Clustering
via
Minimum Volume Ellipsoids

Romy Shioda * Levent Tunçel†

CORR 2005-12

May 25, 2005

Abstract

We propose minimum volume ellipsoids (MVE) clustering as an alternate clustering technique to k-means clustering for Gaussian data points and explore its value and practicality. MVE clustering allocates data points into clusters that minimizes the total volumes of each cluster's covering ellipsoids. Motivations for this approach include its scale-invariance, its ability to handle asymmetric and unequal clusters, and our ability to formulate it as a mixed-integer semidefinite programming problem that can be solved to global optimality. We present some preliminary empirical results that illustrate MVE clustering as an appropriate method for clustering data from mixtures of Gaussian distributions and compare its performance with the k-means clustering algorithm.

Keywords: semidefinite optimization, mixed integer programming, clustering, inscribing ellipsoids, interior-point methods, learning theory

AMS Subject Classification: 90C22, 90C11, 62H30, 90C51

*(rshioda@math.uwaterloo.ca) Department of Combinatorics and Optimization, Faculty of Mathematics, University of Waterloo, Waterloo, Ontario N2L 3G1, Canada. Research of this author was supported in part by a Discovery Grant from NSERC and a research grant from Faculty of Mathematics, University of Waterloo.

†(ltuncel@math.uwaterloo.ca) Department of Combinatorics and Optimization, Faculty of Mathematics, University of Waterloo, Waterloo, Ontario N2L 3G1, Canada. Research of this author was supported in part by a Discovery Grant from NSERC and a PREA from Ontario, Canada.

1 Introduction

Clustering or unsupervised learning is a key problem in data mining, machine learning and statistics. It is the problem of allocating data points into K clusters (where K is a predetermined positive integer) such that the points within each cluster are more "closely" related to one another than the points assigned to other clusters. One of the inherent difficulties of this problem is the lack of a clear generally effective criteria function. The ideal criterion or objective function aims to *maximize similarity* of the data points *within each cluster* and *maximize dissimilarity across clusters*. However, the appropriate measure of "similarity" is ambiguous.

The most common measure of similarity in popular clustering methods, such as k-means and hierarchical clustering, is the Euclidean distances between data points and the cluster center. If the data points have d attribute measurements, each of these attributes is considered a dimension in $\mathbb{R}^d$. The k-means method allocates points into clusters so that for each cluster, the total mean Euclidean distances of all the points in the cluster to the center of that cluster (center calculated as the arithmetic mean of all the points in the cluster) is minimized. The main drawback of such a criteria is its tendency to prefer cluster allocation of equal sizes and spherical shapes. Another key disadvantage of this metric is its scale dependence (i.e., the cluster allocation may change significantly with linear transformations of the data space).

To overcome these difficulties we can use Mahalanobis distances instead of the Euclidean distance. For example, if we know that a particular cluster has covariance Σ, then the similarity within that cluster with center $\boldsymbol{c}$ would be measured by $\|\boldsymbol{x} - \boldsymbol{c}\|_{\Sigma^{-1}}$. This measure is scale invariant and can deal with asymmetric non-spherical clusters. (Symons, 1981) shows that among many cluster quality criteria, the within cluster sum of squares (which is proportional to the sample covariance of the clusters) worked best in general. However, the challenge in using Mahalanobis distances is deciding on the covariance matrix of each cluster. Even if each cluster had the same variability and spread, we often do not know the covariance matrix Σ *a priori* and the sample covariance of the points might provide us with an erroneous solution.

A promising alternative scale-invariant metric of cluster quality is given by the minimum volume ellipsoids, where data points are allocated into clusters so that the volumes of the covering ellipsoids for each cluster is minimized. The problem of finding the minimum volume ellipsoid that covers a set of points can be formulated as a semidefinite programming problem and an efficient algorithm for solving the SDP has been proposed by (Sun & Freund, 2004).

Even with an appropriate criteria function, there is another inherent challenge in clustering – the problem of allocating the data points to optimize the objective function. Optimal allocation has long been considered computationally intractable, thus data points are allocated heuristically. One approach is to use randomized local search methods. For example, the k-means algorithm initially randomly allocates points to clusters then iteratively re-allocates the points to their closest cluster. Other heuristic methods are based on greedy approaches such as in hierarchical clustering. This method initially assigns each point as a singleton cluster and merges the closest clusters together until there are K clusters remaining. These methods work well with Euclidean distances, but face several problems using Mahalanobis distances and minimum volume ellipsoids as the clustering criteria (Kumar *et al.* , 2003). When these alternative measures are used in the k-means heuristic, it often gets stuck in local minima and produce poor cluster allocations that are far from being globally optimal.

