma'_j$, where σ_j is a random variable from a Gamma distribution with $\mathbb{E}\,\sigma_j = 1$ and $\text{Var}\ \sigma_j = s^2_\sigma$. Formally,

$$(\sigma'_{j1}, \sigma'_{j2}, ..., \sigma'_{jK}) \overset{\text{iid}}{\sim} \mathit{Gamma}\left(\frac{1}{s_\sigma^2}, s_\sigma^2\right) \tag{4.11}$$

$$[\sigma_1, \sigma_2, ..., \sigma_K] = \sigma_{\text{avg}}[\sigma'_1, \sigma'_2, ..., \sigma'_K] \tag{4.12}$$

Similarly, the input parameter s_w governs the spread of the component weights around the mean weight, $1/K_{\text{true}}$. Practically, we draw the weights from a Gamma distribution with mean 1 and variance s_w^2 and then normalize the resulting distribution. However, this is the same as a Dirichlet theorem, as it can be proved that if

$$Y_i \sim \mathit{Gamma}(\alpha_i, 1) \tag{4.13}$$

and

$$V = \sum_{i=1}^{n} Y_i \tag{4.14}$$

then

$$\left(\frac{Y_1}{V}, \frac{Y_2}{V}, \cdots, \frac{Y_n}{V}\right) \sim \mathit{Dirichlet}(\alpha_1, \alpha_2, ..., \alpha_n) \tag{4.15}$$

Here, then, drawing the weights is formally the same as drawing them from a Dirichlet distribution with parameters $[K_{\text{true}}/s_w^2, ..., K_{\text{true}}/s_w^2]$. Thus

$$(w_1, w_2, ..., w_{K_{\text{true}}}) \sim \mathit{Dir}\left(\frac{K_{\text{true}}}{s_w^2}, \cdots, \frac{K_{\text{true}}}{s_w^2}\right) \tag{4.16}$$

Unlike the component variances, which may be adjusted later, the parameters drawn here are the final weights for the mixture components.

4.4 Stage 2: Shaping the Components

While other methods use complex base shapes to get complex structure in the resulting distribution, our approach is to generate a sequence of invertible transformation functions that each operate on one or two dimensions

of the input vector. There are three classes of transformations, rotation, scaling, and translation. Rotations rotate the two components a random amount; scaling transformations scale one component, and translations add a value determined by one component to the other. These pair of dimensions that each of these transformations is applied to is chosen at random, and the output of one transformation is fed to the next. Shaping the components using these methods is a distinctive of our method, and we describe each of them in detail below.

The four user parameters governing this process are τ_{rotation}, τ_{scaling}, $\tau_{\text{translation}}$, and α. These are, respectively, the average number of rotations, scalings, and translations per dimension component. Recall that p denotes the dimension. Specifically, there are $\tau_{\text{rotation}}p/2$ rotations, $\tau_{\text{scaling}}p$ scaling transformations, and $\tau_{\text{translation}}p$ translations in total. The order and the components that each operate on are chosen randomly. Each of these transformation types (except rotations) take a severity parameter, α, that ranges between 0 and 1. When $\alpha = 0$, the transformations have no effect on the component shapes, but higher values of α cause increasing changes in the shape.

4.4.1 Formal Notation

Formally, the entire operation can be seen as a sequence of nested functions. For notational conciseness, we omit the index of the mixture model component. The composite transformation function $\mathcal{T}(\mathbf{x})$ is given by

$$\mathcal{T}(\mathbf{x}) = T_r(T_{r-1}(\cdots T_2(T_1(\mathbf{x}))\cdots)) \tag{4.17}$$

$$= (T_r \circ T_{r-1} \circ \cdots \circ T_2 \circ T_1)(\mathbf{x}) \tag{4.18}$$

where T_t is the tth transformation function. The distribution function after this set of transformations is then

$$h(\mathbf{x}) = \mathrm{D}_{\mathbf{x}}\mathcal{T}(\mathbf{x})g(\mathcal{T}(\mathbf{x})) \tag{4.19}$$

$$= \mathrm{D}_{\mathbf{y}_{r-1}}(T_r) \cdot \mathrm{D}_{\mathbf{y}_{r-2}}(T_{r-1}) \cdots \mathrm{D}_{\mathbf{y}_1}(T_2) \cdot \mathrm{D}_{\mathbf{x}}(T_1) g(\mathcal{T}(\mathbf{x})) \tag{4.20}$$

$$= \mathcal{D}_{\mathbf{x}}(\mathcal{T}) g(\mathcal{T}(\mathbf{x})) \tag{4.21}$$

where $\mathrm{D}_{\mathbf{z}}(f)$ is the determinant of the Jacobian of f evaluated at $\mathbf{z}$, $\mathbf{y}_t$ is the location of the current value after the first t evaluations in $\mathcal{T}$, i.e.

$$\mathbf{y}_t = (T_t \circ T_{t-1} \circ \cdots \circ T_2 \circ T_1)(\mathbf{x}) \tag{4.22}$$

and $\mathcal{D}_{\mathbf{x}}(\mathcal{T})$ collects the sequence of Jacobians into a single term.

4.4.2 Acceptable Transformation Functions

Obviously, choosing classes of transformation functions must be done carefully. Perhaps the most important criteria is that our aggregate transformation function needs to preserve the unimodal structure of the distribution; otherwise, the final mixture model will not have the correct number of modes. Second, the reshaping operation must not be undesirably severe.

Our method ensures that the first criteria is met by drawing each individual transformation from classes that will preserve the unimodal structure of the distribution and, with reasonably high probability, cannot yield a sequence or ordering that destroys the unimodal structure of the component.[3]

We meet the second criteria by admitting only transformations that pass several tests. Denote the composite transformation sequence at step t as

$$\mathcal{T}_t = T_t \circ \mathcal{T}_{t-1} \tag{4.23}$$

$$= T_t \circ T_{t-1} \circ \cdots \circ T_2 \circ T_1. \tag{4.24}$$

A proposed transformation function T' passes the test if it keeps a set of $2p$

[3]We believe that it is impossible to sequence the possible set of transformations in such a way, but have not proved this rigorously.

points located at $\{\pm\hat{\mathbf{e}}_1, \pm\hat{\mathbf{e}}_2, ..., \pm\hat{\mathbf{e}}_p\}$ within $r_{\max}$ of the origin, where $\hat{\mathbf{e}}_q$ is the unit vector of the qth dimension.

The second test is similar, except that T' passes the test if $T' \circ \mathcal{T}_{t-1}$ keeps the same set of points within $r_{\max}$ of the origin. The main idea of these transformations is to prevent the transformations from overly spreading apart the central mode of the components.

Optionally, if more cluster consistency is required, T' must pass a second pair of tests similar to the first pair, except these ensure that the set of points $\{\pm 2\hat{\mathbf{e}}_1, \pm 2\hat{\mathbf{e}}_2, ..., \pm 2\hat{\mathbf{e}}_p\}$ stays within $2r_{\max}$ of the origin. This second round of tests place more severe restrictions on some of the translation functions, as many of these apply a much larger translations to points farther from the center.

When constructing the sequence of transformations, we fix initially how the specified number of each type is ordered. However, when setting the parameters for one particular transformation, we redraw the component(s) and transformation parameters until it passes the required tests.

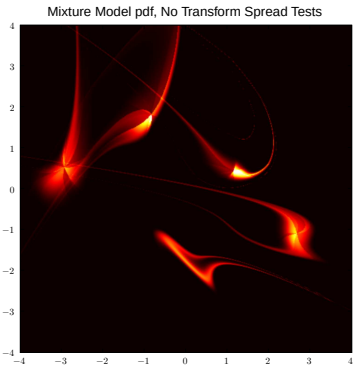
Figure 4.1 show three plots of the pdf of a 5 component mixture model. The input parameters are identical except for the use of these tests. The left one shows the mixture model with no tests, the middle with transformations filtered using the first pair of tests, and the right plot with transformations filtered using both pairs of tests.

4.4.3 Rotation

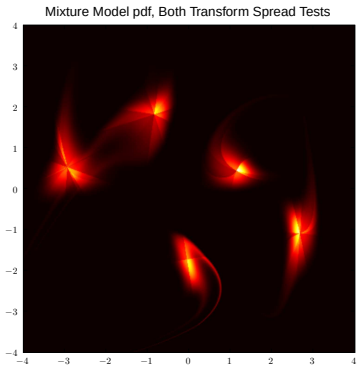
A rotation function T_{rotation} rotates two components m_1 and m_2 of the the imput vector $\mathbf{y}_t$ by θ. In choosing a function from this class, we choose $m_1, m_2 \sim \mathcal{U}n_{\{1,...,p\}}, m_1 \neq m_2$ and $\theta \sim \mathcal{U}n_{[0,2\pi)}$, then the two entries in the point vector would be

$$\begin{bmatrix} y_{t+1,m_1} \\ y_{t+1,m_2} \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} y_{t,m_1} \\ y_{t,m_2} \end{bmatrix} \tag{4.25}$$

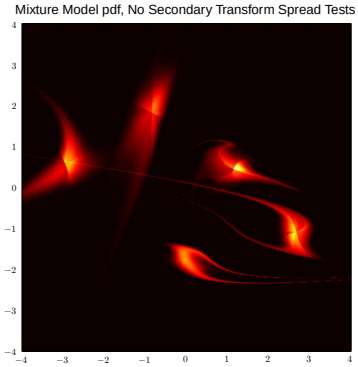
The full transformed point would then be:



(a) the pdf of a mixture model with no transformation checks.



(b) the pdf of a mixture model with only first order transformation checks.



(c) the pdf of a mixture model with both transformation checks.

Figure 4.1: Example plots of three 2d mixture models when there are no checks on the transforms, first order checks, and both first and second order checks. Note that with no transformation checks, exceedingly long and thin cluster shapes are much more likely.

$$T_{\text{rotation}}(\mathbf{y}) = [y_1, y_2, ..., y_{m_1-1}, y_{m_1}\cos\theta + y_{m_2}\sin\theta, ..., \\ -y_{m_1}\cos\theta + y_{m_2}\sin\theta, y_{m_2+1}, ..., y_p]^{\mathrm{T}} \quad (4.26)$$

Note that rotation does not affect the normalizing constant of the distribution, so $\mathrm{D}_{\mathbf{y}_t}(T_{\text{rotation}}) = 1$. The inverse, needed for the sampling stage, is also easily calculated by inverting the rotation.

4.4.4 Coordinate Translation

A translation function $T_{\text{translation}}$ takes one dimension component and uses it to determine the amount of translation for another component. It is determined by choosing a scalar function f randomly from the set listed below, a base component b of $\mathbf{y}$, a separate mapping component m, and a term A drawn from a Gamma distribution indexed by user input parameters. It then maps y_m to $y_m + f(y_b)$. Thus

$$\mathscr{T}_b(\mathbf{y}) = [y_1, y_2, ..., y_{m-1}, y_m + f(y_b, A), y_{m+1}, ..., y_p]^{\mathrm{T}} \tag{4.27}$$

We chose the list below as a non-exhaustive set of function classes that will, with a reasonably high probability, fulfill the criteria given in section 4.4.2. Additionally, the input parameter, α dictates the expected severity of the transformation; $\alpha = 0$ has no expected effect and $\alpha = 1$ has a severe expected effect (recall that all the distributions before the transformation stage have a variance of 1).

1. $f_1(z, A) = Az$
2. $f_2(z, A) = Az^2$, $A \sim \mathit{Gamma}(\alpha, 1)$
3. $f_3(z, A) = Az^3$, $A \sim \mathit{Gamma}(\alpha, 1)$
4. $f_4(z, A) = e^{Az} - 1$, $A \sim \mathit{Gamma}(\alpha, 1)$

where $A \sim \mathit{Gamma}(\alpha, 1)$ is the severity of the effect and α is a user input parameter. Note that the Gamma function is parameterized so $\mathbb{E}\, A = \mathrm{Var}\, A = \alpha$.

The Jacobian in this case is the detriment of the identity matrix with one non-zero off-diagonal and is thus 1. The inverse is easy to compute:

$$\mathscr{T}_b^{-1}(\mathbf{y}) = [y_1, y_2, ..., y_{m-1}, y_m - f(y_b), y_{m+1}, ..., y_p]^{\mathrm{T}} \tag{4.28}$$

4.4.5 Coordinate Scaling

A function T_{scaling} chooses a random component m of $\mathbf{y}$ and scales it by an amount A. Thus

$$T_t(\mathbf{y}) = [y_1, y_2, ..., y_{m-1}, Ay_m, y_{m+1}, ..., y_d]^{\mathrm{T}} \tag{4.29}$$

The scaling factor A is drawn from a Gamma distribution parameterized so that $\mathbb{E}\, A = 1$ and $\text{Var}\ A = \alpha$. Thus $\alpha = 0$ again denotes no expected effect and $\alpha = 1$ denotes relatively severe expected effect.

We are only modifying one component, so the Jacobian for this function is just

$$\left|\mathrm{D}_{\mathbf{y}} T_{\text{scaling}}\right| = A \tag{4.30}$$

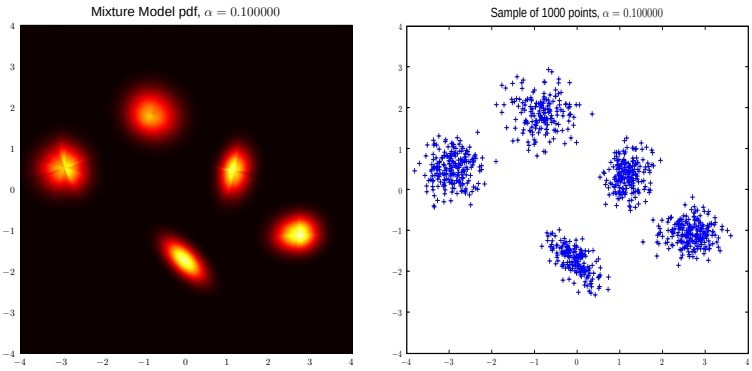
The inverse is also trivial:

$$T^{-1}_{\text{scaling}}(\mathbf{y}) = \left[y_1, y_2, ..., y_{m-1}, \frac{y_m}{A}, y_{m+1}, ..., y_d\right]^{\mathrm{T}} \tag{4.31}$$

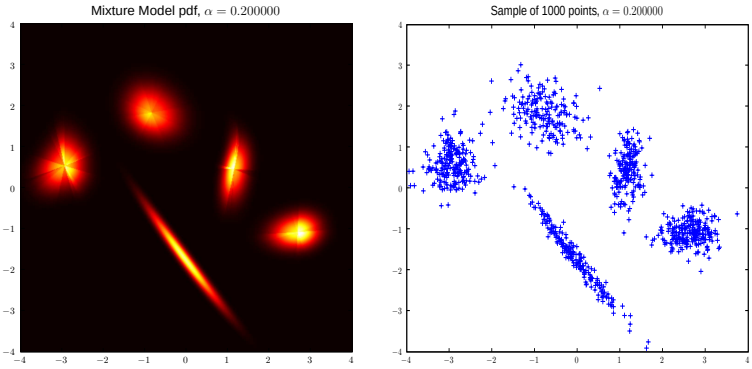
4.4.6 Example Cluster Shaping Parameters

We show in Figure 4.2 mixture models, with corresponding samples, having values of the transformation severity parameter α set at 0.1, 0.2, 0.4, and 0.75. The number of components is 5, and both s_σ and s_w were set to 1. As can be seen, this parameter has a significant effect on the type of mixture model produced.

In Figure 4.3, we show the pdf and corresponding samples for a 5 component mixture models as a function of the number of transformations applied per dimension component. This parameter, along with α, has the most influence over the component shapes. In general, the more transformations, the less the components resemble Gaussian distributions.



(a) A 2d mixture model with corresponding samples and $\alpha = 0.1$



(b) A 2d mixture model with corresponding samples and $\alpha = 0.2$

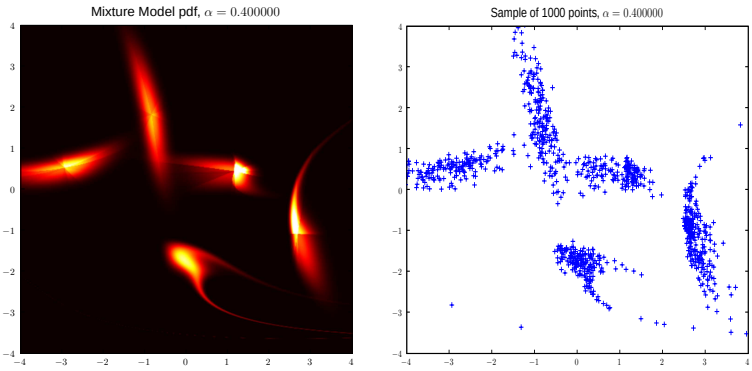
Figure 4.2: 2d mixture models with corresponding samples (2000) as a function of the transform severity parameter α

4.5 Stage 3: Adjusting the Proposed Cluster Distribution

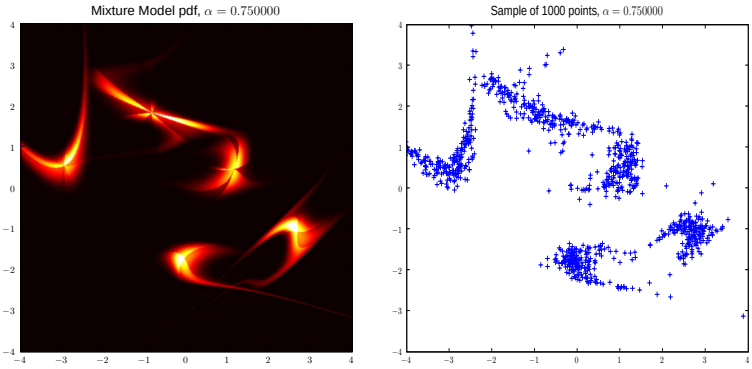
One postcondition of the model generated by our procedure is that all components are sufficiently well separated, i.e.

$$\mathcal{S}_{jk} \leq \beta \ \forall j, k \tag{4.32}$$

The first step in ensuring this condition is met was to set the mean of the cluster distribution. However, when drawing the component variances from



(c) A 2d mixture model with corresponding samples and $\alpha = 0.4$

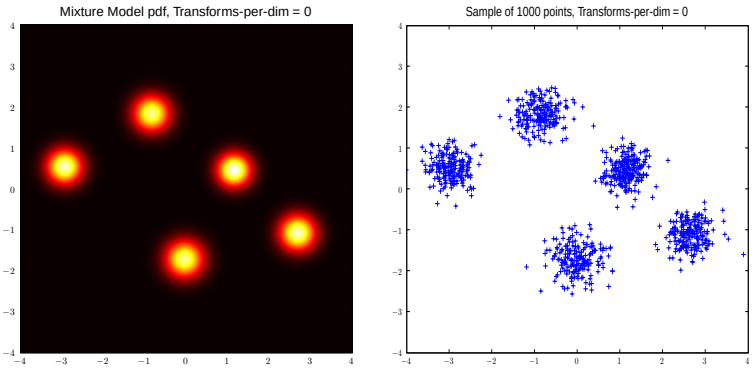


(d) A 2d mixture model with corresponding samples and $\alpha = 0.75$

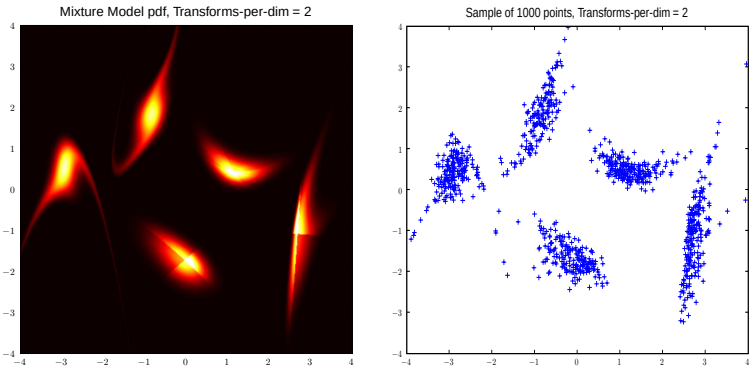
Figure 4.2: 2d mixture models with corresponding samples (2000). The severity of the transformations is far greater than the previous two in Figure 4.2.

a distribution with nonzero variance and shaping the components, there is no guarantee that this is enough. Thus to meet the condition, we selectively shrink the variances of the components until equation (4.32) is satisfied. Recall that the shaping procedure is independent of the variance, so the rest of our process is unaffected.

To shrink this, we give every component a score based on how severely it contributes to violating equation (4.32). Specifically, the score $\mathcal{S}_j$ for component j is



(a) A 2d mixture model with corresponding samples and no transforms per dimension.

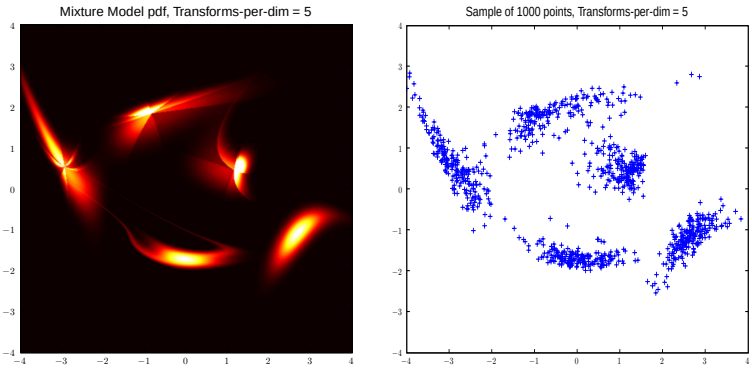


(b) A 2d mixture model with corresponding samples and 2 transforms per dimension.

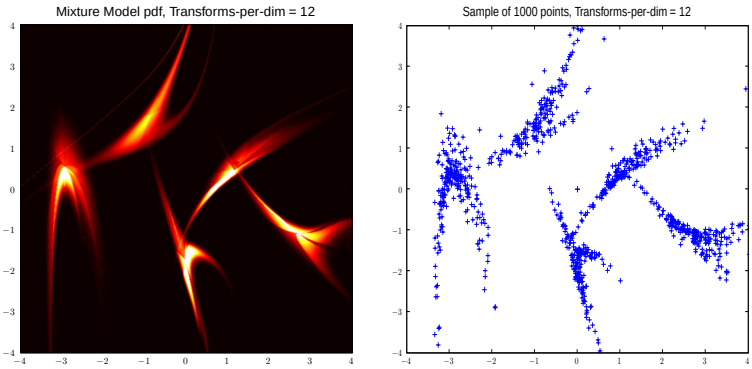
Figure 4.3: 2d mixture models with corresponding samples (2000) as a function of the number of transformations per dimension. The severity of the transformations was set to 0.4, and all the types had equal representation.

$$\mathcal{S}_j = \sum_{k \neq j} \max\{0, \mathcal{S}_{jk} - \beta\}, \tag{4.33}$$

We then shrink the component with the highest score until its score is 0. This causes all the component pairs that component j is a member of to satisfy equation (4.32). This, of course, will also reduce the score of other



(c) A 2d mixture model with corresponding samples and 5 transforms per dimension.



(d) A 2d mixture model with corresponding samples and 12 transforms per dimension.

Figure 4.3: 2d mixture models with corresponding samples (2000) as a function of the number of transformations per dimension.

components. We then recalculate the scores as needed and repeat the procedure, stopping when there are no violations.

4.6 Sampling from the Distribution

The only remaining process to describe is how to sample a set of N points $(\mathbf{X}_1, \mathbf{X}_2, ..., \mathbf{X}_n$ from $h(\mathbf{x})$. There are two aspects to this process; the first is choosing the number of points to draw from each component, and the

second is sampling from a given component.

4.6.1 Determining Component Sample Sizes

We draw the number of points for the clusters from a type of truncated multinomial distribution where an input parameter, $n_{\min}$, specifies the minimum number of points drawn from each component. Allowing this as an input parameter helps ensure that all the modes of the distribution are represented, in some way, in the final dataset. In this section, we present the algorithm we use to generate the cluster sizing information.

We ensure that at least $n_{\min}$ points are drawn from each component using a simple recursive algorithm that is guaranteed to produce the number of points to be drawn from each cluster provided that $n_{\min}K_{\text{true}} \leq N$. The main idea is simple. We start with a vector $\mathbf{n} = (n_1, n_2, ..., n_K)$ drawn from a multinomial distribution with weights set as describe in section 4.3.3. If any of the cluster sizings have less than $n_{\min}$ points, we set these to $n_{\min}$ and then randomly decrease the sizes of clusters that have more than $n_{\min}$ points so that the total number of points is still n. We then repeat this until all the clusters have size $n_{\min}$ or greater. We present the algorithm more formally in algorithm (6).

4.6.2 Sampling from the Components

Now all that is left is to sample n_j points from $h_j(\mathbf{x})$ for each cluster j. We can do this easily by drawing samples from the base distribution g_j and then transforming them by inverting the transformation functions. Formally, we have that

$$\mathbf{X} \sim g(\mathbf{x}) \tag{4.34}$$

$$h_j(\mathbf{y}) \propto g\left(\mathcal{T}_j\left(\frac{\mathbf{y} - \boldsymbol{\mu}_j}{\sigma_j}\right)\right) \tag{4.35}$$

so if $\mathbf{X} \sim g_j(\mathbf{x})$,

Algorithm 6: Determining the number of points to assign to draw from each mixture model component.

Input: The total number of points N, the minimum number of points $n_{\min}$, and a vector $\mathbf{w} = (w_1, w_2, ..., w_K)$ giving the weighting of each component.

Output: A vector $\mathbf{n} = (n_1, n_2, ..., n_K)$ giving the number of points to be drawn from each component.

assert $N \geq n_{\min} * K$ $(n_1, n_2, ..., n_K) \leftarrow \textit{Multinomial}(N, \mathbf{w})$
while $\exists i$ *s.t.* $n_i < n_{\min}$ **do**
  $A \leftarrow \{i : n_i < n_{\min}\}$
  $B \leftarrow \{i : n_i \geq n_{\min} + 1\}$
  $m \leftarrow \sum_{i \in A} n_{\min} - n_i$
  for $i \in A$ **do** $n_i = n_{\min}$
  $d \leftarrow \textit{Multinomial}(m, [\max\{0, n_i - n_{\min}\} / \sum_i \max\{0, n_i - n_{\min}\}])$
  for $i \in B$ **do** $n_i \leftarrow n_i - d_i$
end
return n

$$\mathbf{Y} = \sigma_j \mathcal{T}_j^{-1}(\mathbf{X}) + \mu_j \sim h_j(\mathbf{y}) \tag{4.36}$$

Because each of the transforms described in section 4.4 is invertible, we are able to efficiently sample from the distribution and thus easily create the dataset.