Our goal in this paper is to explore the possibility and the value of optimal cluster allocation using minimum volume ellipsoids (MVE). There is no practical approach for solving the k-means problem to optimality, however, we can formulate the MVE clustering problem as a mixed-integer semidefinite programming problem, allowing us to solve it to optimality. Figure 1 shows the result of running the k-means algorithm with 1000 different starting points and the optimal MVE algorithm on two bivariate Gaussian distributions.

The problem is an example of a union between semidefinite programming and combinatorial optimization. Computing the minimum volume ellipsoid is a semidefinite programming problem whereas cluster allocation is a form of a facility location problem. The problem of determining the minimum volume covering ellipsoid has its roots in classical, ellipsoidal problems in convex geometry which go back at least to (Löwner, 1934) and (Rosen, 1965). (John, 1948) developed significant amount of theory and (Barnes, 1982) proposed a heuristic algorithm based on the eigenvalue decomposition. The current best complexity bound is due to (Khachiyan, 1996). Other interesting theoretical algorithms are presented in (Khachiyan & Todd, 1993) and (Nesterov & Nemirovskii, 1994). Recently, many advances have been made to provide more practical algorithms with good foundations in convex optimization (Zhang & Gao, 2004; Toh, 1999; Sun & Freund, 2004; Kumar & Yıldırım, 2003). We extend the algorithm developed by (Sun & Freund, 2004) to solve the continuous relaxation of this clustering problem and solve it to optimality via various branch-and-bound algorithms.

Dealing with practical problems which lie in the span of this paper would require us to address the problem of *outliers*. In our case, interesting techniques for "outlier detection" compatible with our approach already exist (see (Rousseeuw & Leroy, 1987) as well as (Dunagan & Vempala, 2001)).

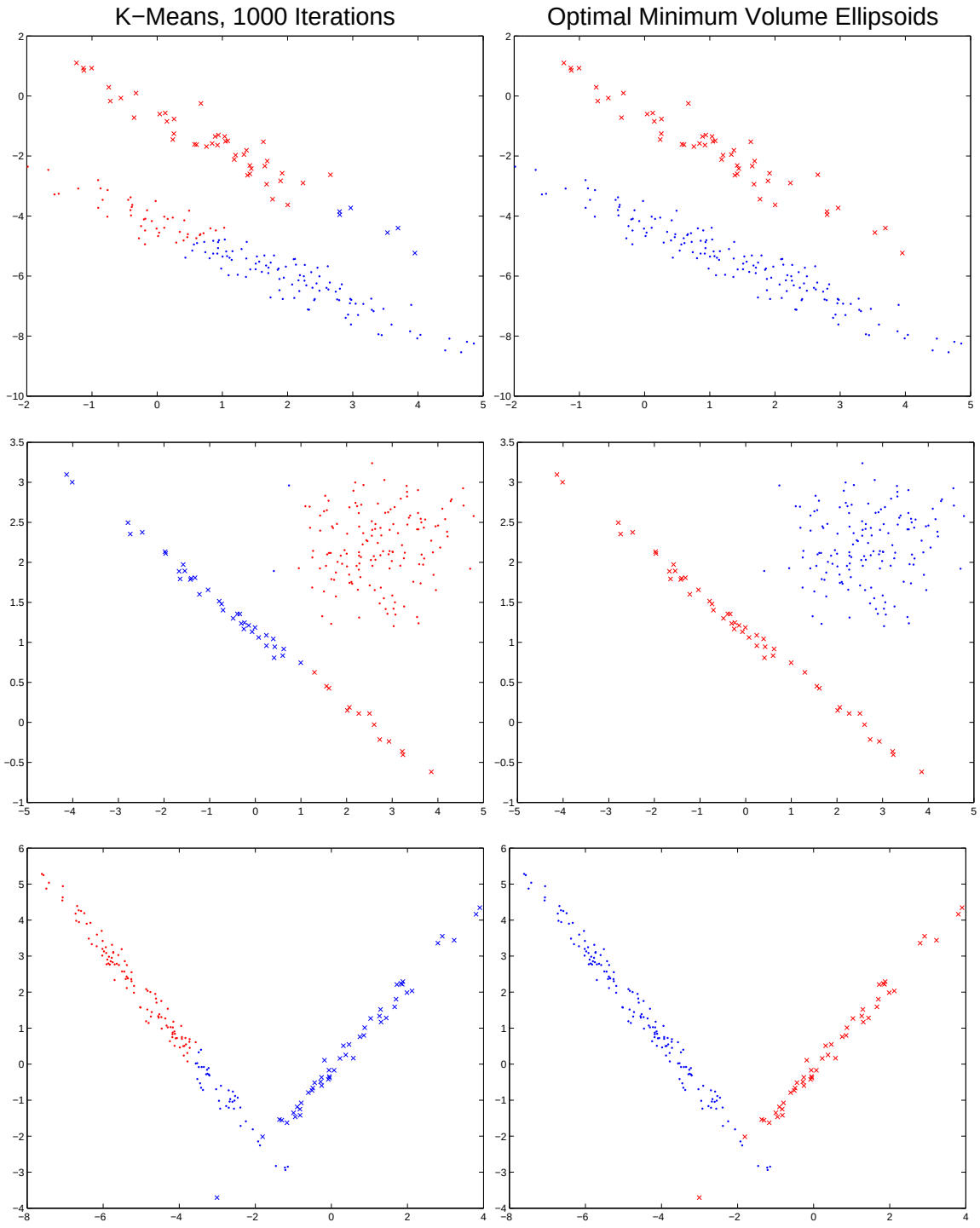

Figure 1: Three examples of two bivariate Gaussian distributions. The original distribution is represented with an "o" (150 points) and "x" (50 points). The clustering algorithms (k-means with 1000 iterations on the left and optimal minimum volume ellipsoids on the right) clustered the points into red and blue clusters.

Therefore, we keep our focus on the underlying optimization problem and consider outlier removal as a separate problem which is outside the scope of the current paper.

We view our contribution as follows:

1. Propose and evaluate formulations and algorithms combining semidefinite programming and mixed-integer programming for solving a clustering problem to optimality.

2. Empirically illustrate the minimum volume covering ellipsoids as an appropriate measure for clustering data from mixtures of Gaussian distributions and compare its performance with the k-means clustering algorithm.

The structure of the paper is as follows: Section 2 describes the problem at hand and proposes two solution methods. Section 3 describes the Pure Branch-and-Bound approach and Section 4 elaborates on the Convex Relaxation Branch-and-Bound approach. Section 5 describes several implementation strategies for both of the branch-and-bound algorithms. Section 6 presents a refinement to the original problem, where cluster membership information for a small subset of the data points are given to us *a priori*. Section 7 describes several alternative nonconvex formulations of the problem. Section 8 illustrates the computational experimentations and results of applying MVE clustering compared to k-means clustering on Gaussian data points. Section 9 shows analogous computational results for the prior information case described in Section 6. Finally, we present our conclusions and future potential work in Section 10.

1.1 Notation

We denote by $\mathbb{S}^d$, the space of symmetric $(d \times d)$ matrices over the reals. $\mathbb{S}^d_+$ stands for the convex cone of positive semidefinite matrices in $\mathbb{S}^d$. Finally, $\mathbb{S}^d_{++}$ is the interior of $\mathbb{S}^d_+$, i.e., $\mathbb{S}^d_{++}$ is the convex cone of $(d \times d)$ symmetric positive definite matrices over the reals. We also use the partial order notation where for $A, B \in \mathbb{S}^d$, we write $A \succeq B$ $(A \succ B)$ to mean $(A - B)$ is positive semidefinite (positive definite).

2 The Problem

Our minimum volume ellipsoid clustering method is as follows. Given n points in $\mathbb{R}^d$: $\boldsymbol{a}_1, \boldsymbol{a}_2, ..., \boldsymbol{a}_n$, find an allocation of these points into K clusters that minimizes the total volume of the minimum volume covering ellipsoid for each cluster. To state this formally, let C_j denote the set of indices of the data

points assigned to Cluster j, $j = 1, 2, ..., K$, and E_j, $E_j \subset \mathbb{R}^d$, is the ellipsoid that contains all the points $\boldsymbol{a}_i \in C_j$.

The parameters determining an ellipsoid E_j are its center $\boldsymbol{c}_j$ and $\boldsymbol{Q}_j \in \mathbb{S}^d_{++}$ that defines its shape and size, i.e.,

$$E_j = \{\boldsymbol{x} | (\boldsymbol{x} - \boldsymbol{c}_j)^T \boldsymbol{Q}_j (\boldsymbol{x} - \boldsymbol{c}_j) \leq 1\}.$$

However, as in (Sun & Freund, 2004) we define it by

$$E_j = \{\boldsymbol{x} | (\boldsymbol{M}_j \boldsymbol{x} - \boldsymbol{z}_j)^T (\boldsymbol{M}_j \boldsymbol{x} - \boldsymbol{z}_j) \leq 1\},$$

where $\boldsymbol{M}_j = \boldsymbol{Q}_j^{\frac{1}{2}}$ and $\boldsymbol{z}_j = \boldsymbol{Q}_j^{\frac{1}{2}} \boldsymbol{c}_j$.