4.7 User Set Parameters

Our procedure has a collection of tunable parameters that control the shape of the mixture model components, how well separated they are, and, in general, the difficulty of a sampled dataset for both clustering algorithms and validation procedures. Most of the parameters indexing the difficulty take values between 0 and 1, with larger values indicating a more difficult problem.

We summarize the parameters in the tables below:

Mixture Model Properties	
K_{true}	The number of mixture components.
p	The dimension of the space.
β	The minimum separation between modes, as defined in equation (4.3). If $\beta = 0$, the components are maximumly separated (reducing the components to delta functions), and $\beta = 1$ denotes no separation required.
s_σ	Controls the spread of the cluster variances as outlined in section 4.3.3.
s_w	Controls the spread of the cluster weights around $1/K_{\text{true}}$ as described in section 4.3.3.
$n_{\min}$	The minimum number of points in a cluster; if any cluster has fewer points than this, the weights and number of points in the clusters are redrawn.

Mixture Component Properties	
τ_{rotation}	The average number of rotation operations affecting each entry in the point vector.
τ_{scaling}	The average number of scaling operations affecting each entry in the point vector.
$\tau_{\text{translation}}$	The average number of translation operations affecting each entry in the point vector.
α	The expected severity of each transformation. If $\alpha = 0$ there is no change; $\alpha = 1$ indicates a severe expected transformation.

4.8 Conclusion

Now that we have outlined a reasonably reliable way of generating synthetic data for testing clustering and cluster validation procures. In particular, while our method can produce data with regular Gaussian components, its strength is in producing data with clusters that can not easily be modeled by any of the standard distributions. We are now ready to compare our

method with all the others using this data.

Chapter 5

Testing and Verification

We show that our approach compares quite favorably to other leading methods. We tested our method extensively on two classes of data, ANOVA data with symmetric Gaussian mixture components, and shaped data, where the data was non-Gaussian and shaped by the procedure presented in chapter 4.

We found that our method outperformed all the other methods when the data was shaped, and outperformed the data perturbation methods and was nearly comparable to the gap statistic when the points were drawn from spherical Gaussians. This was the expected behavior, as the squared-error cost function in the gap statistic would work far better when the clusters were symmetric than when they were shaped. Our method makes only weak modeling assumptions and thus performs significantly better on data that does not fit the modeling assumptions made by other methods. We also tested three varieties of data perturbation methods, and found them to all work fairly well across both ANOVA and shaped data, consistently proving to be decent but slightly less accurate than our method.

We first present in more detail each of the methods tested in section 5.1. We describe the setup of the tests in the next section. Then, in section 5.3, we present a detailed analysis of the results, and in section 5.4 we conclude with a summary discussion.

5.1 Methods Tested

We compare three classes of cluster validity estimation techniques. The first is our method, which has several variants depending on the type of baseline distance matrix, summary statistic, and prediction method. The second is the gap statistic, described in detail in section 1.3. Finally, we compare three different types of data perturbation using both the Hubert-Arabie adjusted Rand index and the Variation of Information. These methods are variants of one of the more popular data perturbation methods, subsampling. While our tests against data perturbation methods are far from exhaustive – such a comparison would be quite difficult – we believe the results on these methods are indicative of many other data stability approaches. All of these methods are outlined in more detail in the following sections.

Each of the techniques above is a method to get a stability estimate for a single clustering. To use these clusterings to predict the number of clusters in a dataset, we generate a set of candidate clusterings of the data, corresponding to a range of possible numbers of clusters, and calculate the specified validity indices of each. We then use this set of validity estimates to predict the number of clusters; this stage is discussed in section 5.1.4.

5.1.1 Bayesian Cluster Validation Methods

From our method, we estimate the validity of a clustering using the averaged pointwise stability as described in section 2.2.2. We use this in combination with scaled distance perturbations (section 3.1) with a location exponential prior (section 3.3). The stability of the clustering is the difference in average pointwise stability between the given data and a null baseline clustering.

We tested the three different baseline types described in section 3.2.3: a reclustered uniform sampling in the bounding box of the data, which we label as APW-RC, a similar uniform sampling but recycling the cluster centers of the clustered data (APW-U), and a baseline distance matrix formed by permuting the original (APW-Perm). Between these three, we found that in the lower dimensions and data with simpler models, the APW-RC tended to work best followed by the APW-U and APW-Perm, but, as the dimension

or model complexity increased, APW-Perm tended to dominate.

5.1.2 Gap Statistic

The second method we tested was gap statistic, described in detail in section 1.3. It uses a reclustered uniform distribution in the bounding box of the data as its baseline.[1] Given the gaps for each k, we test the method using two predictors. First is the one the authors propose, and the second picks the smallest k within a standard deviation of the best. These will be be described more in section 5.1.4.

5.1.3 Data Perturbation

We chose to compare our method against three variants of the subsampling techniques described in section 1.2.1. Recall that subsampling, closely related to bootstrapping, draws n separate datasets from the original data, each containing a subset of the original points. Running each method consists of cluster a set of subsampled datasets, comparing each to the unperturbed clustering using the index, and taking the average as the final stability estimate.

All of the methods below use one of the similarity indices described in section 1.2.2 to compare the perturbed clustering with the original clustering. The first is the Hubert-Arabie adjusted Rand index and the second is the Variation of Information. In general, we found the second to perform better.

In the first method, we randomly draw a subset of the data, recluster it, and compare the resulting labeling against the labeling of the same points in the unperturbed data set. We compared this using 10%, 20%, 30%, 40%, 50%, 60%, 70%, 80%, and 90% of the original points, for 9 stability indices in total. The final stability index v_k is the average of these, and s_k is the population standard deviation. We label this method as SS-SizeRange.

The second method is more careful about which points it chooses. For it,

[1] We did attempt to use the gap statistic with the other two baseline types described in in reference to our method, but doing this causes it to perform horribly so we omit those results.

we break the data into 10 non-overlapping folds and cluster the data from 9 of these folds at each run, leaving out a different fold each time for 10 total runs. Each of these runs is then evaluated based on how well they predict the labels on the original data. Again, the final stability index v_k is the average of these, and s_k is the population standard deviation. This method we call SS-Folds.

The third method is similar to SS-SizeRange, but differs in that there are 10 independent draws of $n/2$ points each rather than a variety of subset sizes. The rational here, like bootstrapping, is that we're simulating multiple samples of the same data set. The final stability index v_k and standard deviation v_k are set the same way. We'll call this method SS-Draws later on, and, since we present more detailed analysis that involves the two different similarity indices, we refer to the one using the Hubert-Arabie adjusted Rand index as SS-Draws-AR and the one using the Variation of Information as SS-Draws-VI.

5.1.4 Predictors

Each of the above techniques yields a list of validity estimates,[2] where each k tested has a validity index v_k. Additionally, recall that each validity estimate has an associated standard deviation s_k. Given the list of validity indices and standard deviations, we test several predictors for each method, where here a predictor is simply a way to extract an estimate $\hat{K}$ for the number of clusters from the lists v_k and s_k. For each validation method, we present results from all the predictors that work reasonably well on at least some of the datasets.

Estimating the number of clusters is done using one of three methods, though not all combinations of $\hat{K}$ estimator and validity estimation work. The first method is to simply choose the k that has the best validity estimate (we refer to this estimation method as BestK).

The first predictor, which we call BestK, is an obvious first choice. It chooses $\hat{K}$ as the k that has the highest validity index v_k. This ap-

[2]While the perturbation based methods produce *stability* indices, we here use *validity* indices to include the gap statistic, which is not stability based.

proach tended to work best for the data perturbation approaches, but lost to StdBeforeBest when used with our method.

The second predictor, which we label StdBeforeBest, is a standard method in machine learning. For it, we estimate k as the smallest k that is within one standard deviation of the best, i.e.

$$\hat{K} = \min \{k : v_k \geq v_\ell - s_\ell\}, \text{ where } \ell = \underset{\ell}{\operatorname{argmax}}\, v_\ell \tag{5.1}$$

This method usually outperformed BestK with our method, but lost on the data perturbation methods. It is the recommended method for our cluster validation approach.

The third method, proposed as part of the original gap statistic, looks for the place in the curve where the validity index levels off and stops improving with increasing k. Specifically,

$$\hat{K} = \min \{k : v_k \geq v_{k+1} - s_{k+1}\} \tag{5.2}$$

This prediction method only worked in combination with the gap statistic, so we omit results using it in combination with the other methods.

5.2 Test Setup

As we discuss in chapter 1, the canonical test for cluster validation methods is how well they predict the true number of clusters in a given dataset. While this method is somewhat artificial – often, in real data, it is difficult to even say what the true number of clusters is – it does allow for some helpful comparisons. However, in much of the literature, the tests on synthetic data (and on real data) have been on only a handful of datasets and thus the comparisons have revealed limited information about the methods. Our simulation is massive by comparison; in total we compare the performance of each of the methods on 22,500 datasets of differing type, dimension, number of true clusters, sample size, and cluster shape. These help reveal the strengths and weaknesses of each method.

5.2.1 Data Types

As mentioned, we test the methods using two types of data, the first being ANOVA style data with spherical cluster centers, and the second being non-Gaussian data shaped according to the method described in chapter 4. For each of these types, we generate datasets having 2, 10, 20, 50, and 100 dimensions and having 2 through 16 true clusters K_{true}. For each type, dimension, and K_{true}, we do 100 runs with 750 points and 50 runs with 100 points to test both medium and small sized data sets. Thus there are $2 \times 5 \times 15 \times 150 = 22500$ datasets total.

Parameters for Shaped Data

Recall from chapter 4 that a handful of parameters govern the shape and "difficulty" of the generated mixture model as a clustering problem and a cluster validation problem. We found that the relative performance of our methods were not very sensitive to these input parameters; we thus adjusted them so that the high score on each data type indicated good but not perfect performance (in the 7-10 range according to the scoring system described below).

5.2.2 Clustering

We used a slight variation on vanilla k-means as our primary clustering algorithm. Our results could potentially be limited by the accuracy of the clustering algorithm, as an incorrectly clustering a data set K_{true} clusters with K_{true} centroids should not be any more stable than clustering it into the incorrect number of clusters. Thus, to minimize this potential problem and ensure that each run gave excellent results, our clustering function returns the best clustering out of 20 runs of k-means as measured by the k-means cost function. This seemed to give excellent results.

5.2.3 Quantitative Evaluation

We present the results for each method in several ways. The first scores each method according to its cumulative accuracy on each type and dimension of

data, summarizing how closely it predicts K_{true} on each run and each K_{true} in a given set, normally 2-16. This presentation provides a good summary of how well each method works. The second way we present the results is as a so-called confusion table that shows how many clusters $\hat{K}$ a given method predicts as a function of the true number of clusters K_{true}.

We observed that when a method incorrectly predicts a value $\hat{K}$ for K_{true}, often the predicted value is way off and becomes essentially random across all the possible values of $\hat{K}$. To keep such estimates from skewing the results, we constructed a scoring function that is a discrete variant of the Smoothly Clipped Absolute Deviation (SCAD) loss function [FL01, Fan97], where we give each estimate $\hat{K}$ a score as follows:

$\lvert\hat{K} - K_{\text{true}}\rvert$	Score
0	10
1	5
2	1
> 2	0

Table 5.1: The possible scores of the cluster validation methods.

The final score of a method is the average score across all 100 runs and the possible values of K_{true}.

To present a more detailed picture of how a certain method behaves, we will also present a table giving how often the method predicted the a given number of versus the true number of clusters. In such a table, a perfect method would have 1's along the diagonal where $\hat{K} = K_{\text{true}}$, and zeros elsewhere. This type of table reveals significantly more about the behavior of each of the methods. For example, some methods perform well when K_{true} is small (< 8) but break down when K_{true} is larger (> 10). We present and describe such results in more specific detail later in this chapter.

5.3 Results

The results of our simulation, were, overall, quite favorable toward our method. Using the difference in average pointwise stability between a baseline distance matrix and data performed comparable to the gap statistic and the data perturbation methods on most cases, and in many cases performed substantially better.

In the next four sections, we present the results for two classes of data, ANOVA and shaped, with two different sample sizes, 750 points and 100 points. The ANOVA data, which allows one to make much stronger modeling assumptions, is perhaps easier. Our shaped data does not readily fit any simple models and thus makes things more much more difficult for model based methods such as the gap statistic. In general, our methods beat the gap statistic and data perturbation approaches while generally matching the gap statistic on ANOVA data.

Each entry in the tables described in sections 5.3.1 - 5.3.4 represents the average score of that method across 100 runs at each of 2-16 true mixture components, for an average over 1500 datasets in total. The test k ranges from 2 through 20 clusters on each dataset. In these tables, boldface entries denote the best performing method, and darker shades of blue (gray) denote higher scores.

In section 5.3.5, we present a more comprehensive and detailed analysis of how each method performs by showing histogram-like tables of how many times each of the methods predicted a given $\hat{K}$ when the true number of clusters was K_{true}. This reveals more insight into the various methods, e.g. whether they tend to predict too high or too low or whether they are better at higher or lower k. Finally, in section 5.4, we summarize the important points of the results.

5.3.1 ANOVA Data, Medium Sample Size

We begin with the easiest data type, given in Table (5.2). This table shores how different methods scored on ANOVA type data of varying dimension and 750 points in each dataset. The separation index, as given by equation

Type	Subtype	Predictor	2D Anova	10D Anova	20D Anova	50D Anova	100D Anova
APW	Perm	Best	9.23	**9.88**	**9.44**	9.38	8.86
		Std Before Best	9.42	**9.88**	**9.44**	**9.40**	8.86
	RC Uniform	Best	**9.63**	**9.88**	**9.44**	9.38	8.87
		Std Before Best	9.62	**9.88**	9.43	9.37	8.86
	Uniform	Best	9.22	9.87	**9.44**	**9.40**	**8.90**
		Std Before Best	9.59	**9.88**	**9.44**	9.39	8.87
Gap		Regular	4.39	**9.88**	**9.44**	9.38	8.89
		Std Before Best	9.61	**9.88**	9.40	9.34	8.76
SS	Draws	Best AR	7.10	9.64	9.13	8.95	8.32
		Best VI	6.84	9.65	9.35	9.20	8.71
	Folds	Best AR	5.05	8.81	6.83	7.59	6.84
		Best VI	4.88	8.76	7.81	8.44	7.92
	SizeRange	Best AR	6.36	9.66	8.94	8.91	8.12
		Best VI	6.22	9.62	9.25	9.20	8.59

Table 5.2: Score results for ANOVA type data with 750 points. Boldface numbers indicate the best performing method(s) one each data type, and darker shades of blue (gray) indicate higher scores.

(4.3), was 0.6. In this particular case, our methods do quite well, with one flavor of APW beating or matching all other methods at every dimension tested.[3]

5.3.2 ANOVA Data, Small Sample Size

The results change slightly when the sample size is sufficiently small. In particular, one flavor of the gap statistic has the best score in 3 out of

[3]The astute reader will observe that in 10 and 2o dimensions many of the methods received the same score. This may be due to wrong clusterings with the correct k, which can definitely occur. If they do, then it's possible that all the methods get 100% on the runs with a clustering for the correct k but predictably fail on the clusterings in which it is wrong. Future analysis will deal more directly with this issue.

Type	Subtype	Predictor	2D Anova	10D Anova	20D Anova	50D Anova	100D Anova
APW	Perm	Best	2.74	4.93	7.62	5.79	7.68
		Std Before Best	6.24	7.65	8.34	6.91	7.87
	RC Uniform	Best	8.10	8.00	**8.49**	**7.39**	8.18
		Std Before Best	6.92	6.64	8.11	6.80	8.04
	Uniform	Best	2.30	3.29	6.91	4.39	7.71
		Std Before Best	6.26	6.58	8.06	6.48	8.12
Gap		Regular	3.05	6.43	8.04	6.78	**8.26**
		Std Before Best	**8.24**	**8.27**	8.10	6.91	7.60
SS	Draws	Best AR	5.02	6.66	7.53	7.14	7.73
		Best VI	5.72	6.69	7.88	6.41	8.19
	Folds	Best AR	3.39	5.75	6.53	6.21	6.61
		Best VI	3.45	5.54	6.97	6.40	7.32
	SizeRange	Best AR	3.94	6.41	7.24	6.63	7.30
		Best VI	4.42	6.53	7.61	6.47	7.82

Table 5.3: Score results for small sample size, ANOVA type data with several predictors.

5 of the dimensions. Several flavors of our method, though, were strong contenders; APW-RC-Best in particular, was consistently close or better.

We suspect that the good performance of the gap statistic here is indicative of its strong modeling assumptions. These assumptions hold perfectly in this case and play more of a role because of the small sample size. On the shaped data where these assumptions do not hold, however, the gap statistic performs significantly worse.

It's also noteworthy that with regards to the APW-RC method, simply predicting the most stable k wins over picking the one that is within one standard deviation of the best. This is the one case we observed where the latter is better than or matches the former.

5.3.3 Shaped Data, Medium Sample Size

Type	Subtype	Predictor	2D Shaped	10D Shaped	20D Shaped	50D Shaped	100D Shaped
APW	Perm	Best	6.06	7.31	7.64	6.79	7.21
		Std Before Best	7.84	**8.55**	**8.57**	**7.80**	7.26
	RC Uniform	Best	8.56	7.54	7.79	7.12	**7.46**
		Std Before Best	**8.80**	8.29	8.33	7.54	**7.46**
	Uniform	Best	6.64	7.90	8.04	7.12	7.45
		Std Before Best	8.04	8.26	8.40	7.52	7.45
Gap		Regular	5.48	6.65	6.23	5.47	6.91
		Std Before Best	7.38	1.34	0.77	0.46	1.46
SS	Draws	Best AR	5.49	7.33	7.30	6.59	6.30
		Best VI	5.40	7.46	7.87	7.28	7.10
	Folds	Best AR	3.40	5.45	5.02	4.70	4.47
		Best VI	3.45	5.63	6.04	6.18	5.93
	SizeRange	Best AR	4.69	7.08	7.22	6.30	6.17
		Best VI	4.66	7.20	7.72	7.02	6.94

Table 5.4: Score results for Shaped data with several predictors.

When the data is shaped and the clusters are no longer Gaussian, the story changes. As shown in Table (5.4), the best method is always within our proposed class of validation methods. Most notably, in 2 and 100 dimensions, every variant of our approach wins over all the data perturbation methods and the standard version of the gap statistic. If we restrict ourselves to using the StdBeforeBest predictor – note that it outperforms BestK in every case – the same is true at every dimension. Within our method, APW-Perm, the fastest one computationally, is the best in 10, 20, and 50 dimensions, and APW-RC is the best on the other two sets.[4]

[4]We suspect that APW-Perm would also be the best at 100 dimensions as well if that generated data was more shaped. However, when we ran these tests, some optimizations in the clustering generation code were not in place and thus we weren't able to apply as

The data perturbations methods, in this case, consistently outperformed the gap statistic but did not match our method. Except in 2 dimensions, results using the Variation of Information were better than results using the Hubert-Arabie adjusted Rand index. Also, note that the best method tended to be SS-Draws; we examine this in more detail later.

The proposed version of the gap statistic performed fairly well, outperforming some of the data perturbations methods on some counts but never winning over any of our methods. The one case where the Gap statistic was able to beat all the data perturbation methods and the worst of our methods was in 2 dimensions when using the StdBeforeBest predictor. Note, however, that this predictor performs horribly on the other dimensions.

5.3.4 Shaped Data, Small Sample Size

Having a smaller sample size (Table (5.5)) shifts the results slightly, though the relative performance between the classes is largely the same. In this case, APW-RC, instead of APW-Perm, is usually the best (with the exception of the runs in 50 dimensions, where SS-Draws-VI wins). This is somewhat expected – APW-Perm, in building the baseline on permutations of the original distance matrix, is the least parametric of the methods and thus requires more data to achieve accuracy comparable to methods that make more assumptions.

The gap statistic and data perturbation methods tended to have similar relative scores to the results in the previous section. We don't have a good explanation for why SS-Draws-VI was best on the 50 dimensional data; this remains an open question.

5.3.5 Detailed Comparison

While the tables in the previous sections offer an excellent summary of the performance of each approach. Knowing whether a method tends to overestimate or underestimate the number of clusters, or whether it performs

many transformations to the mixture components as would have been ideal. Thus that data could reasonably be viewed as half way between the shaped data and the ANOVA data.

Type	Subtype	Predictor	2D Shaped	10D Shaped	20D Shaped	50D Shaped	100D Shaped
APW	Perm	Best	1.19	2.89	5.81	6.74	6.86
		Std Before Best	4.59	6.23	7.56	7.36	7.00
	RC Uniform	Best	7.15	6.95	7.61	7.54	7.56
		Std Before Best	**7.97**	**7.04**	**7.68**	7.74	**7.62**
	Uniform	Best	1.16	2.09	4.91	6.75	7.13
		Std Before Best	4.91	5.18	6.85	7.27	7.30
Gap		Regular	3.84	5.96	6.55	6.38	7.19
		Std Before Best	5.27	0.99	0.85	0.72	1.68
SS	Draws	Best AR	5.25	6.62	7.17	7.40	7.24
		Best VI	5.94	6.31	7.33	**7.87**	7.58
	Folds	Best AR	2.72	4.65	5.51	5.87	6.01
		Best VI	3.01	4.62	5.92	6.57	6.90
	SizeRange	Best AR	3.90	5.87	6.46	6.90	6.95
		Best VI	4.46	5.85	6.87	7.35	7.54

Table 5.5: The score results for Shaped data with several predictors.

well only when K_{true} is small, is valuable information. Therefore, in this section, we present histogram-like tables that give this information. We refer to these results as sampling distributions.

Due to space constraints, we present tables for only a subset of the possible combinations of method, predictor, data type, and dimensions. These are to highlight various properties of the methods and what issues one must be aware of. In general, we try to look at the extremes that most illustrate the tendencies of the various methods. Thus we restrict ourselves to 2 and 100 dimensional data but examine the different combinations within these dimensions more thoroughly. Also, we found that the sampling distribution of SS-Draws-VI was generally representative of the other data perturbation methods, so we restrict ourselves to examining it.

2D ANOVA, 750 Points

The first case we examine is the 2d ANOVA data with 750 points per dataset. This data is fairly easy to cluster, and many of the methods had no problem distinguishing the correct number of clusters.

$K_{\text{true}} \downarrow$ / $\hat{K} \rightarrow$	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
2	**1**	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·
3	·	**1**	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·
4	·	·	**1**	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·
5	·	·	·	**1**	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·
6	·	·	·	·	**1**	·	·	·	·	·	·	·	·	·	·	·	·	·	·
7	·	·	·	·	·	**1**	·	·	·	·	·	·	·	·	·	·	·	·	·
8	·	·	·	·	·	·	**1**	·	·	·	·	·	·	·	·	·	·	·	·
9	·	·	·	·	·	·	·	**1**	·	·	·	·	·	·	·	·	·	·	·
10	·	·	·	·	·	·	·	·	**1**	·	·	·	·	·	·	·	·	·	·
11	·	·	·	·	·	·	·	·	·	**.98**	.02	·	·	·	·	·	·	·	·
12	·	·	·	·	·	·	·	·	·	.03	**.95**	.01	.01	·	·	·	·	·	·
13	·	·	·	·	·	·	·	·	·	·	.13	**.84**	.03	·	·	·	·	·	·
14	·	·	·	·	·	·	·	·	·	·	·	.11	**.78**	.11	·	·	·	·	·
15	·	·	·	·	·	·	·	·	·	·	·	.01	.11	**.75**	.13	·	·	·	·
16	·	·	·	·	·	·	·	·	·	·	·	·	.01	.25	**.57**	.17	·	·	·

(a) APW-RC

$K_{\text{true}} \downarrow$ / $\hat{K} \rightarrow$	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
2	**1**	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·
3	.05	**.94**	.01	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·
4	.02	.02	**.89**	.03	.02	.01	·	·	·	·	·	·	·	·	·	·	·	·	.01
5	·	·	.03	**.97**	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·
6	·	·	·	.01	**.99**	·	·	·	·	·	·	·	·	·	·	·	·	·	·
7	·	·	·	·	.03	**.97**	·	·	·	·	·	·	·	·	·	·	·	·	·
8	·	·	·	·	·	.01	**.98**	.01	·	·	·	·	·	·	·	·	·	·	·
9	·	·	·	·	·	·	.03	**.97**	·	·	·	·	·	·	·	·	·	·	·
10	·	·	·	·	·	·	·	.01	**.99**	·	·	·	·	·	·	·	·	·	·
11	·	·	·	·	·	·	·	·	.06	**.92**	.02	·	·	·	·	·	·	·	·
12	·	·	·	·	·	·	·	·	.01	.04	**.91**	.02	.02	·	·	·	·	·	·
13	·	·	·	·	·	·	·	·	·	·	.09	**.82**	.09	·	·	·	·	·	·
14	·	·	·	·	·	·	·	·	·	·	·	.05	**.76**	.18	.01	·	·	·	·
15	·	·	·	·	·	·	·	·	·	·	·	·	.10	**.71**	.18	.01	·	·	·
16	·	·	·	·	·	·	·	·	·	·	·	·	·	.17	**.56**	.25	.02	·	·

(b) APW-Perm

Table 5.6: Sampling distribution of APW-RC and APW-Perm on 2D ANOVA data with 750 points using StdBeforeBest.

In Table (5.6), we compare APW with two different baselines. The first is the reclustered uniform distribution on the PCA bounding box of the data. Recall that this is the same baseline that the gap statistic uses. In lower

dimensions, and with small sample sizes, this seems to work better than the other two baselines. Possibly, this is because the structure in the original data has less influence on this baseline than using a uniform distribution but recycling the cluster centers or simply permuting the distance matrix. However, note that in only one case the prediction is far wrong from the truth.

$K_{\text{true}} \downarrow$ / $\hat{K} \rightarrow$	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
2	**1**	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·
3	.12	**.88**	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·
4	.19	·	**.81**	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·
5	.10	.03	·	**.87**	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·
6	.15	.02	·	·	**.83**	·	·	·	·	·	·	·	·	·	·	·	·	·	·
7	.13	.03	.01	·	·	**.83**	·	·	·	·	·	·	·	·	·	·	·	·	·
8	.14	.05	·	·	·	·	**.81**	·	·	·	·	·	·	·	·	·	·	·	·
9	.15	.01	·	·	·	·	·	**.84**	·	·	·	·	·	·	·	·	·	·	·
10	.10	.09	.06	.01	·	·	·	·	**.74**	·	·	·	·	·	·	·	·	·	·
11	.16	.09	.06	.04	·	·	·	·	·	**.65**	·	·	·	·	·	·	·	·	·
12	.16	.08	.10	.02	.01	·	.01	·	·	·	**.61**	.01	·	·	·	·	·	·	·
13	.17	.19	.05	.09	.03	.01	.01	·	·	·	.03	**.39**	.03	·	·	·	·	·	·
14	.23	.09	.10	.05	.05	.03	·	·	·	·	·	.01	**.31**	.13	·	·	·	·	·
15	.34	.13	.07	.03	.05	.04	.01	·	·	·	·	.01	·	**.23**	.09	·	·	·	·
16	.27	.21	.13	.06	.04	·	·	·	·	·	·	·	·	.02	**.17**	.10	·	·	·

Table 5.7: Sampling distribution of SS-Draws-VI on 2D ANOVA data with 750 points.

In contrast, the prediction distribution of the data perturbation method using SS-Draws and a Variation of Information index, which we show in Table (5.7), tended to be way off if it got the result wrong. This indicates there are effects in the data that can significantly confuse this method. While it is likely that the data perturbation methods could be further tweaked to give substantially better results, such a step would require careful thought when the ground truth is not known.

On this particular set of data, the gap statistic (Table (5.8)), where the prediction was done using StdBeforeBestrule does remarkably well, matched in this case only by APW-RC. However, the original rule of the gap statistic performs horribly.

$K_{\text{true}} \downarrow$ / $\hat{K} \rightarrow$	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
2	**1**	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·
3	·	**1**	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·
4	·	·	**1**	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·
5	.01	·	·	**.99**	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·
6	.05	.32	·	·	**.63**	·	·	·	·	·	·	·	·	·	·	·	·	·	·
7	.02	.48	·	·	·	**.50**	·	·	·	·	·	·	·	·	·	·	·	·	·
8	.10	.52	·	·	·	·	**.38**	·	·	·	·	·	·	·	·	·	·	·	·
9	.12	.56	.02	·	·	·	·	**.30**	·	·	·	·	·	·	·	·	·	·	·
10	.07	.65	.05	.01	·	·	·	·	**.22**	·	·	·	·	·	·	·	·	·	·
11	.11	.63	.03	.02	·	·	·	·	·	**.21**	·	·	·	·	·	·	·	·	·
12	.18	.59	.02	.04	.01	·	·	·	·	.01	**.15**	·	·	·	·	·	·	·	·
13	.12	.67	.04	.04	.01	.01	·	·	·	·	·	**.11**	·	·	·	·	·	·	·
14	.17	.68	·	.05	.01	.03	·	·	·	·	·	.03	**.03**	·	·	·	·	·	·
15	.24	.66	.01	.03	·	.03	·	·	·	·	·	·	.01	**.02**	·	·	·	·	·
16	.21	.66	.03	.04	·	·	·	·	.01	·	·	·	.01	.04	·	·	·	·	·

(a) Gap statistic.

$K_{\text{true}} \downarrow$ / $\hat{K} \rightarrow$	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
2	**1**	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·
3	·	**1**	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·
4	·	·	**1**	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·
5	·	·	·	**1**	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·
6	·	·	·	·	**1**	·	·	·	·	·	·	·	·	·	·	·	·	·	·
7	·	·	·	·	·	**1**	·	·	·	·	·	·	·	·	·	·	·	·	·
8	·	·	·	·	·	·	**1**	·	·	·	·	·	·	·	·	·	·	·	·
9	·	·	·	·	·	·	·	**1**	·	·	·	·	·	·	·	·	·	·	·
10	·	·	·	·	·	·	·	·	**1**	·	·	·	·	·	·	·	·	·	·
11	·	·	·	·	·	·	·	·	·	**.98**	.02	·	·	·	·	·	·	·	·
12	·	·	·	·	·	·	·	·	·	·	**.95**	.04	.01	·	·	·	·	·	·
13	·	·	·	·	·	·	·	·	·	·	·	**.87**	.13	·	·	·	·	·	·
14	·	·	·	·	·	·	·	·	·	·	·	·	**.79**	.19	.02	·	·	·	·
15	·	·	·	·	·	·	·	·	·	·	·	·	.01	**.76**	.22	.01	·	·	·
16	·	·	·	·	·	·	·	·	·	·	·	·	·	.01	**.57**	.36	.06	·	·

(b) Gap statistic, prediction with StdBeforeBest.

Table 5.8: Sampling distributions of the Gap statistic using two prediction rules on 2D ANOVA data with 750 points.

2D ANOVA, 100 Points

There are several features of the ANOVA data with a lower sample size (100 points) that are noteworthy. The first is that one flavor of the gap statistic is the winner in 3 our of 5 of the cases. More specifically, the gap using StdBeforeBest seems to do the best out of all the methods in 2 and 10 dimensions but gets steadily worse; the regular Gap improves, beating all other methods in 100 dimensions but performing poorly on 2 dimensional

data. The other interesting feature is that, contrary to the norm, APW-RC-Best beats APW-RC. In the following tables, we discuss these results in more detail. In addition, we describe this effect and furthermore compare APW-RC and APW-RC-Best to APW-Perm.

$K_{\text{true}} \downarrow$ / $\hat{K} \rightarrow$	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
2	**1**	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·
3	·	**1**	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·
4	·	·	**1**	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·
5	·	·	.04	**.96**	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·
6	.02	·	.02	.06	**.90**	·	·	·	·	·	·	·	·	·	·	·	·	·	·
7	·	·	.02	.04	.18	**.76**	·	·	·	·	·	·	·	·	·	·	·	·	·
8	·	·	·	·	.06	.28	**.66**	·	·	·	·	·	·	·	·	·	·	·	·
9	·	·	·	·	·	·	.54	**.46**	·	·	·	·	·	·	·	·	·	·	·
10	·	·	·	.02	·	·	.14	.38	**.44**	.02	·	·	·	·	·	·	·	·	·
11	·	·	·	·	·	·	.04	.24	.46	**.26**	·	·	·	·	·	·	·	·	·
12	·	·	·	·	·	·	·	.06	.16	.34	**.36**	.08	·	·	·	·	·	·	·
13	·	·	·	·	·	·	.02	.04	.04	.34	.32	**.10**	.12	.02	·	·	·	·	·
14	·	·	·	·	·	·	·	·	·	.10	.38	.34	**.14**	.04	·	·	·	·	·
15	·	·	.02	·	·	·	·	·	·	.08	.12	.40	.22	**.10**	.04	.02	·	·	·
16	·	·	·	·	·	·	·	·	·	·	.02	.06	.40	.24	**.12**	.08	.06	·	.02

(a) APW-RC, prediction using StdBeforeBest.

$K_{\text{true}} \downarrow$ / $\hat{K} \rightarrow$	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
2	**1**	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·
3	·	**1**	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·
4	·	·	**1**	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·
5	·	·	·	**.98**	.02	·	·	·	·	·	·	·	·	·	·	·	·	·	·
6	·	·	·	·	**1**	·	·	·	·	·	·	·	·	·	·	·	·	·	·
7	·	·	·	·	.04	**.94**	.02	·	·	·	·	·	·	·	·	·	·	·	·
8	·	·	·	·	.02	.02	**.92**	.04	·	·	·	·	·	·	·	·	·	·	·
9	·	·	·	·	·	·	.06	**.94**	·	·	·	·	·	·	·	·	·	·	·
10	·	·	·	·	·	·	.02	.14	**.76**	.08	·	·	·	·	·	·	·	·	·
11	·	·	·	·	·	·	·	.06	.18	**.58**	.16	.02	·	·	·	·	·	·	·
12	·	·	·	·	·	·	·	·	·	.16	**.50**	.24	.08	.02	·	·	·	·	·
13	·	·	·	·	·	·	·	·	·	.04	.22	**.34**	.20	.12	.04	.02	·	·	.02
14	·	·	·	·	·	·	·	·	·	·	.08	.28	**.20**	.30	.12	.02	·	·	·
15	·	·	·	·	·	·	·	·	·	.02	.02	.12	.20	**.18**	.22	.18	.04	.02	·
16	·	·	·	·	·	·	·	·	·	·	·	.04	.10	.14	**.28**	.14	.10	.14	.06

(b) APW-RC, prediction using BestK.

Table 5.9: Sampling distribution of APW-RC on 2D ANOVA data with 100 points.

Table (5.9) compares two methods of prediction using the best performing of all our methods. As can be seen by comparing Table (5.9a) and Table (5.9b), APW-RC, with the StdBeforeBest rule, tends to choose one or two

$K_{\text{true}} \downarrow$ / $\hat{K} \rightarrow$	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
2	**.88**	.02	·	·	·	·	·	·	·	·	·	·	·	.04	.02	.02	·	·	.02
3	·	**.46**	.04	.02	·	·	·	.02	.02	·	.02	.04	.06	.02	.06	.12	.06	.04	.02
4	·	·	**.48**	.02	·	.02	·	·	.02	.04	.06	.04	.04	.08	.10	·	.04	.04	.02
5	·	·	.08	**.56**	.04	.04	.04	·	.02	.02	·	·	·	.02	.02	.06	.06	.04	·
6	·	·	·	.06	**.70**	.04	·	·	.04	·	.02	·	.06	.02	·	·	.02	·	.04
7	·	·	·	.02	.10	**.70**	.14	.02	·	·	.02	·	·	·	·	·	·	·	·
8	·	·	·	·	·	.12	**.70**	.04	.08	·	.02	·	·	·	·	·	.02	.02	·
9	·	·	·	·	·	.02	.32	**.52**	.02	.02	.02	.02	·	·	.02	.02	·	.02	·
10	·	·	·	·	·	·	.02	.36	**.46**	.08	·	.02	·	·	·	.02	.04	·	·
11	·	·	·	·	·	·	.02	.04	.28	**.44**	.10	.02	.02	·	.04	·	.04	·	·
12	·	·	·	·	·	·	·	.02	·	.30	**.34**	.14	.08	.06	·	.04	.02	·	·
13	·	·	·	·	·	·	·	·	·	.08	.22	**.26**	.16	.12	.06	.08	·	.02	·
14	·	·	·	·	·	·	·	·	·	.02	.12	.28	**.18**	.16	.12	.04	.04	.02	.02
15	·	·	·	·	·	·	·	·	·	·	.02	.14	.14	**.30**	.24	.14	.02	·	·
16	·	·	·	·	·	·	·	·	·	·	·	·	.06	.20	**.30**	.20	.14	.04	.06

Table 5.10: Sampling distribution of APW-Perm, StdBeforeBest, on 2D ANOVA data with 100 points.

k's too small. This could be because the sample size does not provide a reliable baseline, but the exact cause is difficult to determine.

Regardless, we note that the number of times APW-RC is significantly far off is still quite small. APW-Perm, however, tended to greatly overestimate the number of clusters when it got it wrong, as seen in Table (5.10). Even though APW-Perm achieves a reasonable score – 6.24 compared to 6.92 for APW-RC – its predictions are significantly less reliable. This is likely due to the permutation being unable to mask the significant structure present in the original distance matrix. This would be more of a problem on this type of data than any other.

The results for the gap statistic are shown in Table (5.11). One noteworthy fact is that the prediction rule proposed as part of the gap statistic performs horribly in this case, failing to get any correct on some of the datasets with many clusters and consistently underestimating the number of clusters in the data. However, using the gap statistic with StdBeforeBest as the prediction rule works quite well. Why this is the case is more difficult to say, but it is something to keep in mind when using the gap statistic in practice.

$K_{\text{true}} \downarrow$ / $\hat{K} \rightarrow$	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
2	**1**	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·
3	·	**1**	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·
4	.08	·	**.92**	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·
5	.18	.10	·	**.72**	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·
6	.30	.20	.02	·	**.48**	·	·	·	·	·	·	·	·	·	·	·	·	·	·
7	.26	.36	.04	.02	.02	**.30**	·	·	·	·	·	·	·	·	·	·	·	·	·
8	.46	.42	.06	.02	·	·	**.04**	·	·	·	·	·	·	·	·	·	·	·	·
9	.50	.38	.06	.04	·	·	·	**.02**	·	·	·	·	·	·	·	·	·	·	·
10	.52	.34	.04	.02	.04	·	·	.04	·	·	·	·	·	·	·	·	·	·	·
11	.42	.46	.04	.02	.02	·	·	·	.02	**.02**	·	·	·	·	·	·	·	·	·
12	.48	.38	.06	.06	·	·	·	·	·	·	**.02**	·	·	·	·	·	·	·	·
13	.58	.38	.02	·	.02	·	·	·	·	·	·	·	·	·	·	·	·	·	·
14	.62	.36	·	·	.02	·	·	·	·	·	·	·	·	·	·	·	·	·	·
15	.68	.28	.04	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·
16	.68	.32	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·

(a) Gap statistic.

$K_{\text{true}} \downarrow$ / $\hat{K} \rightarrow$	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
2	**1**	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·
3	·	**1**	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·
4	·	·	**1**	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·
5	·	·	·	**1**	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·
6	·	·	·	·	**1**	·	·	·	·	·	·	·	·	·	·	·	·	·	·
7	·	·	·	·	.06	**.92**	.02	·	·	·	·	·	·	·	·	·	·	·	·
8	·	·	·	·	·	.10	**.90**	·	·	·	·	·	·	·	·	·	·	·	·
9	·	·	·	·	·	·	.08	**.90**	.02	·	·	·	·	·	·	·	·	·	·
10	·	·	·	·	·	·	.04	.10	**.76**	.10	·	·	·	·	·	·	·	·	·
11	·	·	·	·	·	·	·	·	.28	**.62**	.10	·	·	·	·	·	·	·	·
12	·	·	·	·	·	·	·	·	.02	.16	**.48**	.28	.02	.02	.02	·	·	·	·
13	·	·	·	·	·	·	·	·	·	.04	.32	**.32**	.24	.06	·	.02	·	·	·
14	·	·	·	·	·	·	·	·	·	·	.06	.24	**.20**	.36	.14	·	·	·	·
15	·	·	·	·	·	·	·	·	·	·	·	.04	.20	**.22**	.36	.12	.06	·	·
16	·	·	·	·	·	·	·	·	·	·	·	·	.08	.24	**.26**	.16	.14	.12	·

(b) Gap statistic using StdBeforeBest for prediction.

Table 5.11: Sampling distribution of the gap statistic on 2D ANOVA data with 100 points using the StdBeforeBest prediction rule.

2D Shaped

On the 2 dimensional shaped data, the most interesting cases occur in the low sample size datasets. We here examine a sampling of the most interesting cases.

With APW-RC (Table (5.12)), prediction using with StdBeforeBest performs better than prediction using BestK, which, in this case, tends to overestimate the number of clusters. This is contrary to the analogous ANOVA

$K_{\text{true}} \downarrow$ / $\hat{K} \rightarrow$	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
2	**.86**	.10	.04	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·
3	·	**.88**	.12	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·
4	·	·	**.96**	.02	.02	·	·	·	·	·	·	·	·	·	·	·	·	·	·
5	·	·	·	**.86**	.14	·	·	·	·	·	·	·	·	·	·	·	·	·	·
6	.02	·	·	·	**.82**	.10	.04	.02	·	·	·	·	·	·	·	·	·	·	·
7	·	·	·	.02	·	**.76**	.18	.04	·	·	·	·	·	·	·	·	·	·	·
8	·	·	·	·	·	.02	**.86**	.10	·	·	.02	·	·	·	·	·	·	·	·
9	·	·	·	·	·	·	.04	**.64**	.26	.04	.02	·	·	·	·	·	·	·	·
10	·	·	·	·	·	·	·	.02	**.64**	.20	.10	.04	·	·	·	·	·	·	·
11	·	·	·	·	·	·	·	·	.06	**.46**	.28	.16	.04	·	·	·	·	·	·
12	·	·	·	·	·	·	·	·	·	.14	**.30**	.22	.30	.02	.02	·	·	·	·
13	.02	·	·	·	·	·	·	·	·	·	.02	**.36**	.32	.10	.12	.04	.02	·	·
14	·	·	·	·	·	·	·	·	·	·	.06	.02	**.28**	.26	.18	.12	.04	.02	.02
15	·	·	·	·	·	·	·	·	·	·	·	·	.10	**.16**	.34	.22	.14	.02	.02
16	.02	·	·	·	·	·	·	·	·	·	·	·	·	.08	**.08**	.18	.20	.28	.16

(a) APW-RC using BestK for prediction.

$K_{\text{true}} \downarrow$ / $\hat{K} \rightarrow$	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
2	**.94**	.06	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·
3	·	**.94**	.06	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·
4	·	·	**.98**	.02	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·
5	·	·	.02	**.96**	.02	·	·	·	·	·	·	·	·	·	·	·	·	·	·
6	.02	·	·	.02	**.94**	.02	·	·	·	·	·	·	·	·	·	·	·	·	·
7	·	·	·	.02	.10	**.84**	.04	·	·	·	·	·	·	·	·	·	·	·	·
8	·	·	·	·	·	.08	**.92**	·	·	·	·	·	·	·	·	·	·	·	·
9	·	·	·	·	·	·	.16	**.78**	.06	·	·	·	·	·	·	·	·	·	·
10	·	·	·	·	·	·	.04	.24	**.66**	.06	·	·	·	·	·	·	·	·	·
11	·	·	·	·	·	·	·	.06	.44	**.44**	.06	·	·	·	·	·	·	·	·
12	·	·	·	·	·	·	·	.02	.06	.36	**.40**	.12	.04	·	·	·	·	·	·
13	.02	·	·	·	·	·	·	·	·	.06	.18	**.40**	.24	.08	.02	·	·	·	·
14	·	·	·	·	·	·	·	·	·	·	.08	.32	**.32**	.14	.08	.06	·	·	·
15	·	·	·	·	·	·	·	·	·	·	·	.06	.26	**.18**	.34	.10	.04	·	.02
16	.02	·	·	·	·	·	·	·	·	·	·	.02	.08	.20	**.22**	.28	.08	.06	.04

(b) APW-RC using StdBeforeBest for prediction.

Table 5.12: Sampling distributions of APW-RC with two different predictors on 2D ANOVA data with 100 points.

results. In this case, using the StdBeforeBest prediction rule is quite justified; not only is the predictor more accurate, but the method does not tend to underestimate.

Tables (5.13a) and (5.13b) illustrate the accuracy of using as the baseline distance matrix a permutation of the original. In the low sample size case, the effects of the original structure would be harder to mask with the permutation, and this is illustrated here. While both tend to overestimate

$K_{\text{true}} \downarrow$ / $\hat{K} \rightarrow$	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
2	**.30**	.06	.02	.02	.02	·	·	·	.04	.02	·	.04	.08	.04	.08	.10	.08	.04	.06
3	·	**.20**	.10	.02	.02	·	.02	.04	.04	.04	.06	.02	.12	.06	.06	.08	.06	.02	.04
4	·	·	**.38**	.04	.06	·	·	.04	.02	.02	.06	.08	.02	.12	·	.06	.08	.02	·
5	·	·	·	**.34**	.20	.08	.02	.02	.06	.02	.04	.02	·	.06	.08	·	.04	.02	·
6	·	·	·	·	**.32**	.14	.08	.04	·	.04	.04	.04	.12	.02	.06	.04	.04	·	.02
7	·	·	·	·	.02	**.52**	.14	.08	.04	.02	.06	.02	·	·	·	.06	.02	.02	·
8	·	·	·	·	·	.04	**.52**	.12	.02	.06	.02	.02	.06	.02	.02	.04	.02	.02	.02
9	·	·	·	·	·	·	.10	**.48**	.10	.02	.02	.06	.06	.04	.02	.06	.04	·	·
10	·	·	·	·	·	·	·	.02	**.48**	.24	.06	.06	.04	.02	·	.06	.02	·	·
11	·	·	·	·	·	·	.02	·	.10	**.30**	.26	.10	.10	·	.04	.02	.06	·	·
12	·	·	·	·	·	·	·	·	·	.04	**.24**	.16	.20	.06	.12	.06	.06	.04	.02
13	·	·	·	·	·	·	·	·	·	·	.08	**.28**	.26	.12	.06	.08	.04	.02	.06
14	·	·	·	·	·	·	·	·	·	·	·	.04	**.32**	.26	.20	.10	.04	·	.04
15	·	·	·	·	·	·	·	·	·	·	·	·	.04	**.18**	.38	.22	.14	.02	.02
16	·	·	·	·	·	·	·	·	·	·	·	·	.02	.10	**.20**	.30	.20	.16	.02

(a) APW-Perm with 100 points per dataset

$K_{\text{true}} \downarrow$ / $\hat{K} \rightarrow$	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
2	**.87**	.06	·	.01	·	·	·	·	·	·	·	.01	.02	.02	.01	·	·	·	·
3	·	**.59**	.17	.06	.04	.02	.02	.01	.02	.01	·	·	·	.01	.01	.02	.01	·	.01
4	·	·	**.64**	.09	.06	·	·	.02	.01	·	.03	.03	.01	.02	.02	.02	.02	.01	.02
5	·	·	·	**.76**	.18	.03	.01	·	.01	·	·	.01	·	·	·	·	·	·	·
6	·	·	·	·	**.74**	.16	.04	.01	.03	·	.01	·	·	·	·	·	·	.01	·
7	·	·	·	·	·	**.83**	.11	.03	.01	.01	.01	·	·	·	·	·	·	·	·
8	·	·	·	·	·	.02	**.79**	.13	.03	.01	.01	·	·	·	.01	·	·	·	·
9	·	·	·	·	·	·	·	**.84**	.09	.04	.03	·	·	·	·	·	·	·	·
10	·	·	·	·	·	·	·	·	**.79**	.16	.01	.02	.01	·	·	·	.01	·	·
11	·	·	·	·	·	·	·	·	.02	**.79**	.14	.03	.02	·	·	·	·	·	·
12	·	·	·	·	·	·	·	·	·	.01	**.72**	.22	.03	.01	·	.01	·	·	·
13	·	·	·	·	·	·	·	·	·	·	.01	**.55**	.34	.05	.03	.01	.01	·	·
14	·	·	·	·	·	·	·	·	·	·	·	.01	**.53**	.30	.07	.06	.02	.01	·
15	·	·	·	·	·	·	·	·	·	·	·	·	.01	**.37**	.47	.10	.03	·	.02
16	·	·	·	·	·	·	·	·	·	·	·	·	·	.01	**.34**	.37	.18	.07	.03

(b) APW-Perm with 750 points per dataset

Table 5.13: Sampling distribution of APW-Perm, StdBeforeBest, on 2D Shaped data with various sample sizes.

the number of clusters, the accuracy when the sample size is higher is substantially better.

Again, we give the sampling distribution of SS-Draws-VI, as this represents the other data perturbation methods. We present the results for the lower sample size in Table (5.14); they are indicative of the larger sample size as well. As in the ANOVA case, it tends to significantly underestimate the number of clusters.

$\hat{K} \rightarrow$ $K_{\text{true}} \downarrow$	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
2	**.98**	.02	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·
3	.06	**.92**	.02	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·
4	.08	·	**.86**	.06	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·
5	.10	.04	.08	**.72**	.02	.04	·	·	·	·	·	·	·	·	·	·	·	·	·
6	.08	.04	.08	.04	**.68**	.08	·	·	·	·	·	·	·	·	·	·	·	·	·
7	.06	.04	.02	.04	.16	**.64**	.04	·	·	·	·	·	·	·	·	·	·	·	·
8	.10	.12	.02	.04	.02	.04	**.48**	.14	.04	·	·	·	·	·	·	·	·	·	·
9	.12	.10	.02	.02	·	·	.08	**.52**	.10	.04	·	·	·	·	·	·	·	·	·
10	.10	.10	.08	.02	·	·	.08	.10	**.32**	.16	.04	·	·	·	·	·	·	·	·
11	.08	.18	.02	.02	.06	·	.02	.06	.18	**.30**	.08	·	·	·	·	·	·	·	·
12	.14	.12	.02	.06	·	.02	.02	·	.12	.18	**.18**	.12	.02	·	·	·	·	·	·
13	.10	.06	.04	.04	·	·	·	.02	.02	.02	.18	**.24**	.16	.10	.02	·	·	·	·
14	.14	.10	.04	.08	.02	·	·	·	.02	.04	.14	.12	**.14**	.16	·	·	·	·	·
15	.12	.08	.06	·	·	.02	.02	·	·	.02	.04	.12	.12	**.22**	.12	.06	·	·	·
16	.08	.06	.02	.06	·	.02	.02	·	.02	·	.04	.02	.24	.22	**.14**	.02	.04	·	·

Table 5.14: Sampling distribution of SS-Draws-VI on 2D Shaped data with 100 points.

On this type of data, neither of the two prediction rules tested makes the gap statistic perform well (Table (5.15)). If the proposed rule is used, the method tends to underestimate the number of clusters. If, however, StdBeforeBest is used for predicting K, the method tends to get more correct but overestimates the cases it gets wrong. Perhaps more sophisticated predictors are possible, but selecting other rules would be a difficult task.

100D ANOVA

For the high dimensional case, we present the prediction distribution tables for the ANOVA and shaped data with 100 points. In general, the results we note here apply also to the 750 sample case, and we didn't notice any substantial difference between the results. For conciseness, we omit them.

In the higher dimensions, the permutation based methods performed significantly better than in lower dimensions (Table (5.16)). This is likely due to the fact that lower dimensional distributions tend to have a less homogeneous distribution of distance measures, which would make the original structure harder to disguise using permutations.

In comparing APW-Perm with APW-RC, we note from Table (5.17) that the sampling distribution for APW-Perm-Best and APW-RC-Best both

$K_{\text{true}} \downarrow$ / $\hat{K} \rightarrow$	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
2	**.50**	.10	.10	.02	.04	.04	.02	.04	.02	·	.02	.02	.02	·	.04	·	.02	·	·
3	·	**.64**	.02	·	.02	.02	.04	.04	.02	.02	.02	.04	.06	.02	.04	·	·	·	·
4	·	·	**.56**	.06	·	.02	.04	.02	.06	.02	.04	·	.04	.06	·	.04	.02	·	.02
5	·	·	·	**.60**	.06	.04	.02	.06	.02	.02	·	.04	.02	.02	.02	.02	.02	.04	·
6	·	·	·	·	**.40**	.14	·	.10	.06	.06	.04	.02	.04	.06	.02	.02	.02	·	.02
7	·	·	·	·	·	**.62**	.10	·	.02	.04	.02	.02	.02	.04	·	.08	.02	.02	·
8	·	·	·	·	·	·	**.54**	.22	.08	.02	.02	·	.04	·	·	.02	.04	.02	·
9	·	·	·	·	·	·	·	**.52**	.14	.08	.04	.02	.02	.02	.02	.02	.10	.02	·
10	·	·	·	·	·	·	·	·	**.46**	.20	.18	.08	.02	.02	·	·	·	.04	·
11	·	·	·	·	·	·	·	·	.02	**.38**	.26	.10	.12	·	·	.04	.04	.04	·
12	·	·	·	·	·	·	·	·	·	.04	**.26**	.28	.12	.08	.08	.06	.06	.02	·
13	·	·	·	·	·	·	·	·	·	·	·	**.26**	.30	.22	.06	.06	.10	·	·
14	·	·	·	·	·	·	·	·	·	·	·	.02	**.26**	.32	.08	.12	.08	.08	.04
15	·	·	·	·	·	·	·	·	·	·	·	·	.02	**.12**	.48	.22	.14	.02	·
16	·	·	·	·	·	·	·	·	·	·	·	·	·	.04	**.10**	.26	.26	.26	.08

(a) Gap statistic.

$K_{\text{true}} \downarrow$ / $\hat{K} \rightarrow$	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
2	**.80**	.16	.02	·	·	.02	·	·	·	·	·	·	·	·	·	·	·	·	·
3	·	**.90**	.06	.04	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·
4	.02	·	**.92**	.06	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·
5	.02	·	·	**.90**	.08	·	·	·	·	·	·	·	·	·	·	·	·	·	·
6	.16	.10	.02	·	**.62**	.10	·	·	·	·	·	·	·	·	·	·	·	·	·
7	.14	.28	.04	·	·	**.50**	.04	·	·	·	·	·	·	·	·	·	·	·	·
8	.32	.26	.02	·	·	.02	**.38**	·	·	·	·	·	·	·	·	·	·	·	·
9	.34	.32	.10	·	·	·	.04	**.20**	·	·	·	·	·	·	·	·	·	·	·
10	.30	.38	.04	.02	.02	.02	.02	.08	**.12**	·	·	·	·	·	·	·	·	·	·
11	.40	.44	.04	·	·	·	.02	.02	.04	**.04**	·	·	·	·	·	·	·	·	·
12	.58	.24	.12	.04	·	·	.02	·	·	·	·	·	·	·	·	·	·	·	·
13	.56	.26	.10	.04	·	·	·	.02	·	·	·	**.02**	·	·	·	·	·	·	·
14	.52	.40	.04	.02	.02	·	·	·	·	·	·	·	·	·	·	·	·	·	·
15	.68	.30	.02	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·
16	.68	.22	.08	·	·	·	.02	·	·	·	·	·	·	·	·	·	·	·	·

(b) Gap statistic, prediction using StdBeforeBest.

Table 5.15: Sampling distribution on 2D shaped data with 100 points using the gap statistic.

seem to do well. The main thing to note here, however, is that APW-Perm, while achieving less accuracy than APW-RC, tends to get fewer far wrong. On the contrary, APW-RC can significantly underestimate the number of clusters when K_{true} is higher, whereas APW-Perm does not have this problem.

Note, furthermore, that comparison of the same method on two sample sizes can be misleading, as the results are not standardized between the two.

$K_{\text{true}} \downarrow$ / $\hat{K} \rightarrow$	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
2	**1**	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·
3	·	**1**	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·
4	·	·	**1**	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·
5	·	·	·	**1**	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·
6	·	·	·	·	**1**	·	·	·	·	·	·	·	·	·	·	·	·	·	·
7	·	·	·	·	·	**.96**	.04	·	·	·	·	·	·	·	·	·	·	·	·
8	·	·	·	·	·	·	**1**	·	·	·	·	·	·	·	·	·	·	·	·
9	·	·	·	·	·	·	·	**.92**	.08	·	·	·	·	·	·	·	·	·	·
10	·	·	·	·	·	·	·	.04	**.80**	.14	·	.02	·	·	·	·	·	·	·
11	·	·	·	·	·	·	·	.02	.02	**.56**	.26	.08	.04	·	·	·	·	.02	·
12	·	·	·	·	·	·	·	·	·	.06	**.34**	.48	.08	.02	·	.02	·	·	·
13	·	·	·	·	·	·	·	·	·	·	·	**.44**	.32	.12	.04	.04	.04	·	·
14	·	·	·	·	·	·	·	·	·	·	.02	.04	**.14**	.36	.26	.12	.06	·	·
15	·	·	·	·	·	·	·	·	·	·	·	·	·	**.12**	.32	.28	.14	.12	.02
16	·	·	·	·	·	·	·	·	·	·	·	·	·	.04	**.22**	.18	.24	.24	.08

(a) APW-Perm with 100 points per dataset

$K_{\text{true}} \downarrow$ / $\hat{K} \rightarrow$	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
2	**1**	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·
3	·	**1**	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·
4	·	·	**1**	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·
5	·	·	·	**1**	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·
6	·	·	·	·	**1**	·	·	·	·	·	·	·	·	·	·	·	·	·	·
7	·	·	·	·	·	**1**	·	·	·	·	·	·	·	·	·	·	·	·	·
8	·	·	·	·	·	·	**.99**	.01	·	·	·	·	·	·	·	·	·	·	·
9	·	·	·	·	·	·	.01	**.94**	.04	·	.01	·	·	·	·	·	·	·	·
10	·	·	·	·	·	·	·	.03	**.83**	.14	·	·	·	·	·	·	·	·	·
11	·	·	·	·	·	·	·	·	.03	**.76**	.21	·	·	·	·	·	·	·	·
12	·	·	·	·	·	·	·	·	.01	.02	**.62**	.34	.01	·	·	·	·	·	·
13	·	·	·	·	·	·	·	·	·	·	.07	**.60**	.30	.02	.01	·	·	·	·
14	·	·	·	·	·	·	·	·	·	·	.02	.12	**.42**	.42	.02	·	·	·	·
15	·	·	·	·	·	·	·	·	·	·	·	.02	.12	**.40**	.34	.10	.02	·	·
16	·	·	·	·	·	·	·	·	·	·	·	·	.07	.22	**.37**	.23	.09	.01	.01

(b) APW-Perm with 750 points per dataset

Table 5.16: Sampling distribution of APW-Perm, StdBeforeBest, on 100D ANOVA data with various sample sizes.

Thus the comparisons between corresponding values in the histograms are not justified, but qualitative patterns in the histograms are. We are focusing here on the latter.

Also, although APW-RC performs better on both dataset sizes, the differences between APW-RC and APW-Perm are more pronounced for lower sample sizes, again consistent with what we'd expect. Note, however, that normally APW-Perm tends to perform best in higher dimensions.

$K_{\text{true}} \downarrow$ / $\hat{K} \rightarrow$	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
2	**1**	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·
3	·	**1**	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·
4	·	·	**1**	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·
5	·	·	·	**1**	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·
6	·	·	·	·	**1**	·	·	·	·	·	·	·	·	·	·	·	·	·	·
7	·	·	·	·	.04	**.96**	·	·	·	·	·	·	·	·	·	·	·	·	·
8	·	·	·	·	·	·	**1**	·	·	·	·	·	·	·	·	·	·	·	·
9	·	·	·	·	.02	·	.06	**.92**	·	·	·	·	·	·	·	·	·	·	·
10	·	·	·	·	·	·	·	.20	**.80**	·	·	·	·	·	·	·	·	·	·
11	.02	·	·	·	·	·	·	·	.38	**.56**	.04	·	·	·	·	·	·	·	·
12	·	·	·	·	·	·	·	.02	.08	.50	**.34**	.06	·	·	·	·	·	·	·
13	.02	·	·	·	·	·	·	·	.02	.12	.34	**.44**	.06	·	·	·	·	·	·
14	.12	.02	·	·	·	·	·	·	·	.04	.16	.42	**.16**	.08	·	·	·	·	·
15	.22	·	·	·	·	·	·	·	·	·	.04	.10	.42	**.10**	.08	.04	·	·	·
16	.16	·	·	·	·	·	·	·	·	·	.02	.02	.24	.26	**.20**	.08	.02	·	·

(a) APW-RC with 100 points per dataset

$K_{\text{true}} \downarrow$ / $\hat{K} \rightarrow$	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
2	**1**	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·
3	·	**1**	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·
4	·	·	**1**	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·
5	·	·	·	**1**	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·
6	·	·	·	·	**1**	·	·	·	·	·	·	·	·	·	·	·	·	·	·
7	·	·	·	·	·	**1**	·	·	·	·	·	·	·	·	·	·	·	·	·
8	·	·	·	·	·	.01	**.99**	·	·	·	·	·	·	·	·	·	·	·	·
9	·	·	·	·	·	·	.06	**.94**	·	·	·	·	·	·	·	·	·	·	·
10	·	·	·	·	·	·	·	.17	**.83**	·	·	·	·	·	·	·	·	·	·
11	·	·	·	·	·	·	·	.01	.23	**.76**	·	·	·	·	·	·	·	·	·
12	·	·	·	·	·	·	·	·	.02	.36	**.62**	·	·	·	·	·	·	·	·
13	·	·	·	·	·	·	·	·	·	.03	.36	**.60**	.01	·	·	·	·	·	·
14	·	·	·	·	·	·	·	·	·	·	.05	.48	**.42**	.05	·	·	·	·	·
15	·	·	·	·	·	·	·	·	·	·	·	.13	.43	**.40**	.03	.01	·	·	·
16	·	·	·	·	·	·	·	·	·	·	·	.01	.18	.36	**.39**	.05	.01	·	·

(b) APW-RC with 750 points per dataset

Table 5.17: Sampling distribution of APW-RC, StdBeforeBest, on 100D ANOVA data with various sample sizes.

On the 100 dimensional ANOVA data with 100 points, SS-Draws-VI performs fairly well, but the gap statistic beats it. As in other cases, it can also significantly underestimate the number of clusters. The gap statistic using StdBeforeBest, while usually fairly close, tends to overestimate the number of clusters by one or two.

$K_{\text{true}} \downarrow$ / $\hat{K} \rightarrow$	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
2	**1**	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·
3	·	**1**	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·
4	·	·	**1**	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·
5	.02	·	.02	**.96**	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·
6	·	·	·	·	**1**	·	·	·	·	·	·	·	·	·	·	·	·	·	·
7	·	·	·	·	.02	**.98**	·	·	·	·	·	·	·	·	·	·	·	·	·
8	·	·	·	·	·	·	**1**	·	·	·	·	·	·	·	·	·	·	·	·
9	.02	·	·	·	·	·	.12	**.82**	.04	·	·	·	·	·	·	·	·	·	·
10	·	·	·	·	·	·	·	.18	**.72**	.10	·	·	·	·	·	·	·	·	·
11	.02	·	·	·	·	·	·	.02	.20	**.52**	.20	.02	.02	·	·	·	·	·	·
12	·	·	·	·	·	·	·	·	.04	.32	**.36**	.26	.02	·	·	·	·	·	·
13	·	·	·	·	·	·	·	·	.02	.06	.24	**.40**	.26	.02	·	·	·	·	·
14	.02	·	·	·	·	·	·	·	·	·	.10	.26	**.20**	.30	.08	.04	·	·	·
15	·	·	·	·	·	·	·	·	·	·	.02	.10	.38	**.24**	.12	.14	·	·	·
16	.02	·	·	·	·	·	·	·	·	·	·	.02	.12	.28	**.30**	.12	.08	.04	.02

Table 5.18: Sampling distribution of SS-Draws-VI on 100D ANOVA data with 100 points.

100D Shaped

The prediction tables for the 100 dimensional shaped data have many of the same characteristics as the 2 dimensional shaped data. We present here a few highlights. The data perturbation methods had similar distributions to the lower dimensional cases, so we omit them here. Also, most of the noteworthy characteristics we present also apply to the 100 dimensional shaped data with 750 points.

In comparing APW-Perm with APW-RC, we note from Table (5.21) that the sampling distributions for APW-Perm and APW-RC have similar characteristics. Also, although APW-RC performs better on both dataset sizes, the difference is more pronounced for lower sample sizes, again consistent with what we'd expect (Note, however, that normally in shaped distributions in higher dimensions, APW-Perm tends to perform best). It's worthwhile to note that APW-RC tends to underestimate, while APW-Perm tends to overestimate.

The performance of APW-Perm and APW-RC are similar to the 2 dimensional case, with some minor variations. The first variation is the number of cases in which APW-RC was far wrong and significantly underestimated the number of clusters; this was not a problem in 2 dimensions.

$K_{\text{true}} \downarrow$ / $\hat{K} \rightarrow$	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
2	**1**	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·
3	·	**1**	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·
4	·	·	**1**	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·
5	·	·	·	**1**	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·
6	·	·	·	·	**1**	·	·	·	·	·	·	·	·	·	·	·	·	·	·
7	·	·	·	·	.04	**.96**	·	·	·	·	·	·	·	·	·	·	·	·	·
8	·	·	·	·	·	·	**1**	·	·	·	·	·	·	·	·	·	·	·	·
9	·	·	·	·	·	·	.04	**.92**	.04	·	·	·	·	·	·	·	·	·	·
10	·	·	·	·	·	·	·	.16	**.82**	.02	·	·	·	·	·	·	·	·	·
11	·	·	·	·	·	·	·	·	.28	**.56**	.14	.02	·	·	·	·	·	·	·
12	·	·	·	·	·	·	·	·	.04	.28	**.40**	.26	.02	·	·	·	·	·	·
13	·	·	·	·	·	·	·	·	·	.10	.30	**.52**	.04	.04	·	·	·	·	·
14	·	·	·	·	·	·	·	·	·	.02	.22	.44	**.14**	.16	.02	·	·	·	·
15	·	·	·	·	·	·	·	·	·	·	.08	.22	.54	**.10**	.06	·	·	·	·
16	·	·	·	·	·	·	·	·	·	·	.02	.06	.28	.24	**.32**	.08	·	·	·

(a) Gap statistic.

$K_{\text{true}} \downarrow$ / $\hat{K} \rightarrow$	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
2	**.96**	.02	·	.02	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·
3	·	**.90**	.04	.02	.02	·	·	·	.02	·	·	·	·	·	·	·	·	·	·
4	·	·	**.94**	.06	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·
5	·	·	·	**.96**	.02	.02	·	·	·	·	·	·	·	·	·	·	·	·	·
6	·	·	·	·	**.98**	.02	·	·	·	·	·	·	·	·	·	·	·	·	·
7	·	·	·	·	·	**.90**	.08	.02	·	·	·	·	·	·	·	·	·	·	·
8	·	·	·	·	·	·	**.92**	.08	·	·	·	·	·	·	·	·	·	·	·
9	·	·	·	·	·	·	·	**.92**	.08	·	·	·	·	·	·	·	·	·	·
10	·	·	·	·	·	·	·	·	**.80**	.18	.02	·	·	·	·	·	·	·	·
11	·	·	·	·	·	·	·	·	·	**.52**	.36	.10	.02	·	·	·	·	·	·
12	·	·	·	·	·	·	·	·	·	·	**.34**	.52	.12	.02	·	·	·	·	·
13	·	·	·	·	·	·	·	·	·	·	·	**.40**	.32	.16	.12	·	·	·	·
14	·	·	·	·	·	·	·	·	·	·	·	·	**.12**	.44	.26	.10	.06	·	.02
15	·	·	·	·	·	·	·	·	·	·	·	·	·	**.08**	.38	.32	.12	.10	·
16	·	·	·	·	·	·	·	·	·	·	·	·	·	·	**.14**	.16	.36	.28	.06

(b) Gap statistic, prediction using StdBeforeBest.

Table 5.19: Sampling distribution on 100D ANOVA data with 100 points using the gap statistic.

The flip side is the APW-Perm tended, when there were more clusters, to overestimate the number of clusters slightly but was more reliable overall.

The data perturbation method illustrates some of the same patterns as before; it seems to be fairly robust to changes in shape and dimension. The gap statistic, however, performs less well, as as the modeling assumptions it is based on do not hold. While it is still somewhat competitive with our method, its accuracy is still less than that to APW-Perm which has far fewer

$K_{\text{true}} \downarrow$ / $\hat{K} \rightarrow$	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
2	**.95**	.05	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·
3	·	**.91**	.08	.01	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·
4	·	·	**.90**	.08	.02	·	·	·	·	·	·	·	·	·	·	·	·	·	·
5	·	·	·	**.94**	.06	·	·	·	·	·	·	·	·	·	·	·	·	·	·
6	·	.01	·	·	**.94**	.04	·	·	·	·	.01	·	·	·	·	·	·	·	·
7	·	·	·	·	.02	**.93**	.04	·	·	·	·	·	·	·	·	.01	·	·	·
8	·	·	·	·	·	.07	**.85**	.06	.01	·	·	·	·	.01	·	·	·	·	·
9	·	·	·	·	·	.01	.20	**.69**	.09	.01	·	·	·	·	·	·	·	·	·
10	·	·	·	·	·	·	.01	.39	**.46**	.13	.01	·	·	·	·	·	·	·	·
11	.01	.01	·	·	·	·	·	.04	.29	**.46**	.17	.02	·	·	·	·	·	·	·
12	·	·	.01	·	·	·	·	·	.05	.34	**.29**	.26	.03	.01	.01	·	·	·	·
13	·	.01	·	·	·	·	·	·	.02	.12	.33	**.25**	.15	.09	.02	·	·	.01	·
14	.01	.01	·	·	·	·	·	·	·	.04	.15	.27	**.24**	.17	.08	.03	·	·	·
15	.02	.02	·	·	·	·	·	·	·	.01	.11	.24	.14	**.16**	.15	.11	.03	.01	·
16	.01	·	·	·	·	·	·	·	·	·	.02	.08	.29	.24	**.11**	.12	.08	.05	·

(a) APW-RC.

$K_{\text{true}} \downarrow$ / $\hat{K} \rightarrow$	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
2	**.96**	.04	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·
3	·	**.92**	.06	.01	·	·	·	·	·	·	·	·	·	.01	·	·	·	·	·
4	·	·	**.91**	.06	.01	·	·	·	·	·	·	·	·	·	.02	·	·	·	·
5	·	·	·	**.96**	.04	·	·	·	·	·	·	·	·	·	·	·	·	·	·
6	·	·	·	·	**.94**	.04	·	·	·	·	·	.01	·	·	·	·	·	.01	·
7	·	·	·	·	·	**.94**	.04	.01	·	·	·	·	·	·	·	.01	·	·	·
8	·	·	·	·	·	.01	**.85**	.11	.01	.01	·	·	·	.01	·	·	·	·	·
9	·	·	·	·	·	·	.02	**.69**	.24	.04	.01	·	·	·	·	·	·	·	·
10	·	·	·	·	·	·	.01	.06	**.45**	.39	.08	.01	·	·	·	·	·	·	·
11	·	·	·	·	·	·	·	.02	.08	**.44**	.34	.08	.03	·	.01	·	·	·	·
12	·	·	·	·	·	·	·	·	.03	.15	**.29**	.39	.06	.04	.03	·	.01	·	·
13	·	·	·	·	·	·	.01	·	.02	.07	.15	**.23**	.21	.23	.04	·	·	.02	.02
14	·	·	·	·	·	·	·	.01	.01	.05	.09	.19	**.19**	.19	.16	.07	.02	.01	.01
15	·	·	·	·	·	·	·	·	.01	.01	.06	.14	.10	**.17**	.20	.18	.07	.04	.02
16	·	·	·	·	·	·	·	·	·	.01	.03	.08	.17	.17	**.09**	.14	.17	.09	.05

(b) APW-Perm.

Table 5.20: Sampling distribution of APW-RC and APW-Perm, StdBeforeBest, on 100D Shaped data with 750 points.

computational issues.

5.4 Conclusions

In this chapter, we have demonstrated, in a large simulation, that our method can perform on par with or better than previous methods. We did this by testing our methods using a large simulation on two types of

$K_{\text{true}} \downarrow$ / $\hat{K} \rightarrow$	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
2	**.98**	.02	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·
3	·	**.98**	·	.02	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·
4	·	·	**.98**	.02	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·
5	.02	·	·	**.94**	.04	·	·	·	·	·	·	·	·	·	·	·	·	·	·
6	.02	·	·	·	**.96**	.02	·	·	·	·	·	·	·	·	·	·	·	·	·
7	.02	·	·	·	.04	**.94**	·	·	·	·	·	·	·	·	·	·	·	·	·
8	·	·	·	·	·	.08	**.86**	.06	·	·	·	·	·	·	·	·	·	·	·
9	·	·	·	·	·	·	.08	**.82**	.10	·	·	·	·	·	·	·	·	·	·
10	·	.02	.04	·	·	·	.04	.26	**.40**	.22	.02	·	·	·	·	·	·	·	·
11	·	·	·	.02	·	·	·	.02	.28	**.44**	.24	·	·	·	·	·	·	·	·
12	.02	·	.04	.04	·	·	·	·	·	.28	**.24**	.26	.08	.04	·	·	·	·	·
13	·	.02	·	·	·	·	·	·	·	.04	.36	**.10**	.18	.22	.06	.02	·	·	·
14	·	.04	·	·	·	·	·	·	·	·	.08	.32	**.28**	.18	.08	.02	·	·	·
15	.02	.02	·	·	·	·	·	·	·	·	·	.10	.20	**.20**	.24	.12	.08	.02	·
16	.04	·	·	·	·	·	·	·	·	·	·	·	.08	.12	**.28**	.26	.12	.02	.08

(a) APW-RC.

$K_{\text{true}} \downarrow$ / $\hat{K} \rightarrow$	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
2	**.98**	.02	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·
3	·	**.98**	·	.02	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·
4	·	·	**.98**	.02	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·
5	·	·	·	**.88**	.10	·	·	·	·	·	.02	·	·	·	·	·	·	·	·
6	·	·	·	·	**.94**	.04	·	.02	·	·	·	·	·	·	·	·	·	·	·
7	·	·	·	·	·	**.92**	.04	·	.02	·	·	·	·	·	·	·	·	·	.02
8	·	·	·	·	·	.02	**.86**	.08	·	.04	·	·	·	·	·	·	·	·	·
9	·	·	·	·	·	·	.04	**.82**	.14	·	·	·	·	·	·	·	·	·	·
10	·	·	·	·	·	·	·	.12	**.40**	.28	.10	·	.06	.04	·	·	·	·	·
11	·	·	·	·	·	·	·	·	.12	**.44**	.24	.08	.08	·	.02	.02	·	·	·
12	·	·	·	·	·	·	·	·	·	.14	**.20**	.26	.22	.06	.04	.02	.02	·	.04
13	·	·	·	·	·	·	·	·	·	·	.16	**.06**	.26	.22	.16	.12	.02	·	·
14	·	·	·	·	·	·	·	·	.02	·	.02	.08	**.14**	.24	.18	.16	.12	.02	.02
15	·	·	·	·	·	·	·	·	·	·	·	·	.08	**.10**	.24	.20	.24	.12	.02
16	·	·	·	·	·	·	·	·	·	·	·	·	.06	.06	**.16**	.26	.14	.10	.22

(b) APW-Perm.

Table 5.21: Sampling distribution of APW-RC and APW-Perm, StdBeforeBest, on 100D Shaped data with 100 points.

data, ANOVA and shaped, at several dimensions and sample sizes. We feel this test demonstrated well how and when the methods tested break down and when the appear to work well. We summarize the results here across several categories.

$K_{\text{true}} \downarrow$ / $\hat{K} \rightarrow$	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
2	**1**	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·
3	.02	**.96**	.02	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·
4	·	·	**.98**	.02	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·
5	.02	·	·	**.98**	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·
6	·	·	·	.02	**.94**	.02	.02	·	·	·	·	·	·	·	·	·	·	·	·
7	·	·	·	·	.04	**.94**	.02	·	·	·	·	·	·	·	·	·	·	·	·
8	·	·	·	·	·	.10	**.84**	.06	·	·	·	·	·	·	·	·	·	·	·
9	·	·	·	·	·	·	.16	**.78**	.06	·	·	·	·	·	·	·	·	·	·
10	.04	.02	·	·	·	·	.06	.36	**.40**	.12	·	·	·	·	·	·	·	·	·
11	.02	·	·	·	·	·	·	.06	.34	**.44**	.14	·	·	·	·	·	·	·	·
12	.10	·	·	·	·	·	·	.02	.18	.30	**.30**	.10	·	·	·	·	·	·	·
13	.02	·	·	·	·	·	·	·	·	.26	.38	**.22**	.08	.04	·	·	·	·	·
14	.04	·	·	·	·	·	·	·	·	.02	.20	.38	**.28**	.06	.02	·	·	·	·
15	.04	·	·	·	·	·	·	·	·	·	.10	.26	.36	**.20**	.04	·	·	·	·
16	.02	·	·	·	·	·	·	·	·	·	.04	.12	.28	.32	**.20**	.02	·	·	·

Table 5.22: Sampling distribution of SS-Draws-VI on 100D Shaped data with 100 points.

$K_{\text{true}} \downarrow$ / $\hat{K} \rightarrow$	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
2	**.84**	.10	.06	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·
3	·	**.82**	.16	·	.02	·	·	·	·	·	·	·	·	·	·	·	·	·	·
4	·	·	**.70**	.30	·	·	·	·	·	·	·	·	·	·	·	·	·	·	·
5	·	·	·	**.72**	.24	.02	·	·	·	·	.02	·	·	·	·	·	·	·	·
6	·	·	·	·	**.82**	.16	·	.02	·	·	·	·	·	·	·	·	·	·	·
7	·	·	·	·	.04	**.80**	.16	·	·	·	·	·	·	·	·	·	·	·	·
8	·	·	·	·	·	.06	**.76**	.16	.02	·	·	·	·	·	·	·	·	·	·
9	·	·	·	·	·	·	.10	**.76**	.12	.02	·	·	·	·	·	·	·	·	·
10	·	·	·	·	·	·	.02	.30	**.38**	.18	.08	·	.02	.02	·	·	·	·	·
11	·	·	·	·	·	·	·	.04	.18	**.52**	.22	.02	·	.02	·	·	·	·	·
12	·	·	·	.02	·	·	.02	·	.06	.14	**.40**	.26	.08	.02	·	·	·	·	·
13	·	·	·	·	·	·	.02	·	.06	.12	.32	**.22**	.10	.12	.04	·	·	·	·
14	·	·	·	·	·	·	·	·	.02	.06	.08	.32	**.20**	.16	.14	.02	·	·	·
15	·	·	·	·	·	·	·	·	·	·	.06	.24	.24	**.14**	.24	.06	.02	·	·
16	·	·	·	·	·	·	·	·	·	·	.02	.06	.28	.20	**.24**	.16	.04	·	·

Table 5.23: Sampling distribution of the gap statistic on 100D Shaped data with 100 points.

5.4.1 Increasing Dimension

The methods that performed will in lower dimensions did not always perform well as the dimension increased. Here are some sample observations:

Gap Statistic The predictor proposed with the gap statistic performs better and better as dimension increases, but doesn't work as well in lower dimensions. However, in 2 dimensions, StdBeforeBest works much bet-

ter but fails as the dimension increases.

APW APW-RC tended to perform quite well regardless of the dimension. APW-Perm performed poorly in lower dimensions but better with increasing dimension.

Data Perturbation Methods Data perturbation methods tended to be fairly consistent in their accuracy, exhibiting the same problems and strengths regardless of dimension. In only one case out of 20, though, did one of the data perturbation methods win.

5.4.2 Cluster Shape

Cluster shape presented different challenges, most notably to the methods making some modeling assumptions.

Gap Statistic The gap statistic undoubtedly is the most sensitive to cluster shape. It performs best in lower dimensional, low sample size ANOVA data, where its Gaussian modeling assumptions hold. In all cases, it performs noticeably worse on the shaped data than on the comparable ANOVA data relative to the other methods.

APW APW-RC seems to exhibit some dependence on the shape of the mixture components in the generated data, something possibly implicit in the reclustering stage of forming the baseline. APW-Perm, however, seems to handle shaped data without any problems. In many such cases, it is the winner.

Data Perturbation Methods Again, these methods seem fairly robust against changes in the data shape. This is what would be expected, as they make no modeling assumptions.

5.4.3 Sample Size

Gap Statistic The gap statistic performed better relative to the other methods as sample size decreased if its modeling assumptions held

(i.e. on ANOVA data), but performed worse if they didn't (i.e. on shaped data).

APW APW-RC tended to perform the same relative to everyone else in decreasing sample size, but the performance of APW-Perm decreased. This is to be expected, as the permutation tests in APW-Perm work better with more data.

Data Perturbation Methods These methods, while performing slightly worse, did not exhibit a substantial drop in performance.

5.4.4 Other Considerations

It is worth noting that we compared the analytically weakest tool from our method – a scalar stability index – against the other methods. While our method provides many more analysis tools such as the heatmap plot and per-cluster stability indices, any additional tools from the other methods provide limited additional useful information. Furthermore, while we did not explicitly test the other stability statistics provided by our method, the fact that the scalar statistic summarizing all of them performs so well indicates that the more informative parts will as well.

Future work in this matter includes testing our method extensively on real data. This includes analysis of several classification datasets where the classification labels are hidden. A good clustering method combined with a good cluster validation technique should be able to reproduce the labels with a reasonable amount of accurately provided the original classes were well separated.

Chapter 6

Limits in High Dimensions

While developing a full theory surrounding our proposed approach to cluster stability is a difficult task, several relevant results can be obtained. In this chapter, we describe a number of results concerning the theoretical properties of our method in high dimensions. In particular, we look at the assymptotics of our method as $p \to \infty$. While not perhaps applicable to the more common low dimensional uses, this gives us an idea about how our method is likely to perform on high dimensional data.

The chapter is divided logically into two parts. The first concerns the limits and properties clusterings based on the squared error cost function as $p \to \infty$. In particular, we show that in the presence of noisy data, the empirical variance and covariance of the nonrandom components of the points in the partition must grow at a minimum rate of $\Omega\left(\sqrt{p}\right)$ in order for the cost function to be meaningful. The most telling corollary to this result is that if there are only a finite number of meaningful, non-random components, and the rest are noisy, feature selection must be employed.

Having established the limits of clustering as the dimension increases, we then look at the properties of our method in the same situation. Our general strategy is to show that when the clustering breaks down, so does our method, and that our method may give reasonable results when the clustering is meaningful.

6.1 Limits of Clustering in High Dimensions

To investigate the limits of clustering in high dimensions, we use the widely accepted model of real process where the true values of the data are obscured by a noisy random variable, i.e. the observed variable Y is related to the true value x by

$$Y = x + \varepsilon \tag{6.1}$$

where ε is a random variable having a distribution with a mean of zero and non-zero variance. Using this model, we intend to show that clustering based on the vanilla version of the cost function breaks down in high dimensions unless the distance between the true data points grows at a rate of $\Omega\big(\sqrt{p}\big)$. In many contexts, this implies that feature selection is absolutely necessary if a clustering of the data is going to be meaningful.

To prove this result, we first need formal definitions of partitions on the data, the cost function, and the noise distribution ε.

6.1.1 Partitions

A partitioning, or clustering, on a set of n data points is just a grouping of those points into K non-overlapping sets, or partitions. For our purposes, we restrict ourselves to non-empty partitions, so each partition has at least one point assigned to it. Also, as a matter of notation, our partitions are defined in terms of the indices $1, 2, ..., n$. We present this formally in the next definition:

Definition 6.1.1. Partitioning

Given n points and a number of clusters K, a partitioning

$$\mathcal{P} = \{P_1, P_2, ..., P_K\} \tag{6.2}$$

is a set of K subsets of $\{1, 2, ..., n\}$ such that

A. $\forall\, k \in \{1, 2, ..., K\},\ P_k \neq \emptyset$.

B. $\forall\, i, j \in \{1, 2, ..., K\},\ i \neq j,\ P_i \cap P_j = \emptyset$.

C. $\cup_{k=1}^{K} P_k = \{1, 2, ..., n\}$.

In other words, the sets are non-empty, non-overlapping, and all indices are in at least one set.

Definition 6.1.2. Class of Partitionings Let $\mathscr{P}_{nK}$ be the set of all partitionings $\mathcal{P}$ of n points into K partitions.

6.1.2 Properties of The Squared Error Cost Function

We use the same cost function that is used in many clustering algorithms, namely the squared error cost function. Recall the definition of squared error cost from section 1.1.3:

$$\text{cost}\Big(\mathscr{X}^{(p)}\Big) = \sum_{i=1}^{n} \|\mathbf{x}_i - \boldsymbol{\mu}\|_2^2 \tag{6.3}$$

where

$$\boldsymbol{\mu} = \frac{1}{n} \sum_{i=1}^{n} \mathbf{x}_i. \tag{6.4}$$

The corresponding version for partitions is just

$$\text{cost}\Big(\mathscr{X}^{(p)}, \mathcal{P}\Big) = \sum_{k} \text{cost}\Big(\Big\{\mathbf{x}_i \in \mathscr{X}^{(p)} : i \in P_k\Big\}\Big). \tag{6.5}$$

This cost function is the basis for our investigation of clustering in high dimensions. To proceed, we need to first need to formally note a few properties of the cost function. These provide the basis for our result of clustering in high dimensions.

Theorem 6.1.3. Expected Cost *Let $\mathcal{E}^{(p)} = \{\varepsilon_1, \varepsilon_2, ..., \varepsilon_n\}$ be a set of n points such that for $i = 1, 2, ..., n$,*

A. $\mathbb{E}\,\varepsilon_i = 0$.

B. $\mathbb{E}\,\varepsilon_i^2 = \sigma^2$, $0 < \sigma \leq C < \infty$.

Suppose $\mathcal{P} \in \mathscr{P}_{nK}$. *Then*

$$\mathbb{E}\ \mathrm{cost}\Big(\mathcal{E}^{(p)}, \mathcal{P}\Big) = (n-K)\sigma^2 \tag{6.6}$$

Proof. Let

$$M_k = \frac{1}{|P_k|}\sum_{i\in P_k} \varepsilon_i \tag{6.7}$$

be the mean of each partition. We proceed by expanding out the cost function:

$$\mathbb{E}\ \mathrm{cost}\Big(\mathcal{E}^{(p)}, \mathcal{P}\Big) = \mathbb{E}\left[\sum_{k=1}^{K}\sum_{i\in P_k}(\varepsilon_i - M_k)^2\right] \tag{6.8}$$

$$= \sum_{k=1}^{K}\sum_{i\in P_k} \mathbb{E}\,\varepsilon_i^2 - 2\,\mathbb{E}\left[\varepsilon_i M_k\right] + \mathbb{E}\left[M_k^2\right]. \tag{6.9}$$

Since the partitions of $\mathcal{P}$ are disjoint, there are exactly n terms of the form $\mathbb{E}\,\varepsilon_i^2$ in total, with total expected value $n\sigma^2$:

$$\mathbb{E}\ \mathrm{cost}\Big(\mathcal{E}^{(p)}, \mathcal{P}\Big) = n\sigma^2 \sum_{k=1}^{K}\sum_{i\in P_k} -2\,\mathbb{E}\left[\varepsilon_i M_k\right] + \mathbb{E}\left[M_k^2\right] \tag{6.10}$$

Now for each partition P_k,

$$\mathbb{E}\left[\varepsilon_i M_k\right] = \mathbb{E}\left[\varepsilon_i^2 + \frac{1}{|P_k|}\sum_{\substack{j\in P_k\\ j\neq i}} \varepsilon_i\varepsilon_j\right] \tag{6.11}$$

$$= \mathbb{E}\,\varepsilon_i^2 + \frac{1}{|P_k|}\sum_{\substack{j\in P_k\\ j\neq i}} (\mathbb{E}\,\varepsilon_i)(\mathbb{E}\,\varepsilon_j) \tag{6.12}$$

$$= \sigma^2 \tag{6.13}$$

Furthermore,

$$\mathbb{E}\left[M_k^2\right] = \mathbb{E}\left[\left(\frac{1}{|P_k|}\sum_{i\in P_k}\varepsilon_i\right)\left(\frac{1}{|P_k|}\sum_{j\in P_k}\varepsilon_j\right)\right] \tag{6.14}$$

$$= \mathbb{E}\left[\frac{1}{|P_k|^2}\sum_{i\in P_k}\left(\varepsilon_i^2 + \sum_{\substack{j\in P_k\\ j\neq i}}\varepsilon_i\varepsilon_j\right)\right] \tag{6.15}$$

$$= \frac{1}{|P_k|^2}\left(\sum_{i\in P_k}\mathbb{E}\,\varepsilon_i^2 + 0\right) \tag{6.16}$$

$$= \frac{1}{|P_k|}\sigma^2 \tag{6.17}$$

Thus we have that

$$\mathbb{E}\ \mathrm{cost}\Big(\mathcal{E}^{(p)}, \mathcal{P}\Big) = n\sigma^2 + \sum_{k=1}^{K}\sum_{i\in P_k}\left(-\frac{2\sigma^2}{|P_k|} + \frac{\sigma^2}{|P_k|}\right) \tag{6.18}$$

$$= n\sigma^2 + \sum_{k=1}^{K}\left(-\sigma^2\right) \tag{6.19}$$

$$= (n-K)\sigma^2 \tag{6.20}$$

which completes the proof. □

Often, in our analysis, we are looking at the difference of two cost functions. For that, we have the following handy corollary.

Corollary 6.1.4. Expected Difference of Cost Functions *Let* $\mathcal{E}^{(p)} = \{\varepsilon_1, \varepsilon_2, ..., \varepsilon_n\}$ *be a set of* n *points such that for* $i = 1, 2, ..., n$,

A. $\mathbb{E}\,\varepsilon_i = 0$.

B. $\mathbb{E}\,\varepsilon_i^2 = \sigma^2$, $0 < \sigma \leq C < \infty$.

Suppose $\mathcal{P}, \mathcal{Q} \in \mathscr{P}_{nK}$. *Then*

$$\mathbb{E}\left[\text{cost}\Big(\mathcal{E}^{(p)}, \mathcal{P}\Big) - \text{cost}\Big(\mathcal{E}^{(p)}, \mathcal{Q}\Big)\right] = 0 \tag{6.21}$$

Proof. The proof follows from the previous theorem.

$$\mathbb{E}\left[\text{cost}\Big(\mathcal{E}^{(p)}, \mathcal{P}\Big) - \text{cost}\Big(\mathcal{E}^{(p)}, \mathcal{Q}\Big)\right] = (n-K)\sigma^2 - (n-K)\sigma^2 \tag{6.22}$$

$$= 0 \tag{6.23}$$

□

We have one final set of results concerning properties of squared error cost of random variables. The following theorems, while perhaps intuitively obvious, allow us to formally prove the impossibility theorem in the next section. The main idea is to prove that the probability that the cost of two nonrandom partitions over independent random variables is always less than one. This is a necessary step in applying the central limit theorem as part of the impossibility theorem.

Lemma 6.1.5. *Let* $\mathscr{X}^{(p)} = x_1, x_2, ..., x_n$ *be a set of* p *points, and suppose* $\mathcal{P}, \mathcal{Q} \in \mathscr{P}_{nK}$, $\mathcal{P} \neq \mathcal{Q}$. *Then there exists a symmetric* $n \times n$ *matrix* $B = [b_{ij}]$ *such that*

$$\text{cost}\Big(\mathscr{X}^{(p)}, \mathcal{P}\Big) - \text{cost}\Big(\mathscr{X}^{(p)}, \mathcal{Q}\Big) = \mathbf{x}^T\mathbf{B}\mathbf{x} \tag{6.24}$$

where $\mathbf{x} = (x_1, x_2, ..., x_n)$.

Proof. Recall that the squared error cost function can be expressed in terms of the difference between points (theorem 1.1.5):

$$\text{cost}\Big(\mathcal{E}^{(p)}, \mathcal{P}\Big) = \frac{1}{2}\sum_k \frac{1}{|P_k|}\sum_{i,j\in P_k}(x_i - x_j)^2 \tag{6.25}$$

$$= \sum_i x_i^2 - \sum_k \frac{1}{|P_k|}\sum_{i,j\in P_k} x_i x_j \tag{6.26}$$

$$= \sum_{i,j} a_{ij}^P x_i x_j \tag{6.27}$$

where

$$a_{ij}^P = \begin{cases} (|P_k| - 1)/\,|P_k| & i = j;\, i,j \in P_k \\ -\,1/\,|P_k| & \exists\, k \text{ s.t. } i,j \in P_k, i \neq j \\ 0 & \text{otherwise} \end{cases} \tag{6.28}$$

The same is true for partition $\mathcal{Q}$. We can re-express this above in matrix form, where $\mathbf{A}_P = a_{ij}^P$ and likewise for $\mathbf{A}_Q$.

$$\text{cost}\Big(\mathcal{E}^{(p)}, \mathcal{P}\Big) = \mathbf{x}^T \mathbf{A}_P \mathbf{x} \tag{6.29}$$

$$\text{cost}\Big(\mathcal{E}^{(p)}, \mathcal{Q}\Big) = \mathbf{x}^T \mathbf{A}_Q \mathbf{x} \tag{6.30}$$

Note that $a_{ij}^P = a_{ji}^P$, so $\mathbf{A}_P$ and $\mathbf{A}_Q$ are symmetric. The difference between the two cost functions can then be expressed as:

$$\text{cost}\Big(\mathcal{E}^{(p)}, \mathcal{P}\Big) - \text{cost}\Big(\mathcal{E}^{(p)}, \mathcal{Q}\Big) \tag{6.31}$$

$$= \mathbf{x}^T \mathbf{A}_P \mathbf{x} - \mathbf{x}^T \mathbf{A}_Q \mathbf{x} \tag{6.32}$$

$$= \mathbf{x}^T (\mathbf{A}_P - \mathbf{A}_Q)\mathbf{x} \tag{6.33}$$

Thus the theorem is proved. □

Now that we have shown these basic properties of the squared error cost

function, we just need to formalize our notion of noisy data. We will then be ready to present our main clustering impossibility theorem.

6.1.3 Noisy Data

In this section, we present a formal definition of noisy data, or, more correctly, the class of noise generating distributions.

Definition 6.1.6. Noise Generating Distributions

Let $\mathfrak{F}$ be the set of all sequences of distributions $\mathscr{F} = (F_1, F_2, ...)$ that satisfy the following properties:

A. $\forall\, q \in \{1, 2, 3, ...\}$, if $\varepsilon_q \sim F_q$, then $\mathbb{E}\,\varepsilon_q = 0$.

B. $\forall\, q \in \{1, 2, 3, ...\}$, if $\varepsilon_q \sim F_q$, then $\mathbb{E}\,\varepsilon_q^4 = \rho_q$, $0 < L_q \leq \rho_q \leq U_q < \infty$.

The fourth moment is necessary to govern the convergence of the second moment of the squared error cost function. Note that such a constraint provides similar bounds on all the lower order moments.

One of the key steps in the theorem is showing that a sequence of similarly defined random variables functions converges to a normal distribution centered at 0. Such a result is significant in its own right, so we present it, and another useful lemma, in the following section.

6.1.4 Lemmas Concerning the Central Limit Theorem

Lemma 6.1.7. *Let $Z_q, q = 1, 2, ...$ be a sequence of random variables with $\mathbb{E}\, Z_q = r_q$, where r_q is a non-random sequence such that*

$$\frac{1}{\sqrt{p}} \sum_{q=1}^{p} r_q \to 0 \text{ as } p \to \infty. \tag{6.34}$$

Suppose $\mathbb{E}\left[Z_q^2\right] = \sigma_q^2$, $0 < L \leq \sigma_q \leq U < \infty$. Let $S_q = Z_1 + Z_2 + \cdots + Z_q$ and $c_q^2 = \sigma_1^2 + \sigma_2^2 + \cdots + \sigma_q^2$ Then

$$\frac{S_n}{c_n} \xrightarrow{D} \varphi \text{ as } n \to \infty \tag{6.35}$$

where φ is a random variable having a standard normal distribution.

Proof. Let F_q be the distribution function of $Z_q - r_q$.

The proof follows from the Lindeberg-Feller conditions on the central limit theorem [ADD99]. Let F_q be the distribution function of Z_q, and let

$$\mu_n = \sum_{q=1}^{n} r_q \tag{6.36}$$

If, $\forall\, \varepsilon > 0$,

$$LF_n = \frac{1}{c_n^2} \sum_{q=1}^{n} \int_{\{x\,:\,|x-r_q| \geq \varepsilon c_n\}} (x - r_q)^2 \mathrm{d}F_q(x) \to 0 \text{ as } n \to \infty, \tag{6.37}$$

then Lindeberg's theorem [ADD99] gives us

$$\frac{S_n - \mu_n}{c_n} \xrightarrow{D} \varphi \text{ as } n \to \infty \tag{6.38}$$

We later show that the μ_n term drops out in the limit.

The strategy now in showing that this condition is satisfied is to upper bound the expression and show that this bound goes to zero.

Let $s_n^2 = \underset{q \in \{1,2,\ldots,n\}}{\text{mean}} \sigma_q^2$. Then

$$LF_n = \frac{1}{s_n^2 n} \sum_{q=1}^{n} \int_{\left\{x\,:\,|x-r_q| \geq \varepsilon s_n \sqrt{n}\right\}} (x - r_q)^2 \mathrm{d}F_q(x) \tag{6.39}$$

$$\leq \frac{1}{s_n^2 n} n \max_{q \in \{1,2,\ldots,n\}} \int_{\left\{x\,:\,|x-r_q| \geq \varepsilon s_n \sqrt{n}\right\}} (x - r_q)^2 \mathrm{d}F_q(x). \tag{6.40}$$

Now we know that σ_q is bounded below by L, and the integral is always positive, so

$$LF_n \leq \frac{1}{L^2} \max_{q \in \{1,2,\ldots,n\}} \int_{\{x\,:\,|x-r_q| \geq \varepsilon c_n\}} (x - r_q)^2 \mathrm{d}F_q(x) \tag{6.41}$$

$$\leq \frac{1}{L^2} \max_{q \in \{1,2,\ldots,n\}} \int_{\left\{x\,:\,|x-r_q| \geq \varepsilon L \sqrt{n}\right\}} (x - r_q)^2 \mathrm{d}F_q(x) \tag{6.42}$$

Now for all r_q, $\{x : |x - r_q| \geq \varepsilon L\sqrt{n}\} \searrow \emptyset$ as $n \to \infty$, so $\forall\, q$,

$$\int_{\{x\,:\,|x-r_q|\geq\varepsilon L\sqrt{n}\}} (x - r_q)^2 \mathrm{d}F_q(x) \;\to\; 0 \text{ as } n \to \infty. \tag{6.43}$$

Thus

$$\frac{1}{L^2} \max_{q\in\{1,2,\ldots,n\}} \int_{\{x\,:\,|x-r_q|\geq\varepsilon L\sqrt{n}\}} (x - r_q)^2 \mathrm{d}F_q(x) \;\to\; 0 \text{ as } n \to \infty. \tag{6.44}$$

so

$$\frac{S_n - \mu_n}{\sqrt{n}s_n} \;\xrightarrow{D}\; \varphi \text{ as } n \to \infty. \tag{6.45}$$

Note that this is equivalent to equation (6.38). However, by assumption,

$$\frac{1}{s_n}\frac{\mu_n}{\sqrt{n}} \to 0 \text{ as } n \to \infty, \tag{6.46}$$

so equation (6.45) reduces to

$$\frac{S_n}{c_n} \;\xrightarrow{D}\; \varphi \text{ as } n \to \infty. \tag{6.47}$$

proving the lemma. □

Lemma 6.1.8. *Let Z_q be a sequence of random variables such that*

$$\sum_{q=1}^{p} \frac{Z_q}{\sigma\sqrt{p}} \;\xrightarrow{D}\; \varphi \tag{6.48}$$

where $\varphi \sim \mathcal{N}(0, 1)$ and σ is some constant. Then as $p \to \infty$,

$$\mathbb{P}\Big(\sum\nolimits_{q=1}^{p} Z_q \geq 0\Big) \;\to\; \frac{1}{2}. \tag{6.49}$$

Proof. We can rewrite the expression in equation (6.49) as

$$\mathbb{P}\Big(\sum\nolimits_{q=1}^{p} Z_q \geq 0\Big) = \mathbb{P}\bigg(\frac{1}{\sqrt{p}}\sum\nolimits_{q=1}^{p} Z_q \geq 0\bigg) \tag{6.50}$$

However, by assumption,

$$\sum_{q=1}^{p} \frac{Z_q}{\sigma\sqrt{p}} \xrightarrow{D} \varphi. \tag{6.51}$$

Thus by definition of convergence in distribution,

$$\mathbb{P}\Big(\sum\nolimits_{q=1}^{p} Z_q \geq 0\Big) \rightarrow \mathbb{P}(\varphi \geq 0). \tag{6.52}$$

But φ comes from a distribution that is symmetric about 0, so

$$\mathbb{P}(\varphi \geq 0) = \frac{1}{2} \tag{6.53}$$

and the statement is proved. □

We now have everything in place to present our clustering impossibility theorem.

6.1.5 Impossibility Theorem for Clustering in High Dimensions

We now present our result limiting the meaningfulness of clustering in high dimensions in the following theorem.

Theorem 6.1.9. Limits of Clustering in High Dimensions.

Suppose $\mathscr{X}^{(p)} = \mathbf{x}_1^{(p)}, \mathbf{x}_2^{(p)}, ..., \mathbf{x}_n^{(p)}$, $p = 1, 2, ...$ *is a given sequence of sets of* n *points with increasing dimension* p *and such that* $x_q^{(p)} = x_q^{(r)}$ *for all* $p \neq r$ *and* $q \leq \min\{p, r\}$.

Suppose there exists a sequence of points $\mathbf{u}^{(p)}$ *so that* $\forall\, i, j \in \{1, 2, ..., n\}$,

$$(\mathbf{x}_i^{(p)} - \mathbf{u}^{(p)})^T(\mathbf{x}_j^{(p)} - \mathbf{u}^{(p)}) \in \mathrm{o}(\sqrt{p}) \tag{6.54}$$

Next, suppose $F_q \in \mathfrak{F}$, $q = 1, 2, ...$ *is a sequence of noise generating distributions. Let* $\boldsymbol{\varepsilon}_{iq}^{(p)} \sim F_q$ *for* $i = 1, 2, ..., n$, *with* $\boldsymbol{\varepsilon}_i^{(p)} = (\varepsilon_{i1}, \varepsilon_{i2}, ..., \varepsilon_{ip})$, *and define*

$$\mathbf{Y}_i^{(p)} = \mathbf{x}_i^{(p)} + \boldsymbol{\varepsilon}_i^{(p)} \tag{6.55}$$

$$\mathscr{Y}^{(p)} = (\mathbf{Y}_1^{(p)}, \mathbf{Y}_2^{(p)}, ..., \mathbf{Y}_n^{(p)}). \tag{6.56}$$

Finally, suppose $\mathcal{P}, \mathcal{Q} \in \mathscr{P}_{nK}$, $\mathcal{P} \neq \mathcal{Q}$, *are two different partitions of* n *points. Then*

$$\mathbb{P}\Big(\text{cost}\Big(\mathscr{Y}^{(p)}, \mathcal{P}\Big) \leq \text{cost}\Big(\mathscr{Y}^{(p)}, \mathcal{Q}\Big)\Big) \;\to\; \frac{1}{2} \tag{6.57}$$

as $p \to \infty$.

Proof. The structure of the proof is to show that the difference the cost functions two cost functions divided by $\sqrt{p}$ converges to a standard normal distribution as $p \to \infty$. A random variable from this distribution has equal probability of being negative or positive, so one cost function has a probability of $\frac{1}{2}$ of being greater than the other.

We begin by setting up the notation and re-expressing the difference in cost functions, then show that the terms either drop out or together converge in distribution to a normal.

Continuing, recall that

$$\mathcal{P} = (P_1, P_2, ..., P_K) \tag{6.58}$$

$$\mathcal{Q} = (Q_1, Q_2, ..., Q_K) \tag{6.59}$$

Let $\mathbf{u}^{(p)}$ be a sequence in p of p-dimensional points[1] such that $\forall\, i, j \in \{1, 2, ..., n\}$,

$$(\mathbf{x}_i^{(p)} - \mathbf{u}^{(p)})^T(\mathbf{x}_j^{(p)} - \mathbf{u}^{(p)}) \in \text{o}(\sqrt{p}), \tag{6.60}$$

as guaranteed by the assumptions. For convenience, define

[1]The $\mathbf{u}^{(p)}$ terms here allow us to accomidate sequences of points with a non-zero mean.

$$\mathbf{x'}_i^{(p)} = \mathbf{x}_i^{(p)} - \mathbf{u}^{(p)} \tag{6.61}$$

$$\mathscr{X}'^{(p)} = \mathbf{x'}_1^{(p)}, \mathbf{x'}_2^{(p)}, ..., \mathbf{x'}_n^{(p)} \tag{6.62}$$

$$\mathbf{Y'}_i^{(p)} = \mathbf{x'}_i^{(p)} + \boldsymbol{\varepsilon}_i^{(p)} \tag{6.63}$$

$$\mathscr{Y}'^{q} = (\mathbf{Y'}_1^{(p)}, \mathbf{Y'}_2^{(p)}, ..., \mathbf{Y'}_n^{(p)}) \tag{6.64}$$

By corrolary 1.1.7, the cost function is invariant to such shifts, so

$$\text{cost}\big(\mathscr{Y}'^{q}, \mathcal{P}\big) = \text{cost}\Big(\mathscr{Y}^{(p)}, \mathcal{P}\Big) \tag{6.65}$$

$$\text{cost}\big(\mathscr{Y}'^{q}, \mathcal{Q}\big) = \text{cost}\Big(\mathscr{Y}^{(p)}, \mathcal{Q}\Big) \tag{6.66}$$

Thus proving the theorem for the shifted version is sufficient. Now, for convenience, define the following terms:

$$n_k^{\mathcal{P}} = |P_k| \tag{6.67}$$

$$n_k^{\mathcal{Q}} = |Q_k| \tag{6.68}$$

$$\mathbf{M}_k^{\mathcal{P}} = \underset{i \in P_k}{\text{mean}}\, \boldsymbol{\varepsilon}_i^{(p)} = \frac{1}{n_k^{\mathcal{P}}} \sum_{i \in P_k} \boldsymbol{\varepsilon}_i^{(p)} \tag{6.69}$$

$$\mathbf{M}_k^{\mathcal{Q}} = \underset{i \in Q_k}{\text{mean}}\, \boldsymbol{\varepsilon}_i^{(p)} = \frac{1}{n_k^{\mathcal{Q}}} \sum_{i \in Q_k} \boldsymbol{\varepsilon}_i^{(p)} \tag{6.70}$$

$$\boldsymbol{\mu}_k^{\mathcal{P}} = \underset{i \in P_k}{\text{mean}}\, \mathbf{x'}_i^{(p)} = \frac{1}{n_k^{\mathcal{P}}} \sum_{i \in P_k} \mathbf{x'}_i^{(p)} \tag{6.71}$$

$$\boldsymbol{\mu}_k^{\mathcal{Q}} = \underset{i \in Q_k}{\text{mean}}\, \mathbf{x'}_i^{(p)} = \frac{1}{n_k^{\mathcal{Q}}} \sum_{i \in Q_k} \mathbf{x'}_i^{(p)} \tag{6.72}$$

By theorem 1.1.3, the cost function is separable into components. Furthermore, by assumption, these components are identical for different p. Thus we can define a sequence of random variables Z_q as the cost difference of the qth component. Let

$$Z_q = \text{cost}\big(\mathscr{Y'}^q_q, \mathcal{P}\big) - \text{cost}\big(\mathscr{Y'}^q_q, \mathcal{Q}\big). \tag{6.73}$$

Note that given this definition, if we can show that $\mathbb{P}\Big(\sum_{q=1}^{p} Z_q \geq 0\Big) \to 0$ as $p \to \infty$, then $\mathbb{P}\big(\text{cost}\big(\mathscr{Y}^{(p)}, \mathcal{P}\big) \geq \text{cost}\big(\mathscr{Y}^{(p)}, \mathcal{Q}\big)\big) \to \frac{1}{2}$ and the theorem is proved. Our strategy is to do exactly this, relying heavily on lemma 6.1.7 and lemma 6.1.8.

Now lemma 6.1.5 allows us to re-express the differences in cost functions as a matrix equation. Let $\mathbf{B}$ be a symmetric matrix such that

$$Z_q = \big(\mathbf{y}'_{:q}\big)^T \mathbf{B} \mathbf{y}'_{:q} = \text{cost}\big(\mathscr{Y'}^q_q, \mathcal{P}\big) - \text{cost}\big(\mathscr{Y'}^q_q, \mathcal{Q}\big) \tag{6.74}$$

Note that $\mathbf{B}$ depends only on the partitioning, so it will be the same for all q. We can expand this expression out:

$$\begin{aligned} Z_q &= \big(\mathbf{y}'_{:q}\big)^T \mathbf{B} \mathbf{y}'_{:q} && (6.75)\\ &= (\mathbf{x}'_{:q} + \boldsymbol{\varepsilon}_{:q})^T \mathbf{B} (\mathbf{x}'_{:q} + \boldsymbol{\varepsilon}_{:q}) && (6.76)\\ &= {\mathbf{x}'_{:q}}^T \mathbf{B} \mathbf{x}'_{:q} + \boldsymbol{\varepsilon}^T_{:q} \mathbf{B} \boldsymbol{\varepsilon}_{:q} - 2{\mathbf{x}'_{:q}}^T \mathbf{B} \boldsymbol{\varepsilon}_{:q} && (6.77) \end{aligned}$$

We now show that Z_q satisfies the assumptions of lemma 6.1.7 and the theorem follows directly.

The expectation of Z_q is just

$$\begin{aligned} \mathbb{E}\, Z_q = {\mathbf{x}'_{:q}}^T \mathbf{B} \mathbf{x}'_{:q} + \mathbb{E}\left[\boldsymbol{\varepsilon}^T_{:q} \mathbf{B} \boldsymbol{\varepsilon}_{:q}\right] - 2{\mathbf{x}'_{:q}}^T \mathbf{B} (\mathbb{E}\, \boldsymbol{\varepsilon}_{:q}) && (6.78)\\ && (6.79) \end{aligned}$$

Now

$$\mathbb{E}\left[\boldsymbol{\varepsilon}^T_{:q} \mathbf{B} \boldsymbol{\varepsilon}_{:q}\right] = \mathbb{E}\left[\text{cost}\Big(\mathcal{E}^{(p)}, \mathcal{P}\Big) - \text{cost}\Big(\mathcal{E}^{(p)}, \mathcal{Q}\Big)\right] = 0 \tag{6.80}$$

by corrolary 6.1.4. This leaves us with

$$\mathbb{E}\, Z_q = \mathbf{x}'^{\,T}_{:q}\mathbf{B}\mathbf{x}'_{:q} - 2\mathbf{x}'^{\,T}_{:q}\mathbf{B}(\mathbb{E}\,\boldsymbol{\varepsilon}_{:q}) \tag{6.81}$$

$$= \mathbf{x}'^{\,T}_{:q}\mathbf{B}\mathbf{x}'_{:q} \tag{6.82}$$

as $\mathbb{E}\,\varepsilon_{iq} = 0\ \forall i, q$. Now, to satisfy the conditions of lemma 6.1.7, we need to show that

$$\frac{1}{\sqrt{p}}\sum_{q=1}^{p}\mathbf{x}'^{\,T}_{:q}\mathbf{B}\mathbf{x}'_{:q} \to 0 \text{ as } p \to \infty. \tag{6.83}$$

Now we have by assumption that

$$\frac{\mathbf{x'}^T\mathbf{x'}}{\sqrt{p}} \to 0 \text{ as } p \to \infty \tag{6.84}$$

so also

$$\left(\max_{i,j} b_{ij}\right)\frac{\mathbf{x'}^T\mathbf{x'}}{\sqrt{p}} \to 0 \text{ as } p \to \infty. \tag{6.85}$$

Thus equation (6.83) is proved. We now show that $0 < L \leq \mathbb{E}\left[Z_q^2\right] \leq U < \infty$, which is the other condition in lemma 6.1.7.

Since $\mathrm{Var}\,[X] = \mathbb{E}\left[X^2\right] - (\mathbb{E}\,X)^2$, showing that the variance is lower and upper bounded is sufficient to say that the second moment is also upper and lower bounded. This allows us to ignore the nonrandom $\mathbf{x}'^{\,T}_{:q}\mathbf{B}\mathbf{x}'_{:q}$ term as it does not affect the variance.

Now

$$\boldsymbol{\varepsilon}^T_{:q}\mathbf{B}\boldsymbol{\varepsilon}_{:q} - 2\mathbf{x}'^{\,T}_{:q}\mathbf{B}\boldsymbol{\varepsilon}_{:q} = (\boldsymbol{\varepsilon}_{:q} - 2\mathbf{x}'_{:q})^T\mathbf{B}\boldsymbol{\varepsilon}_{:q} \tag{6.86}$$

$$= \sum_{i,j} b_{ij}(\varepsilon_{iq} - 2x_{iq})\varepsilon_{jq} \tag{6.87}$$

For $i \neq j$, ε_{iq} and ε_{jq} are independent, and $\mathrm{Var}\,[\varepsilon_{iq}] \leq U < \infty$ for all i and q, so $\exists\, U'$ such that

$$\mathrm{Var}\left[(\varepsilon_{iq} - 2x_{iq})\varepsilon_{jq}\right] \leq U' < \infty \tag{6.88}$$

If $i = j$ and $b_{ij} \neq 0$, we have by assumption that $\mathbb{E}\left[\varepsilon_{iq}^2\right] \leq U < \infty$, so that term is bounded.

Furthermore, $\mathbf{B} \neq \mathbf{0}$, so $b_{ij} \neq 0$ for at least one pair of indices i and j, and $\mathrm{Var}\left[\varepsilon_{iq}\right] \geq L > 0$ for all i and q. Because upper bounds exist for each term in equation (6.87), and a lower bound exists for at least one term, there exist bounds L'' and U'' such that

$$0 < L'' \leq \mathrm{Var}\left[\boldsymbol{\varepsilon}_{:q}^T \mathbf{B} \boldsymbol{\varepsilon}_{:q} - 2\mathbf{x}_{:q}'^T \mathbf{B} \boldsymbol{\varepsilon}_{:q}\right] \leq U'' < \infty \tag{6.89}$$

$$\Leftrightarrow 0 < L'' \leq \mathrm{Var}\left[Z_q\right] \leq U'' < \infty \tag{6.90}$$

Thus the conditions of lemma 6.1.5 are satisfied.

From lemma 6.1.5, we thus have that

$$\frac{1}{\sqrt{p}} \sum_{q=1}^{p} \frac{Z_q}{s_p} \xrightarrow{D} \varphi \text{ as } p \to \infty \tag{6.91}$$

where $s_p^2 = \sum_{q=1}^{p} \sigma_q^q$. This also satisfies the conditions of lemma 6.1.8; thus

$$\mathbb{P}\Big(\sum\nolimits_{q=1}^{p} Z_q \geq 0\Big) \to \frac{1}{2}. \tag{6.92}$$

But

$$\sum_{q=1}^{p} Z_q = \mathrm{cost}\Big(\mathscr{Y}_q^{(p)}, \mathcal{P}\Big) - \mathrm{cost}\Big(\mathscr{Y}_q^{(p)}, \mathcal{Q}\Big), \tag{6.93}$$

so the theorem is proved. □

To relate the above theorem to several practical situations, we also present the following corollary:

Corollary 6.1.10. *Let $\mathcal{X}^{(p)}$ be a p dimensional dataset with n components drawn from the same random process in which only a finite set of dimension*

components $C \subset \{1, 2, ...\}$ are non-random. Let the components C have no noise, and let the components C^C be drawn from a noise generating distribution. Let $\mathcal{P}$ be the optimal partitioning of the data considering only the non-random components, and let $\mathcal{Q}$ be any other partitioning. Then

$$\mathbb{P}\Big(\text{cost}\Big(\mathcal{X}^{(p)}, \mathcal{P}\Big) - \text{cost}\Big(\mathcal{X}^{(p)}, \mathcal{Q}\Big) \leq 0\Big) \;\rightarrow\; \frac{1}{2} \tag{6.94}$$

as $p \rightarrow \infty$.

Proof. The proof follows directly from theorem 6.1.9. Because there are only finitely many non-noisy components of $\mathcal{X}^{(p)}$, we have trivially that

$$\Big(\mathbf{x}_i^{(p)}\Big)^T \mathbf{x}_j^{(p)} \in \text{o}(\sqrt{p}). \tag{6.95}$$

for all $\mathbf{x}_i^{(p)}, \mathbf{x}_j^{(p)} \in \mathcal{X}^{(p)}$. Furthermore, by the Kolmogorov 0-1 law, setting finitely many components of the sequence

$$\text{cost}\Big(\mathscr{Y}^{(p)}, \mathcal{P}\Big) - \text{cost}\Big(\mathscr{Y}^{(p)}, \mathcal{Q}\Big),\; p = 1, 2, ... \tag{6.96}$$

to 0 cannot affect the convergence of the series. The theorem is proved. □

6.2 Behavior of the Averaged Assignment Matrix

This section concerns general results on the partial membership matrix, i.e. what happens under various assumptions on the inputs.

For convenience, we sometimes use the functional notation for $\mathbf{\Phi}$ (equation (3.6)). Restricting ourselves to hard clusterings, we have

$$\phi_{ij} = \psi_j(\mathbf{d}_i, \theta) = \int \mathbb{I}\{d_{ij}\lambda_j \leq d_{i\ell}\lambda_\ell \;\; \forall\, \ell \neq j\}\pi(\boldsymbol{\lambda}; \theta)\mathrm{d}\boldsymbol{\lambda} \tag{6.97}$$

Alternately, we extend some of our proofs to a more general form of $\mathbf{\Phi}$, where

$$\phi_{ij} = \psi_j(\mathbf{d}_i, \mathbf{g}_i, \theta) = \int \mathbb{I}\{d_{ij}\lambda_j + g_{ij} \leq d_{i\ell}\lambda_\ell + g_{i\ell} \ \forall\, \ell \neq j\}\pi(\boldsymbol{\lambda}; \theta)\mathrm{d}\boldsymbol{\lambda} \tag{6.98}$$

The advantage of using this notation is that it can also apply when the priors between different λ_ℓ's differ by a scaling parameter and a constant. In other words, if

$$\pi_\ell(\lambda_\ell) \overset{\mathrm{D}}{=} \pi\left(\frac{\lambda_\ell - c_\ell}{\beta_\ell}\right) \tag{6.99}$$

for some constants β_ℓ and c_ℓ, then

$$\begin{aligned}\int \mathbf{1}_{d_j\lambda_j \leq d_\ell\lambda_\ell \ \forall\, \ell \neq j} \prod_\ell \pi_\ell(\lambda_\ell)\mathrm{d}\lambda_\ell \\ = \int \mathbf{1}_{d_j\beta_j\lambda_j + c_j d_j \leq d_\ell\beta_\ell\lambda_\ell + c_\ell d_\ell \ \forall\, \ell \neq j} \prod_\ell \pi(\lambda_\ell)\mathrm{d}\lambda_\ell\end{aligned} \tag{6.100}$$

which differs only notationally from equation (6.97). Thus the proofs that use this form will also apply to the location exponential prior and the shifted Gamma prior from chapter 3 where the slope or scale parameters differ between clusters.

Because our proofs often treat $\mathbf{d}_i$, the input vector of point-to-centroid distances, as a random vector $\mathbf{D}_i$, we denote the random form of ϕ_{ij} as H_{ij}. Formally,

$$H_{ij} = \psi(\mathbf{D}_i, \mathbf{G}_i, \theta) \tag{6.101}$$

Alternately, we generalize some of the results to the case when ψ_j takes both a distance measure $\mathbf{d}_i$ and additive parameters $\mathbf{g}_i$.

6.2.1 Convergence of $\mathbf{\Phi}$ as a function of Random Variables

We present here a lemma proving that the entries of our partial membership matrix $\mathbf{\Phi}$, as given by equation (3.1), converge asymptotically to non-random values under mild assumptions on the inputs. We use more general version of $\mathbf{\Phi}$:

$$\begin{aligned}\mathbf{\Phi} &= [\phi_{ij}]_{i=1,\ldots,n;j=1,\ldots,K} \\ \phi_{ij} &= \int_{\mathbf{\Lambda}} \mathbb{I}\{d_{ij}\lambda_j + g_{ij} \leq d_{i\ell}\lambda_\ell + g_{i\ell} \ \forall\, \ell \neq j\}\pi(\boldsymbol{\lambda})\mathrm{d}\boldsymbol{\lambda} \end{aligned} \tag{6.102}$$

For this lemma, we replace the d's and g's with converging sequences of random variables and show that ϕ_{ij} will also converge. For notational conciseness, we omit the i index in the following proof; the result applies for each row of $\mathbf{\Phi}$.

Lemma 6.2.1. *Suppose $D_{j1}, D_{j2}, \ldots$ and $G_{j1}, G_{j2}, \ldots$, $j = 1, 2, \ldots, K$, are sequences of random variables such that*

$$\frac{D_{jp}}{p^\alpha} \xrightarrow{\mathbb{P}} \delta_j \tag{6.103}$$

$$\frac{G_{jp} - G_{\ell p}}{p^\alpha} \xrightarrow{\mathbb{P}} \gamma_{j\ell} \tag{6.104}$$

as $p \to \infty$ for some $\delta_j \neq 0$, $\gamma_{j\ell} \in \mathbb{R}$, and $\alpha \in \mathbb{R}$. Let

$$H_{jp} = \int \mathbb{I}[\lambda_j D_{jp} + G_{jp} \leq \lambda_\ell D_{\ell p} + G_{\ell p} \ \forall\, \ell \neq j]\pi(\boldsymbol{\lambda})\mathrm{d}\boldsymbol{\lambda}. \tag{6.105}$$

Then if $0 \leq \pi(\boldsymbol{\lambda}) \leq C < \infty \ \forall\, \boldsymbol{\lambda} \in \mathbb{R}^K$,

$$H_{jp} \xrightarrow{\mathbb{P}} \int \mathbb{I}[\lambda_j \delta_j \leq \delta_\ell \lambda_\ell + \gamma_{j\ell} \ \forall\, \ell \neq j]\pi(\boldsymbol{\lambda})\mathrm{d}\boldsymbol{\lambda} \tag{6.106}$$

as $p \to \infty$.

Proof. Let $\boldsymbol{\mathcal{S}}_j$ be the function taking two $K \times K$ matrices as input and returning a set on $\mathbb{R}^K$ that is defined by

$$\boldsymbol{\mathcal{S}}_j\Big([y_{j\ell}]_{j,\ell=1,2,...,K}\,,\ [z_{j\ell}]_{i,j=1,2,...,K}\Big) = \{\boldsymbol{\lambda} : \lambda_j \leq y_{j\ell}\lambda_\ell + z_{j\ell}\ \ \forall\, \ell \neq j\}\,. \tag{6.107}$$

We can rewrite the indicator function in the definition of H_{jp} to use this function:

$$\begin{aligned}
&\mathbb{I}\{\lambda_j D_{jp} + G_{jp} \leq \lambda_\ell D_{\ell p} + G_{\ell p}\} \\
&\qquad = \mathbb{I}\left\{\lambda_j \leq \lambda_\ell \frac{D_{\ell p}}{D_{jp}} + \frac{G_{\ell p} - G_{jp}}{D_{jp}}\right\} &(6.108)\\
&\qquad = \mathbb{I}\left\{\boldsymbol{\lambda} \in \boldsymbol{\mathcal{S}}_j\left(\left[\frac{D_{\ell p}}{D_{jp}}\right], \left[\frac{G_{\ell p} - G_{jp}}{D_{jp}}\right]\right)\right\} &(6.109)
\end{aligned}$$

Let

$$\mathbf{Y}_p = \left[\frac{D_{\ell p}}{D_{jp}}\right]_{i,j=1,2,...,K} \tag{6.110}$$

$$\mathbf{Z}_p = \left[\frac{G_{\ell p} - G_{jp}}{D_{jp}}\right]_{i,j=1,2,...,K}. \tag{6.111}$$

Then

$$H_{jp} = \int_{\boldsymbol{\lambda} \in \boldsymbol{\mathcal{S}}_j(\mathbf{Y}_p, \mathbf{Z}_p)} \pi(\boldsymbol{\lambda}) \mathrm{d}\boldsymbol{\lambda}. \tag{6.112}$$

Now let

$$\mathbf{Y}_\infty = [\delta_\ell / \delta_j] \tag{6.113}$$

$$\mathbf{Z}_\infty = [\gamma_{j\ell} / \delta_j] \tag{6.114}$$

Since $D_{jp} > 0$ for $j = 1, ..., K$,

$$\mathbf{Y}_p = \left[\frac{D_{\ell p}}{D_{jp}}\right] = \left[\frac{D_{\ell p}/p^\alpha}{D_{jp}/p^\alpha}\right] \xrightarrow{\mathbb{P}} [\delta_\ell/\delta_j] = \mathbf{Y}_\infty \tag{6.115}$$

as $p \to \infty$. Similarly,

$$\mathbf{Z}_p = \left[\frac{G_{\ell p} - G_{jp}}{D_{jp}}\right] = \left[\frac{(G_{\ell p} - G_{jp})/p^\alpha}{D_{jp}/p^\alpha}\right] \xrightarrow{\mathbb{P}} [\gamma_{j\ell}/\delta_j] = \mathbf{Z}_\infty \tag{6.116}$$

as $p \to \infty$. Now $\pi(\boldsymbol{\lambda}) \leq C < \infty \ \forall\, \boldsymbol{\lambda} \in \mathbb{R}^K$, so the function

$$f(\mathbf{Y}_p, \mathbf{Z}_p) = \int_{\boldsymbol{\lambda} \in \boldsymbol{\mathcal{S}}_j(\mathbf{Y}_p, \mathbf{Z}_p)} \pi(\boldsymbol{\lambda}) \mathrm{d}\boldsymbol{\lambda} \tag{6.117}$$

is a continuous function of the entries in $\mathbf{Y}_p$ and $\mathbf{Z}_p$. Furthermore, since $\int \pi(\boldsymbol{\lambda}) \mathrm{d}\boldsymbol{\lambda} = 1$, $H_{jp} \leq 1$. Thus we have

$$H_{jp} = \int_{\boldsymbol{\lambda} \in \boldsymbol{\mathcal{S}}_j(\mathbf{Y}_p, \mathbf{Z}_p)} \pi(\boldsymbol{\lambda}) \mathrm{d}\boldsymbol{\lambda} \xrightarrow{\mathbb{P}} \int_{\boldsymbol{\lambda} \in \boldsymbol{\mathcal{S}}_j(\mathbf{Y}_\infty, \mathbf{Z}_\infty)} \pi(\boldsymbol{\lambda}) \mathrm{d}\boldsymbol{\lambda} \tag{6.118}$$

as $p \to \infty$, which completes the proof. □

6.3 Asymptotic Behavior of Φ in High Dimensions

In this section, we both show the theoretical limits of our measure in high dimensions under certain conditions. Effectively, we argue that if the clustering is not sensible, then our method will produce a uniform result, indicating no stability. Ultimately, we examine the case where we hold the number of points n fixed as we let the dimension p in which the points are located increase to infinity.

This type of assymptotics has received some attention in the literature [HMN05, HAK00]; however, it is fairly limited. Perhaps the lack of attention to this type of assymptotics is due in part to the fact that n points lie in a subspace of dimension at most $n - 1$, which, in many cases, means the results of such assymptotics is not applicable. However, in our case, we are building our proofs on distances between points, so all of them hold under an

arbitrary rotation of the coordinate system. This includes the PCA rotation, which projects the original n p-dimensional points, where $p \geq n$, into a $n-1$ dimensional subspace.

ANOVA Type Data

For these results, we make the ANOVA assumption. Specifically, we assume that the data is distributed according to a mixture model of K Gaussians defined by a set of centers $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, ..., \boldsymbol{\mu}_K$ and a corresponding set of diagonal covariance $\Sigma_1, \Sigma_2, ..., \Sigma_K$ matrices, where $\Sigma_j = \text{diag}\left(\sigma^2_{j\,1}, \sigma^2_{j\,2}, ..., \sigma^2_{j\,p}\right)$ and p is the dimension of the space.

In examining the behavior of the partial membership matrix $\boldsymbol{\Phi}$ as the dimension p increases, we examine the best case scenario regarding the clustering. We assume that the clustering algorithm perfectly partitions the dataset – that is, we assume that every point in a particular partition element has been drawn from the same mixture model component, and every point drawn from the same component ends up in the same partition. Because of this one-to-one assumption, we refer to all the points drawn from a component as a cluster. While both the ANOVA assumption and this are fairly strong assumptions, and unrealistic in many contexts, they do allow us to demonstrate some interesting properties of our stability measure.

Throughout this section, we use the notation defined in tables (6.1) and (6.2). The first table defines general clustering notation, whereas the second refers specifically to the ANOVA style mixture models we base our results on.

6.3.1 Asymptotic Behavior of Φ with ANOVA-type Mixture Models

The following theorems describe the asymptotic behavior of $\boldsymbol{\Phi}$ under the ANOVA assumption, described above, as $p \to \infty$. We assume the notational definitions as given in tables (6.1) and (6.2). Also, for these results, we treat the elements in $\boldsymbol{\Phi}$ as random variables, so we define $H^{(p)}_{ij}$ to be the result of

p	The dimension.
$\mathbf{X}_i^{(p)}$	A p-dimensional data point.
X_{iq}	The qth component of $\mathbf{X}_i^{(p)}$.
N	The number of data points in total.
K	The number of mixture model components.
C_j	The set of the indices for all points drawn from component j.
n_j	The number of points in cluster j.
$\mathcal{C}$	A mapping function that gives the index of the point a cluster is assigned to, i.e. $\mathcal{C}(i) = j$ s.t. $i \in C_j$.
i	Index used to index data points.
j, ℓ	Indices used to index clusters or mixture model components.

Table 6.1: Notational conventions used for asymptotic proofs of mixture models in increasing dimension.

$\boldsymbol{\mu}^{(\mathbf{p})}{}_j$	The mean of the jth mixture component in the underlying distribution.
$\mu_{jq}^{(p)}$	The mean of the qth dimension of jth mixture component in the underlying distribution.
σ_{jq}^2	The variance of the qth dimension of the jth mixture component.
τ_{jq}^2	The average variance of the first q dimensions of the jth mixture component, equal to $\frac{1}{q}\sum_{q'=1}^{q}\sigma_{jq'}^2$
τ_j^2	The limit of τ_{jq}^2 as $q \to \infty$.
$\bar{\mathbf{X}}_j^{(p)}$	The empirical mean of the jth cluster in p dimensions, equal to $\frac{1}{n_j}\sum_{i \in C_j}\mathbf{X}_j^{(p)}$.

Table 6.2: Notational conventions used for asymptotic proofs of ANOVA mixture models.

$$H_{ij}^{(p)} = \int_{\boldsymbol{\lambda}\in\mathbb{R}^K} \pi(\boldsymbol{\lambda})\, \mathbb{I}\Big\{\lambda_j D_{ij}^{(p)} \leq \lambda_\ell D_{i\ell}^{(p)} \;\; \forall\, \ell \neq j\Big\} \mathrm{d}\boldsymbol{\lambda}. \tag{6.119}$$

where

$$D_{ij}^{(p)} = \|\mathbf{X}_i^{(p)} - \bar{\mathbf{X}}_j^{(p)}\|_2 \tag{6.120}$$

is the distance between the ith data point and the jth cluster center, defined as a random variable. We use the superscript (p) to indicate the dimension of the vector. Furthermore, let $\nu_{j\ell}$ be the distance between the jth and ℓth mixture component centers:

$$\nu_{j\ell} = \lim_{p \to \infty} \frac{\|\mu_j^{(p)} - \mu_\ell^{(p)}\|_2}{\sqrt{p}}, \tag{6.121}$$

and let δ_{ij} be defined in terms of the component properties as follows:

$$\delta_{ij} = \begin{cases} \sqrt{\tau_{\mathcal{C}(i)}^2 - \tau_j^2/n_j} & j = \mathcal{C}(i) \\ \sqrt{\tau_{\mathcal{C}(i)}^2 + \tau_j^2/n_j + \nu_{j,\mathcal{C}(i)}^2} & j \neq \mathcal{C}(i) \end{cases} \tag{6.122}$$

This factor plays a key role in our analysis.

Lemma 6.3.1. *As $p \to \infty$,*

$$\frac{D_{ij}^{(p)}}{\sqrt{p}} \xrightarrow{\mathbb{P}} \delta_{ij} \tag{6.123}$$

Proof. By definition of the L2-norm, we can rewrite $D_{ij}^{(p)}/\sqrt{p}$:

$$\frac{D_{ij}^{(p)}}{\sqrt{p}} = \sqrt{\sum_{q=1}^{p} \frac{\left(X_{iq} - \bar{X}_{jq}\right)^2}{p}} \tag{6.124}$$

Now, by definition, we know that

$$X_{iq} \sim \mathcal{N}\Big(\mu_{\mathcal{C}(i),q}, \sigma_{\mathcal{C}(i),q}^2\Big) \tag{6.125}$$

$$\Rightarrow \bar{X}_{\mathcal{C}(i),q} \sim \mathcal{N}\Big(\mu_{\mathcal{C}(i),q}, \sigma_{\mathcal{C}(i),q}^2/n_j\Big). \tag{6.126}$$

Note that between values of q we have independence. Thus the strong law of large numbers gives us

$$\sum_{q=1}^{p} \frac{\left(X_{iq} - \bar{X}_{jq}\right)^2}{p} \xrightarrow{\mathbb{P}} \lim_{p \to \infty} \sum_{q=1}^{p} \frac{\mathbb{E}\left(X_{iq} - \bar{X}_{jq}\right)^2}{p} \tag{6.127}$$

We will ultimately evaluate this as two cases, when $\mathbf{x}_i$ is a member of cluster j and when it is not; the result will be different in the two cases.

Now the square root is a continuous function, so the above implies that

$$\sqrt{\sum_{q=1}^{p} \frac{\left(X_{iq} - \bar{X}_{jq}\right)^2}{p}} \xrightarrow{\mathbb{P}} \lim_{p \to \infty} \sqrt{\sum_{q=1}^{p} \frac{\mathbb{E}\left(X_{iq} - \bar{X}_{jq}\right)^2}{p}} \tag{6.128}$$

Now let

$$\mathbf{Y}_i = (Y_{i1}, Y_{i2}, ..., Y_{ip}) = \mathbf{X}_i^{(p)} - \mu_{\mathcal{C}(i)}^{(p)} \tag{6.129}$$

$$\mathbf{Z}_j = (Z_{j_1}, Z_{j_2}, ..., Z_{j_p}) = \bar{\mathbf{X}}_j^{(p)} - \mu_j^{(p)}. \tag{6.130}$$

We now need to worry about two cases – case 1, when $j \neq \mathcal{C}(i)$, and case 2, when $j = \mathcal{C}(i)$.

Case 1: $j \neq \mathcal{C}(i)$

Since $j \neq \mathcal{C}(i)$, $\mathbf{X}_i^{(p)}$ and $\mu_{\mathcal{C}(i)}^{(p)}$ are independent from $\bar{\mathbf{X}}_j^{(p)}$. Thus

$$\begin{aligned}
\mathbb{E}\left(X_{iq} - \bar{X}_{jq}\right)^2 &= \mathbb{E}\left(\mu_{\mathcal{C}(i),q} + Y_{iq} - \mu_{jq} - Z_{jq}\right)^2 && (6.131)\\
&= \left(\mu_{\mathcal{C}(i),q} - \mu_{jq}\right)^2 + 2\left(\mu_{\mathcal{C}(i),q} - \mu_{jq}\right)\left(\mathbb{E}\, Y_{iq} - \mathbb{E}\, Z_{jq}\right) \\
&\qquad + \mathbb{E}\left(Y_{iq} - Z_{jq}\right)^2 && (6.132)\\
&= \left(\mu_{\mathcal{C}(i),q} - \mu_{jq}\right)^2 + \mathbb{E}\, Y_{iq}^2 - 2\,\mathbb{E}\, Y_{iq}\,\mathbb{E}\, Z_{jq} + \mathbb{E}\, Z_{jq}^2 && (6.133)\\
&= \left(\mu_{\mathcal{C}(i),q} - \mu_{jq}\right)^2 + \sigma_{\mathcal{C}(i),q}^2 + \sigma_{jq}^2/n_j && (6.134)
\end{aligned}$$

We can use this to find the limit of equation (6.124).

$$\Rightarrow \frac{D_{ij}^{(p)}}{\sqrt{p}} \stackrel{P}{=} \sqrt{\frac{1}{p}\sum_{q=1}^{p}\left(X_{iq}-\bar{X}_{jq}\right)^2} \tag{6.135}$$

$$\stackrel{\mathbb{P}}{\longrightarrow} \sqrt{\sigma^2_{\mathcal{C}(i)}+\sigma_j^2/n_j+\frac{1}{p}\sum_{q=1}^{p}\left(\mu_{\mathcal{C}(i),q}-\mu_{jq}\right)^2} \tag{6.136}$$

$$= \sqrt{\tau^2_{\mathcal{C}(i)}+\tau_j^2/n_j+\nu^2_{j,\mathcal{C}(i)}} \tag{6.137}$$

Case 2: $j=\mathcal{C}(i)$

Now if $j=\mathcal{C}(i)$, then $\mathbf{X}_i^{(p)}$ and $\mu_j^{(p)}$ are not independent:

$$\mathbb{E}\left(X_{iq}-\bar{X}_{jq}\right)^2 = \mathbb{E}\left(X_{iq}-\tfrac{1}{n_j}\textstyle\sum_{i'\in Cj}X_{i'q}\right)^2 \tag{6.138}$$

$$= \mathbb{E}\left(\mu_{jq}+Y_{jq}-\tfrac{1}{n_j}\textstyle\sum_{i'\in C_j}\mu_{jq}+Y_{i'q}\right)^2 \tag{6.139}$$

$$= \mathbb{E}\left(Y_{jq}-\tfrac{1}{n_j}\textstyle\sum_{i'\in C_j}Y_{i'q}\right)^2 \tag{6.140}$$

$$= \mathbb{E}\,Y_{jq}^2+\tfrac{1}{n_j}\mathbb{E}\left(\textstyle\sum_{i'\in C_j}Y_{i'q}\right)^2-2\tfrac{1}{n_j}\mathbb{E}\left[Y_{iq}\textstyle\sum_{i'\in C_j}Y_{i'q}\right] \tag{6.141}$$

Now Y_{iq} and $Y_{i'q}$ are independent for $i\neq i'$, and $\mathbb{E}\,Y_{iq}=0$, so many of these terms drop out:

$$\Rightarrow \mathbb{E}\left(X_{iq}-\bar{X}_{jq}\right)^2 = \mathbb{E}\,Y_{jq}^2+\tfrac{1}{n_j}\mathbb{E}\left[\textstyle\sum_{i'\in C_j}Y_{i'q}^2\right]-2\,\mathbb{E}\,\tfrac{1}{n_j}Y_{iq}^2 \tag{6.142}$$

$$= \sigma_{jq}^2+\tfrac{\sigma_{jq}^2}{n_j}-2\tfrac{\sigma_{jq}^2}{n_j} \tag{6.143}$$

$$= \sigma^2_{\mathcal{C}(i),q}-\sigma_{jq}^2/n_j\ . \tag{6.144}$$

Thus

$$\frac{D_{i,\mathcal{C}(i)}^{(p)}}{\sqrt{p}} \stackrel{P}{=} \sqrt{\tfrac{1}{p}\textstyle\sum_{q=1}^{p}\left(X_{iq}-\bar{X}_{\mathcal{C}(i),q}\right)^2} \tag{6.145}$$

$$\stackrel{\mathbb{P}}{\longrightarrow} \sqrt{\tau^2_{\mathcal{C}(i)}-\tau_j^2/n_j}\,. \tag{6.146}$$

We now have that

$$\frac{D_{ij}^{(p)}}{\sqrt{p}} \xrightarrow{\mathbb{P}} \delta_{ij} \tag{6.147}$$

which completes the lemma. □

Theorem 6.3.2. *Suppose*

$$H_j^{(p)} = \int_{\boldsymbol{\lambda}\in\mathbb{R}^K} \pi(\boldsymbol{\lambda})\, \mathbb{I}\Big\{\lambda_j D_j^{(p)} + G_j^{(p)} \leq \lambda_\ell D_\ell^{(p)} + G_\ell^{(p)} \;\; \forall\, \ell \neq j\Big\} \mathrm{d}\boldsymbol{\lambda}. \tag{6.148}$$

Then, as $p \to \infty$,

$$H_j^{(p)} \xrightarrow{\mathbb{P}} \int_{\boldsymbol{\lambda}\in\mathbb{R}^K} \pi(\boldsymbol{\lambda})\, \mathbb{I}\{\lambda_j\delta_j \leq \lambda_\ell\delta_\ell + \nu_{j\ell} \;\; \forall\, \ell \neq j\} \mathrm{d}\boldsymbol{\lambda} \tag{6.149}$$

Proof. From lemma 6.3.1, we have that $D_{ij}/\sqrt{p} \xrightarrow{\mathbb{P}} \delta_{ij}$. The theorem then follows directly from lemma 6.2.1, and $\alpha = 0.5$. □

6.3.2 An Impossibility Theorem for Φ in High Dimensions

Using these lemmas, we can get a non-trivial impossibility theorem for our stability measure, essentially saying that unless the distance between cluster centers grows at a rate $\Omega\big(\sqrt{p}\big)$ as the dimension p increases, the partial point assignment matrix decays to an uninformative uniform.

Theorem 6.3.3. *If*

$$\|\mu_j^{(p)} - \mu_\ell^{(p)}\|_2 \in \mathrm{o}(\sqrt{p}) \tag{6.150}$$

then

$$H_{ij}^{(p)} \xrightarrow{\mathbb{P}} \frac{1}{K} \tag{6.151}$$

as $p \to \infty$ *and* $n_j \to \infty$.

Proof. Let $p \to \infty$ first. We have, by definition, that

$$\nu_{j\ell} = \lim_{p \to \infty} \frac{\|\mu_j^{(p)} - \mu_\ell^{(p)}\|_2}{\sqrt{p}} = 0. \tag{6.152}$$

By lemma 6.3.1,

$$\frac{D_{ij}^{(p)}}{\sqrt{p}} \xrightarrow{\mathbb{P}} \delta_{ij} \tag{6.153}$$

where

$$\delta_{ij} = \begin{cases} \sqrt{\tau_{\mathcal{C}(i)}^2 - \tau_j^2/n_j} & j = \mathcal{C}(i) \\ \sqrt{\tau_{\mathcal{C}(i)}^2 + \tau_j^2/n_j} & j \neq \mathcal{C}(i) \end{cases}. \tag{6.154}$$

In the large sample limit, where $n_j \to \infty$, this becomes

$$\frac{D_{ij}^{(p)}}{\sqrt{p}} \xrightarrow{\mathbb{P}} \bar{\sigma}_{\mathcal{C}(i)} = \bar{\sigma} \tag{6.155}$$

where for notational convenience we drop the i's as they are constant through the rest of the proof.

Now by theorem 6.3.2, we have that

$$\begin{aligned} H_{ij}^{(p)} &\xrightarrow{\mathbb{P}} \int_{\boldsymbol{\lambda} \in \mathbb{R}^K} \pi(\boldsymbol{\lambda})\, \mathbb{I}\{\lambda_j \delta_j \leq \lambda_\ell \delta_\ell \ \forall\, \ell \neq j\} \mathrm{d}\boldsymbol{\lambda} \\ &= \mathbb{P}(\lambda_j \delta_j \leq \lambda_\ell \delta_\ell \ \forall\, \ell \neq j) \\ &= \mathbb{P}(\lambda_j \bar{\sigma} \leq \lambda_\ell \bar{\sigma} \ \forall\, \ell \neq j) \\ &= \mathbb{P}(\lambda_j \leq \lambda_\ell \ \forall\, \ell \neq j) \end{aligned}$$

which is just the probability that λ_j is the lowest in a set of K i.i.d. random variables. This probability is $1/K$.

□

6.4 Prior Behavior

Now that we have this result, we can make some useful observations relating to the type of prior we should use. The next theorem shows that we can design a prior capable of making the partial membership of the indicator function sensible provided that the distances themselves are distinguishable.

Theorem 6.4.1. *Suppose that* $d_j < d_\ell \ \forall \, \ell \neq j$. *Then* $\forall \, \eta > 0$, *a prior* π *exists such that*

$$\phi_{i,\mathcal{C}(i)} - \max_{\ell \neq \mathcal{C}(i)} \phi_{i,\ell} \geq 1 - \varepsilon \tag{6.156}$$

for ε *sufficiently small.*

Proof. The proof follows directly from lemma 3.6.1.

Let $\pi(\boldsymbol{\lambda})$ be the location exponential distribution. Recall that $\forall \, \varepsilon \in (0,1)$ and $S \subset \{1, ..., K\}$, if $j \notin S$ and $d_k \geq d_j \Big(1 - \frac{\log \varepsilon}{\theta}\Big)$ for all $k \in S$, then $\sum_{k \in S} \phi_k(\mathbf{d}, \boldsymbol{\theta}) \leq \varepsilon$. Thus, if we using the location exponential prior with the slope θ set so

$$\theta \geq (-\log \varepsilon) \left(\frac{1}{\frac{d_k}{d_j} - 1} \right), \tag{6.157}$$

we have that

$$\sum_{\ell \neq j} \phi_\ell(\mathbf{d}, \boldsymbol{\theta}) \leq \varepsilon \tag{6.158}$$

If this is the case, we know that

$$\phi_{i,\mathcal{C}(i)} - \max_{\ell \neq \mathcal{C}(i)} \phi_{i,\ell} \geq 1 - \varepsilon, \tag{6.159}$$

which completes the proof. □

Chapter 7

Additional Relevant Research

7.1 Validation Algorithms involving Reclustering

In this section we consider options for efficiently calculating equation (3.1) when the perturbation introduced requires the data to be reclustered. In this case, we present an algorithm that may enable such a calculation to be computationally feasible. Our starting assumption is that the computational expense of initially creating a clustering solution greatly outweighs the computational expense of modifying a clustering with a close value of the perturbation hyperparameter. While the motivating case for the algorithm we propose is from clustering, other functions in this class are easy to imagine. For example, many can be evaluated only through an iterative procedure where an approximate solution can quickly be refined to one with a desired level of accuracy. Therefore, it makes sense to look at the general formulation of the problem instead of the specific motivating example of clustering.

The problem of calculating equation (3.1) is essentially the problem of calculating the normalizing constant of a posterior distribution, which we formulate as

$$Z = \int s(\theta)p(\theta)\mathrm{d}\theta. \tag{7.1}$$

The solution to this problem is a difficult problem, especially in high dimensions, and it has received some attention in the Monte Carlo literature.

7.1.1 Monte Carlo Abstraction

As Monte Carlo methods are applied to increasingly complex and non-standard functions, designing MC algorithms that take the complexities into account are increasingly important. Additionally, one of the more challenging problems for which MC is used is evaluating normalizing constants in high-dimensional space. Numerous techniques for doing so have been proposed, including kernel density based methods, importance sampling methods, and using the harmonic mean [Dou07]. By and large, these methods break down in higher dimensions, but bridge sampling, path sampling [GM98], and linked importance sampling [Nea05] have shown significant improvements. Even so, it is still an open problem and will likely remain so for some time.

The algorithm we propose is a modification of path sampling. In section section 7.1.2, we review path sampling and how the method can be used to estimate the value of a normalizing constant. In section 7.1.2 we describe sweep sampling as an extension to path sampling, present computational results in 7.1.3, and discuss the strengths and limitations of path sampling in 7.1.4.

7.1.2 Evaluating the Normalizing Constant

Of the many proposed Monte Carlo techniques for evaluating $Z = \int \gamma(\theta)\mathrm{d}\theta$, path sampling [GM98] remains one of the most robust. Most other methods, including kernel-based methods, simple importance sampling, or harmonic mean techniques, break down in high dimensions; often the variance of the result increases exponentially with the number of dimensions.

Path Sampling

The idea behind path sampling is to start from a distribution $p_0(\theta)$, from which we can readily sample and for which we know the normalizing constant Z_0. We then construct a smooth path from $p_0(\theta)$ to the target distribution $p_1(\theta)$, requiring the probability to be nonzero for all $\theta \in \mathbf{\Theta}$ and $\alpha \in [\alpha_{start}, \alpha_{end}]$. For example, the geometric path – which we employ later in this paper – is

$$\gamma(\theta|\alpha) = p_0(\theta)^{1-\alpha} p_1(\theta)^{\alpha} \tag{7.2}$$

where we index the path by $\alpha \in [0, 1]$ and denote the (unnormalized) distribution along the path by $\gamma(\theta|\alpha)$. Given such a path, the log ratio of the two normalizing constants is given by the path sampling identity:

$$\begin{aligned} \lambda = \log \frac{Z(\alpha = \alpha_{end})}{Z(\alpha = \alpha_{start})} &= \int_{\alpha_{start}}^{\alpha_{end}} \int_{\mathbf{\Theta}} \left[\frac{\mathrm{d}}{\mathrm{d}\alpha} \log \gamma(\theta|\alpha) \right] \pi(\theta|\alpha) \mathrm{d}\theta \, \mathrm{d}\alpha \\ &= \int_{\alpha_{start}}^{\alpha_{end}} G(\alpha) \mathrm{d}\alpha \end{aligned} \tag{7.3}$$

where $\pi(\theta|\alpha) = \gamma(\theta|\alpha)/Z(\alpha)$ is the normalized version of $\gamma(\theta|\alpha)$, i.e.

$$\int \pi(\theta|\alpha) \mathrm{d}\theta = 1. \tag{7.4}$$

Note that the geometric path as described in equation (7.2) uses $\alpha_{start} = 0$ and $\alpha_{end} = 1$.

To estimate (7.3) using Monte Carlo, we first break the integral over α into a set L discrete values:

$$A = \{\alpha_{start} = \alpha^{(1)} < \alpha^{(2)} < ... < \alpha^{(L)} = \alpha_{end}.\} \tag{7.5}$$

For most applications, a uniformly spaced grid is sufficient, though [GM98] investigates sampling the $\alpha^{(j)}$ from an optimal prior distribution $p(\alpha)$.

Sampling

Given an $\alpha^{(j)} \in A$, we can estimate $G(\alpha^{(j)})$ in equation (7.3) by obtaining N i.i.d. samples from $\pi(\theta|\alpha = \alpha^{(j)})$ and evaluating

$$\widehat{G(\alpha^{(j)})} = \frac{1}{N}\sum_{i=1}^{N} \frac{\mathrm{d}}{\mathrm{d}\alpha} \log \gamma(\theta^{(i)}|\alpha)\Big|_{\alpha=\alpha^{(j)}} \tag{7.6}$$

In general, we can only sample from $p(\theta|\alpha = \alpha_{start})$, but we can obtain i.i.d. samples for $p(\theta|\alpha^{(j)})$ by constructing an MCMC chain from $\gamma(\theta|\alpha = \alpha_{start})$ to $\gamma(\theta|\alpha = \alpha_{end})$ that redistributes the samples from $p(\theta|\alpha^{(j-1)})$ according to $p(\theta|\alpha^{(j)})$ using a Metropolis-Hastings random walk [AdFDJ03]. We can then obtain an estimate of $\hat{\lambda}$ by performing numerical integration on the sequence $\widehat{G(\alpha^{(1)})}, \widehat{G(\alpha^{(2)})}, ..., \widehat{G(\alpha^{(L)})}$. Numerous possible methods exist for doing this, e.g. trapezoidal or cubic spline integration.

Sweep Sampling

Here we propose here an extension to path sampling that allows for computationally expensive distributions. We can divide the region $\mathbf{\Theta}$ into R distinct subregions $\mathbf{\Theta}_1, \mathbf{\Theta}_2, ..., \mathbf{\Theta}_R$, allowing us write equation (7.3) as:

$$\lambda = \sum_r \int_{\alpha_{start}}^{\alpha_{end}} \int_{\mathbf{\Theta}_r} \left[\frac{\mathrm{d}}{\mathrm{d}\alpha} \log \gamma(\theta|\alpha)\right] \pi(\theta|\alpha)\mathrm{d}\theta\mathrm{d}\alpha. \tag{7.7}$$

where notationally, r indexes the regions. Furthermore, let $\rho(\theta)$ denote the index of the region θ is in (i.e. $\rho(\theta) = r \Leftrightarrow \theta \in \mathbf{\Theta}_r$).

Informally, the idea behind sweep sampling is to hold the regions $\mathbf{\Theta}_r$ at the proposal distribution until the target distribution becomes computationally feasible. Initially, at $\alpha = \alpha_{start}$, we define $\mathbf{\Theta}_1$ as a given collection of one or more seed points where $s(\theta)$ has been solved. As soon as we define $\mathbf{\Theta}_r$, we start the region on a path to the target distribution. We store a solution to $s(\theta)$ for $\theta \in \mathbf{\Theta}_r$, which allows $s(\theta)$ to be computationally feasible for $\theta \in \mathbf{\Theta}_{r+1}$. This "domino effect" causes the transition region to sweep through $\mathbf{\Theta}$, hence the name.

Formally we can thus define our terms (table 7.1):

$\boldsymbol{\mathcal{P}}(\alpha)$:	The set of all regions still at the proposal distribution (i.e. $\theta \in \boldsymbol{\mathcal{P}}(\alpha) \Rightarrow \gamma(\theta\|\alpha) \propto p(\theta)$).
$\boldsymbol{\mathcal{Q}}(\alpha)$:	The set of all regions currently in transition from the prior to the target.
$\boldsymbol{\mathcal{T}}(\alpha)$:	The set of all regions that have arrived at the target distribution (i.e. $\theta \in \boldsymbol{\mathcal{T}}(\alpha) \Rightarrow \gamma(\theta\|\alpha) \propto s(\theta)p(\theta)$).
t_r:	The value of α at which region r begins transitioning from the proposal to the target.
α_r:	The local value of alpha, $\alpha - t_r$.
$C(\alpha)$:	A global constant factor attached to the proposal, adjusted to regulate the flow of particles across the transition region (discussed in section 7.1.2).
c_r:	The value of $C(\alpha)$ when region r begins transitioning (i.e. $c_r = C(t_r)$).

Table 7.1: The terms used in this paper.

Without loss of generality, we assume a path length of 1 for each individual region; in other words, $\gamma(\theta \in \boldsymbol{\Theta}_r|\alpha = t_r) = c_r p(\theta)$ and $\gamma(\theta \in \boldsymbol{\Theta}_r|\alpha = t_r + 1) = s(\theta)p(\theta)$. Allowing each region to follow a geometric path from proposal to target, we can define $\gamma(\theta|\alpha) \propto \pi(\theta|\alpha)$ as

$$\gamma(\theta|\alpha) = \begin{cases} C(\alpha)p(\theta) & \theta \in \boldsymbol{\mathcal{P}}(\alpha) \\ s(\theta)^{\alpha_r} c_r^{1-\alpha_r} p(\theta) & \theta \in \boldsymbol{\Theta}_r \subseteq \boldsymbol{\mathcal{Q}}(\alpha) \\ s(\theta)p(\theta) & \theta \in \boldsymbol{\mathcal{T}}(\alpha) \end{cases} \tag{7.8}$$

By breaking $\boldsymbol{\Theta}$ into the proposal, transition and post-transition regions as and sliding the interval of the transition region to $[0, 1]$, we can express (7.3) as

$$\lambda = \sum_r \int_0^1 G_r(\alpha_r)\mathrm{d}\alpha_r + \int_0^{t_r} H_r(\alpha)\mathrm{d}\alpha \tag{7.9}$$

where $\frac{\mathrm{d}}{\mathrm{d}a} \log \gamma(\theta|\alpha) = 0$ in the post-transition region and

$$G_r(\alpha_r) = \int_{\boldsymbol{\Theta}_r} \log \frac{s(\theta)}{c_r} \pi(\theta|\alpha = \alpha_r + t_r) \mathrm{d}\theta \tag{7.10}$$

$$H_r(\alpha) = \int_{\boldsymbol{\Theta}_r} \left[\frac{\mathrm{d}}{\mathrm{d}\alpha} \log C(\alpha) \right] \pi(\theta|\alpha) \mathrm{d}\theta \tag{7.11}$$

We can obtain a Monte Carlo estimate of (7.9) as described in section (7.1.2) using the same technique as for regular path sampling. We construct the sequence $A = \{\alpha^{(1)}, \alpha^{(2)}, ..., \alpha^{(L)}\}$ as in equation (7.5) on the interval $[\alpha_{start}, \alpha_{end}] = [0, 1 + \max_r t_r]$. At each value of $\alpha^{(j)}$, we can estimate (7.10) and (7.11) using MC:

$$\widehat{G_r(\alpha_r)} = \frac{1}{N} \sum_{i:\theta^{(i)} \in \boldsymbol{\Theta}_r} \log \frac{s(\theta^{(i)})}{c_r} \tag{7.12}$$

$$\widehat{H_r(\alpha)} = \frac{N_r}{N} \left. \frac{\mathrm{d}}{\mathrm{d}a} \log C(a) \right|_{a=\alpha} \tag{7.13}$$

where N_r denotes the number of samples from $\boldsymbol{\Theta}_r$. We can then evaluate λ by summing over the regions for each $\alpha^{(j)}$ and numerically integrating the resulting sequence.

Defining the Subregions

Because we do not know beforehand the precise behavior of the cost function and/or the geometry of the target distribution, we cannot, in general, determine the regions $\boldsymbol{\Theta}_1, ..., \boldsymbol{\Theta}_R$ beforehand. If we assume that the cost of moving to closer points is usually less than or equal to the cost of moving to further points, then we can define each region in terms of a set $\boldsymbol{\vartheta}_r$ of fixed points. Informally, a point θ is in region r if it is closer to at least one point in $\boldsymbol{\vartheta}_r$ than those of any other region. Formally,

$$\theta \in \boldsymbol{\Theta}_r \Leftrightarrow \underset{\phi \in \boldsymbol{\vartheta}_1 \cup \boldsymbol{\vartheta}_2 \cup ...}{\operatorname{argmin}} \; d(\theta, \phi) \in \boldsymbol{\vartheta}_r \tag{7.14}$$

where $d(\cdot, \cdot)$ is the distance metric. This will allow us to use Monte Carlo

samples to define new regions. Specifically, $\boldsymbol{\vartheta}_r$ can be a non-empty set such that

$$\boldsymbol{\vartheta}_r \subseteq \{\theta^{(i)} : \theta^{(i)} \in \boldsymbol{\mathcal{P}}(\alpha), \text{cost}(\theta^{(i)}, \phi \in \boldsymbol{\vartheta}_{1:r-1}) \leq \text{CMax}\} \tag{7.15}$$

where the user-specified parameter CMax sets the maximum cost the algorithm will consider between a current and a new evaluation of $s(\theta)$, $\theta^{(i)}$ is an existing MC particle, and $\boldsymbol{\vartheta}_{1:r}$ is shorthand for $\boldsymbol{\vartheta}_1 \cup \boldsymbol{\vartheta}_2 \cup ... \cup \boldsymbol{\vartheta}_r$. In our algorithm, we defined a new region at each $\alpha^{(j)}$ using any particles not previously reachable. Specifically,

$$\boldsymbol{\vartheta}_{r+1} = \left\{ \theta^{(i)} : \theta^{(i)} \in \boldsymbol{\mathcal{P}}(\alpha^{(j)}), \begin{array}{ll} \forall \phi \in \boldsymbol{\vartheta}_{1:r-1} & \text{cost}(\theta^{(i)}, \phi) > \text{CMax} \\ \exists \phi \in \boldsymbol{\vartheta}_{1:r} & \text{cost}(\theta^{(i)}, \phi) \leq \text{CMax} \end{array} \right\}. \tag{7.16}$$

If no such particles exist, we wait until some particles satisfy (7.16) to define a new region. Furthermore, because we use a nearest neighbor search to determine which region a particle is in, storing a solution to $s(\theta)$ for each point in $\boldsymbol{\vartheta}_r$ makes it efficient to calculate $s(\theta)$ for any MC particles in $s(\theta)$.

Calculating $\mathbf{c}(\boldsymbol{\alpha})$

While the vanilla version above method will work fine if $s(\theta)\pi(\theta)$ is roughly the same magnitude as $\pi(\theta)$, problems may arise if the target distribution is several orders of magnitude different then the proposal. When this happens, a disproportional number of samples will either leave or remain in $\boldsymbol{\mathcal{P}}(\alpha)$. We thus propose the following adaptive method as an attempt to solve this problem.

We can chose $C(\alpha)$, the scaling factor of the pre-transition region to everything else, arbitrarily. We thus construct a function that attempts to hold the average probability in the pre-transition region at roughly the same value as that in the post-transition regions. Specifically, this means:

$$\frac{1}{|\boldsymbol{\mathcal{P}}(\alpha)|}C(\alpha)\int_{\boldsymbol{\mathcal{P}}(\alpha)}p(\theta)\mathrm{d}\theta \simeq \frac{1}{|\boldsymbol{\mathcal{T}}(\alpha)|}\int_{\boldsymbol{\mathcal{T}}(\alpha)}s(\theta)p(\theta)\mathrm{d}\theta \tag{7.17}$$

where $|A| = \int_A \mathrm{d}\theta$ denotes size. Assuming $\boldsymbol{\mathcal{P}}(\alpha) \neq \emptyset$ and $\boldsymbol{\mathcal{T}}(\alpha) \neq \emptyset$, then we can construct an online estimate $\tilde{C}(\alpha = a)$ from (7.17) as follows:

$$\log \tilde{C}(\alpha = a) = \log \frac{|\boldsymbol{\mathcal{T}}(a)| \int_{\boldsymbol{\mathcal{P}}(a)} \gamma(\theta|\alpha = a)\mathrm{d}\theta}{|\boldsymbol{\mathcal{P}}(a)| \int_{\boldsymbol{\mathcal{T}}(a)} p(\theta)\mathrm{d}\theta} \tag{7.18}$$

$$= \log \frac{\int_{\boldsymbol{\mathcal{P}}(a)} \gamma(\theta|\alpha = a)\mathrm{d}\theta}{\int_{\boldsymbol{\mathcal{P}}(a)} \gamma(\theta|\alpha = 0)\mathrm{d}\theta} + \log \frac{|\boldsymbol{\mathcal{T}}(a)|}{|\boldsymbol{\mathcal{P}}(a)|}\frac{\int_{\boldsymbol{\mathcal{P}}(a)} p(\theta)\mathrm{d}\theta}{\int_{\boldsymbol{\mathcal{T}}(a)} p(\theta)\mathrm{d}\theta} \tag{7.19}$$

$$= \lambda(a) + \omega(a) \tag{7.20}$$

where

$$\omega(a) = \log \frac{|\boldsymbol{\mathcal{T}}(a)|}{|\boldsymbol{\mathcal{P}}(a)|}\frac{\int_{\boldsymbol{\mathcal{P}}(a)} p(\theta)\mathrm{d}\theta}{\int_{\boldsymbol{\mathcal{T}}(a)} p(\theta)\mathrm{d}\theta} \tag{7.21}$$

and $\lambda(a)$ is just the log ratio of the normalizing constants of the post-transition region. Restricting equation (7.9) to $\boldsymbol{\mathcal{T}}(a)$, we have

$$\lambda(a) = \sum_{r:\boldsymbol{\Theta}_r \subseteq \boldsymbol{\mathcal{T}}(a)} \int_0^1 G_r(\alpha_r)\mathrm{d}\alpha + \int_0^{t_r} H_r(\alpha)\mathrm{d}\alpha. \tag{7.22}$$

Note that in (7.22) $t_r \leq a-1$ as all regions in $\boldsymbol{\mathcal{T}}$ have reached the target. We can thus estimate using values of $G_r(\alpha_r)$ and $H_r(\alpha)$ already available. Furthermore, we can estimate $\omega(\alpha)$ by sampling $N^{\{U\}}$ particles from $\mathcal{U}n(()\boldsymbol{\Theta})$ and $N^{\{\mathrm{Pr}\}}$ particles from $\Pr(\boldsymbol{\Theta})$. An MC estimate for $\omega(a)$ is then:

$$\widehat{\omega(a)} = \log \frac{N^{\{U\}}_{\boldsymbol{\mathcal{P}}(a)} + 1}{N^{\{U\}}_{\boldsymbol{\mathcal{T}}(a)} + 1}\frac{N^{\{\mathrm{Pr}\}}_{\boldsymbol{\mathcal{T}}(a)} + 1}{N^{\{\mathrm{Pr}\}}_{\boldsymbol{\mathcal{P}}(a)} + 1} \tag{7.23}$$

Dim	N	CMax	True λ	λ from PS	SS λ (best)	SS λ (avg)	SS λ (worst)
2	400	0.1	-1.4794	-1.5711	-1.4927	-1.5182	-1.5409
3	600	0.15	-2.2191	-2.3301	-2.2699	-2.2953	-2.3109
5	1000	0.25	-3.6986	-4.0149	-3.886	-3.9502	-3.9865
8	1600	0.4	-5.9177	-6.6576	-6.3333	-7.1453	-9.4663
10	2000	0.5	-7.3972	-8.5057	-7.926	-10.3783	-12.9788
15	3000	0.75	-11.0957	-13.0993	-12.2428	-13.514	-17.2248
20	6000	2	-14.7943	-17.6714	-17.3069	-17.3296	-17.3556
30	9000	3	-22.1915	-26.7143	-25.5913	-25.6584	-25.7884

Table 7.2: The results for the target distribution with $C(\alpha) = 1$.

where we add 1 to each term to make the ratio well defined if one quantity goes to zero.

In estimating both the final $\hat{\lambda}$ or λa, we can use simple numerical differentiation on the sequence A to evaluate $\frac{\mathrm{d}}{\mathrm{d}\alpha}C(\alpha)$.

7.1.3 Computational Results

To test the accuracy of sweep sampling, we started with a uniform distribution as the proposal and used a (transition) path length of 50 (we found that results compared similarly at other path lengths). We generated a target distribution by starting with a spherical multivariate normal distribution centered at the origin and with $\sigma^2 = 0.5$. We then added 100 n-dimensional hypercubes with side-length 0.25 randomly to the distribution. This allowed us to choose the dimension and structure of the target distribution while still allowing us to compute $\int s(\theta)p(\theta)\mathrm{d}\theta$ exactly.

We compared path sampling with the geometric path (equation 7.2) with sweep sampling at a number of dimensions and found that in many cases sweep sampling gave comparably accurate results. Because MATLAB did not have the proper data structures (such as dynamic kd-trees) for parts of the sweep sampling algorithm, we could not accurately compare execution times. We defined the cost simply as the squared L2 distance. Where we found sweep sampling tended to break down was in how we defined new regions; as the dimension of the space increased, the number of total particles needs to increase exponentially to achieve a comparable rate of growth. We offset this by increasing the maximum cost, but it remains an issue and area for further study.

Various results are shown in 7.3. The results for path sampling are the

Dim	N	CMax	True λ	λ from PS	SS λ (best)	SS λ (avg)	SS λ (worst)
2	400	0.1	-1.4794	-1.5711	-1.4927	-1.5028	-1.3655
3	600	0.15	-2.2191	-2.3301	-2.2699	-2.2953	-2.3109
5	1000	0.25	-3.6986	-4.0149	-3.602	-3.9118	-3.9747
8	1600	0.4	-5.9177	-6.6576	-5.7327	-6.466	-7.722
10	2000	0.5	-7.3972	-8.4789	-7.4203	-7.749	-8.0961
15	3000	0.75	-11.0957	-13.0613	-10.0865	-10.1348	-23.8546
20	6000	2	-14.7943	-17.6515	-16.802	-17.0037	-17.2968
30	9000	3	-22.1915	-26.7133	-25.4506	-25.6692	-25.8925

Table 7.3: The results for the target distribution with $C(\alpha)$ dynamically updated.

average of two cases, as repeated results were very similar. We ran the sweep sampling tests 10 times for 2-15 dimensions and 4 times for 20 and 30 dimensions. We threw out obvious outliers (λ several orders of magnitude or more away from the median) and tabulated the results. In general, updating the constant dynamically improves the results and yielded the most accurate results.

The bias toward smaller values of the normalizing constant is likely due to the sharp "edges" in the target distribution, which pose a challenge for any method and require significantly more samples to estimate. The purpose of this simulation was to push the methods to their limits for the purpose of comparison; we suspect that a real distribution will be better behaved.

7.1.4 Discussion

This method demonstrates that sweep sampling can achieve favorable results while taking into account one type of computationally difficult distribution. However, there are still several areas for possible improvement.

For one, the estimate of C is bound to be highly noisy. One possible way of improving the algorithm, then, is to treat the estimation of $C(\alpha^{(j)})$ as a time-series control problem in which $\tilde{C}(\alpha^{(j)})$ is a noisy input. This problem is well studied, and constructing a Kalman filter [WB01], unscented Kalman filter [WVDM00], or particle filter could be a fruitful area for further study.

Bibliography

[ADD99] R.B. Ash and C.A. Doleans-Dade. *Probability and Measure Theory*. Academic Press, 1999.

[AdFDJ03] C. Andrieu, N. de Freitas, A. Doucet, and M.I. Jordan. An Introduction to MCMC for Machine Learning. *Machine Learning*, 50(1):5–43, 2003.

[ALA+03] O. Abul, A. Lo, R. Alhajj, F. Polat, and K. Barker. Cluster validity analysis using subsampling. *Systems, Man and Cybernetics, 2003. IEEE International Conference on*, 2, 2003.

[AV07] D. Arthur and S. Vassilvitskii. k-means++: the advantages of careful seeding. *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1027–1035, 2007.

[AWR96] G.B. Arfken, H.J. Weber, and L. Ruby. Mathematical Methods for Physicists. *American Journal of Physics*, 64:959, 1996.

[BB99] A. Baraldi and P. Blonda. A survey of fuzzy clustering algorithms for pattern recognition. II. *IEEE Transactions on Systems, Man and Cybernetics, Part B*, 29(6):786–801, 1999.

[BDvLP06] S. Ben-David, U. von Luxburg, and D. Pal. A sober look at clustering stability. *Proceedings of the 19th Annual Conference on Learning Theory (COLT)*, pages 5–19, 2006.

[Ber02] P. Berkhin. Survey of clustering data mining techniques. 2002.

[BHEG02] A. Ben-Hur, A. Elisseeff, and I. Guyon. A stability based method for discovering structure in clustered data. *Pacific Symposium on Biocomputing*, 7:6–17, 2002.

[Bol98] D. Boley. Principal Direction Divisive Partitioning. *Data Mining and Knowledge Discovery*, 2(4):325–344, 1998.

[Bre89] James N. Breckenridge. Replicating cluster analysis: method, consistency, and validity. *Multivariate Behavioral Research*, 24(2):147–161, 1989.

[Bre00] J.N. Breckenridge. Validating cluster analysis: consistent replication and symmetry. *Multivariate Behavioral Research*, 35(2):261–285, 2000.

[Chv83] Vaek Chvtal. *Linear Programming*. W. H. Freeman and Company, New York, 1983.

[DF02] S. Dudoit and J. Fridlyand. A prediction-based resampling method for estimating the number of clusters in a dataset. *Genome Biology*, 3(7):1–21, 2002.

[DHZ+01] Chris H. Q. Ding, Xiaofeng He, Hongyuan Zha, Ming Gu, and Horst D. Simon. A min-max cut algorithm for graph partitioning and data clustering. In *Proceedings of ICDM 2001*, pages 107–114, 2001.

[Dou07] Arnaud Doucet. Computing normalizing constants. April 2007.

[Fan97] J. Fan. Comments on Wavelets in statistics: A review, by A. Antoniadis. *J. Italian Statist. Soc*, 6:131–138, 1997.

[FD01] J. Fridlyand and S. Dudoit. Applications of resampling methods to estimate the number of clusters and to improve the accuracy of a clustering method. Technical report, Division of Biostattistics, University of California, Berkley., 2001.

[FL01] J. Fan and R. Li. Variable Selection Via Nonconcave Penalized Likelihood and Its Oracle Properties. *Journal of the American Statistical Association*, 96(456):1348–1361, 2001.

[FTT04] G.W. Flake, R.E. Tarjan, and K. Tsioutsiouliklis. Graph clustering and minimum cut trees. *Internet Mathematics*, 1(4):385–408, 2004.

[GGG02] J. Goldberger, H. Greenspan, and S. Gordon. Unsupervised image clustering using the information bottleneck method. *Proc. DAGM*, 2002.

[GM98] A. Gelman and X.L. Meng. Simulating normalizing constants: from importance sampling to bridge sampling to path sampling. *Statist. Sci*, 13(2):163–185, 1998.

[GT04] C.D. Giurcaneanu and I. Tabus. Cluster Structure Inference Based on Clustering Stability with Applications to Microarray Data Analysis. *EURASIP Journal on Applied Signal Processing*, 2004(1):64–80, 2004.

[HA85] L. Hubert and P. Arabie. Comparing partitions. *Journal of Classification*, 2(1):193–218, 1985.

[HAK00] Alexander Hinneburg, Charu C. Aggarwal, and Daniel A. Keim. What is the nearest neighbor in high dimensional spaces? In *The VLDB Journal*, pages 506–515, 2000.

[Hen04] C. Hennig. A general robustness and stability theory for cluster analysis. 07 2004.

[HG05] K.A. Heller and Z. Ghahramani. Bayesian hierarchical clustering. In *Proceedings of the 22nd international conference on Machine learning*, pages 297–304. ACM Press New York, NY, USA, 2005.

[HMN05] P. Hall, JS Marron, and A. Neeman. Geometric representation of high dimension, low sample size data. *Journal of the Royal Statistical Society series B*, 67:427–444, 2005.

[HMS02] A. Hotho, A. Maedche, and S. Staab. Text clustering based on good aggregations. *Künstliche Intelligenz (KI)*, 16(4), 2002.

[IT95] Makoto Iwayama and Takenobu Tokunaga. Hierarchical Bayesian clustering for automatic text classification. In Chris E. Mellish, editor, *Proceedings of IJCAI-95, 14th International Joint Conference on Artificial Intelligence*, pages 1322–1327, Montreal, CA, 1995. Morgan Kaufmann Publishers, San Francisco, US.

[JMF99] AK Jain, MN Murty, and PJ Flynn. Data clustering: a review. *ACM Computing Surveys (CSUR)*, 31(3):264–323, 1999.

[JTZ04] D. Jiang, C. Tang, and A. Zhang. Cluster analysis for gene expression data: a survey. *Knowledge and Data Engineering, IEEE Transactions on*, 16(11):1370–1386, 2004.

[Kuh55] HW Kuhn. The Hungarian Method for the Assignment Algorithm. *Naval Research Logistics Quarterly*, 1(1/2):83–97, 1955.

[KVV04] R. KANNAN, S. VEMPALA, and A. VETTA. On Clusterings: Good, Bad and Spectral. *Journal of the ACM*, 51(3):497–515, 2004.

[LBRB02] T. Lange, M. Braun, V. Roth, and J.M. Buhmann. Stability-based model selection. *Advances in Neural Information Processing Systems*, 15, 2002.

[LRBB04] T. Lange, V. Roth, M.L. Braun, and J.M. Buhmann. Stability-based validation of clustering solutions. *Neural Computation*, 16(6):1299–1323, 2004.

[MB02] U. Maulik and S. Bandyopadhyay. Performance evaluation of some clustering algorithms and validity indices. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(12):1650–1654, 2002.

[MC85] GW MILLIGAN and MC COOPER. An examination of procedures for determining the number of clusters in a data set. *Psychometrika*, 50(2):159–179, 1985.

[Mei03] M. Meila. Comparing clusterings. *Proceedings of the Conference on Computational Learning Theory (COLT)*, 2003.

[Mei07] M. Meila. Comparing clusterings: an information based distance. *Journal of Multivariate Analysis*, 98:873–895, 2007.

[MP07] Marina Meila and William Pentney. Clustering by weighted cuts in directed graphs. In *Proceedings of the 2007 SIAM International Conference on Data Mining*, 2007.

[MR06] U. Moller and D. Radke. A Cluster Validity Approach based on Nearest-Neighbor Resampling. *Proceedings of the 18th International Conference on Pattern Recognition (ICPR'06)-Volume 01*, pages 892–895, 2006.

[Nea05] R. M. Neal. Estimating Ratios of Normalizing Constants Using Linked Importance Sampling. *ArXiv Mathematics e-prints*, November 2005.

[NJW02] A. Ng, M. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. *Advances in Neural Information Processing Systems 14: Proceedings of the 2002 [sic] Conference*, 2002.

[PZ] Y. Pei and O. Zaıane. A Synthetic Data Generator for Clustering and Outlier Analysis. Technical report, Technical report, TR06-15, Department of computing Science, University of Alberta, 2006.

[RLBB02] V. Roth, T. Lange, M. Braun, and J. Buhmann. A resampling approach to cluster validation. *Statistics–COMPSTAT*, pages 123–128, 2002.

[RW79] R.H. Randles and D.A. Wolfe. *Introduction to the Theory of Nonparametric Statistics*. Krieger Publishing Company, 1979.

[SB99] K. Stoffel and A. Belkoniene. Parallel k/h-Means Clustering for Large Data Sets. *Proceedings of the 5th International Euro-Par Conference on Parallel Processing*, pages 1451–1454, 1999.

[SG03] M. Smolkin and D. Ghosh. Cluster stability scores for microarray data in cancer studies. *BMC Bioinformatics*, 4:36, 2003.

[SKK00] M. Steinbach, G. Karypis, and V. Kumar. A comparison of document clustering techniques. *KDD Workshop on Text Mining*, 34:35, 2000.

[Slo02] N. Slonim. The Information Bottleneck: Theory and Applications. *Unpublished doctoral dissertation, Hebrew University, Jerusalem, Israel*, 2002.

[ST00a] N. Slonim and N. Tishby. Agglomerative information bottleneck. *Advances in Neural Information Processing Systems*, 12:617–23, 2000.

[ST00b] N. Slonim and N. Tishby. Document clustering using word clusters via the information bottleneck method. *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 208–215, 2000.

[ST07] O. Shamir and N. Tishby. Cluster Stability for Finite Samples. *Advances in Neural Information Processing Systems*, 2007.

[Ste04] D. Steinley. Properties of the hubert-arabie adjusted rand index. *Psychol Methods*, 9(3):386–96, 2004.

[Ste06] D. Steinley. K-means clustering: A half-century synthesis. *British Journal of Mathematical and Statistical Psychology*, 59(1):1–34, 2006.

[TPB00] N. Tishby, F.C. Pereira, and W. Bialek. The information bottleneck method. *Arxiv preprint physics/0004057*, 2000.

[TS00] N. Tishby and N. Slonim. Data clustering by Markovian relaxation and the information bottleneck method. *Advances in Neural Information Processing Systems*, 13:640–646, 2000.

[TW05] R. Tibshirani and G. Walther. Cluster Validation by Prediction Strength. *Journal of Computational & Graphical Statistics*, 14(3):511–528, 2005.

[TWH01] R. Tibshirani, G. Walther, and T. Hastie. Estimating the number of clusters in a data set via the gap statistic. *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, 63(2):411–423, 2001.

[VM01] D. Verma and M. Meila. A comparison of spectral clustering algorithms. *University of Washington Computer Science & Engineering, Technical Report*, pages 1–18, 2001.

[WB01] G. Welch and G. Bishop. An Introduction to the Kalman Filter. *ACM SIGGRAPH 2001 Course Notes*, 2001.

[WVDM00] EA Wan and R. Van Der Merwe. The unscented Kalman filter for nonlinear estimation. *Adaptive Systems for Signal Processing, Communications, and Control Symposium 2000. AS-SPCC. The IEEE 2000*, pages 153–158, 2000.

[XW05] R. Xu and D. Wunsch. Survey of clustering algorithms. *IEEE Transactions on Neural Networks*, 16(3):645–678, 2005.

[YR01] K.Y. Yeung and W.L. Ruzzo. Details of the Adjusted Rand index and Clustering algorithms Supplement to the paper An

empirical study on Principal Component Analysis for clustering gene expression data(to appear in Bioinformatics). *Bioinformatics*, 17(9):763–774, 2001.

[ZK02] Y. Zhao and G. Karypis. Comparison of agglomerative and partitional document clustering algorithms. *SIAM (2002) workshop on Clustering High-dimentional Data and Its Applications*, pages 02–014, 2002.