Since the volume of E_j is proportional to $\det(\boldsymbol{M}_j)^{-1}$, we wish to find the optimal allocation C_j, $j = 1, ..., K$, that solves:

$$\begin{aligned} \min \quad & -\sum_{j=1}^{K} \ln \det(\boldsymbol{M}_j), \\ \text{s.t.} \quad & (\boldsymbol{M}_j \boldsymbol{a}_i - \boldsymbol{z}_j)^T (\boldsymbol{M}_j \boldsymbol{a}_i - \boldsymbol{z}_j) \leq 1, \quad i \in C_j, j = 1, ..., K, \\ & \bigcup_j C_j = \{1, ..., n\}, \\ & \boldsymbol{M}_j \in \mathbb{S}^d_{++}. \end{aligned} \tag{1}$$

Note that we simply chose to add the terms $-\ln \det(\boldsymbol{M}_j)$ for all j in the objective function. In some applications it might make sense to take a weighted combination of these terms with given positive weights w_j so that the objective function becomes

$$\min -\sum_{j=1}^{K} w_j \ln \det(\boldsymbol{M}_j).$$

Indeed, from a mathematical point of view, everything we do in this paper can trivially be adapted to this more general situation.

We test two different approaches in solving this semidefinite mixed integer programming problem, namely (1) Pure Branch-and-Bound and (2) Convex Relaxation Branch-and-Bound.

3 Pure Branch-and-Bound

Pure Branch-and-Bound method is a simple branching procedure that adds a point to a cluster until all of the points are assigned. The sketch of the procedure is as follows:

Step 0: *Initialization.* Set $C_j := \emptyset$, $j = 1, ..., K$. $C_1 := \{1\}$, $UB := \infty$.

Step 1: *Calculate MVE.* For each Cluster j, $j = 1, ..., K$, calculate the minimum volume ellipsoid E_j where $\boldsymbol{a}_i \in E_j$, $\forall i \in C_j$, parameterized by $\boldsymbol{M}_j$ and $\boldsymbol{z}_j$. If $-\sum_{j=1}^{K} \ln \det(\boldsymbol{M}_j) > UB$, Stop branching.

Step 2: *Find Interior Point.* For each unassigned point, $\boldsymbol{a}_i$, $i \in \{1, ..., n\} \setminus \bigcup_{j=1,...,K} C_j$, assign the point to Cluster j if $\boldsymbol{a}_i \in E_j$. If all n points are assigned to a cluster, Stop Branching. If $-\sum_{j=1}^{K} \ln \det(\boldsymbol{M}_j) < UB$, set $UB = -\sum_{j=1}^{K} \ln \det(\boldsymbol{M}_j)$.

Step 3: *Branch on Unassigned Point.* Pick an unassigned point $\boldsymbol{a}_i$, $i \in \{1, ..., n\} \setminus \bigcup_{j=1,...,K} C_j$. For each Cluster j, $j = 1, ..., K$, set $C_j = C_j \cup \{i\}$ and go to Step 1.

At first, it seems to be a trivial enumeration procedure, but we implement the branching so that many unassigned points are assigned in Step 2. Thus, we only need to branch on a fraction of the total number of points in practice. We will elaborate on the specific implementation of the branching procedure in Section 5.

To calculate the minimum volume covering ellipsoid in Step 1, we solve the following problem:

$$\begin{array}{rrcl} \text{Min} & -\sum_{j=1}^{K} \ln\left(\det\left(\boldsymbol{M}_j\right)\right) & & \\ & \left(\boldsymbol{M}_j\boldsymbol{a}_i - \boldsymbol{z}_j\right)^T \left(\boldsymbol{M}_j\boldsymbol{a}_i - \boldsymbol{z}_j\right) & \leq & 1, \quad i \in C_j, \forall j \in \{1, 2, \ldots, K\}; \\ & \boldsymbol{M}_j \in \mathbb{S}^d, \boldsymbol{z}_j \in \mathbb{R}^d, & & \forall j \in \{1, 2, \ldots, K\}, \end{array}$$

where the positive definiteness of $\boldsymbol{M}_j$ is enforced by the domain of the objective function. We solve the above problem using the Dual Reduced Newton algorithm proposed by (Sun & Freund, 2004).

4 Convex Relaxation Branch-and-Bound Method

In the Convex Relaxation Branch-and-Bound method, we wish to solve Problem (1) as a mixed-integer semidefinite programming problem.

Let

$$\beta_{ij} := \begin{cases} 1, & \text{if point } i \text{ is assigned to Cluster } j; \\ 0, & \text{otherwise.} \end{cases}$$

Let $R_j \in \mathbb{Z}_{++}$ be a large enough constant such that for every i and for every j, we have

$$\left(\boldsymbol{M}_j^*\boldsymbol{a}_i - \boldsymbol{z}_j^*\right)^T \left(\boldsymbol{M}_j^*\boldsymbol{a}_i - \boldsymbol{z}_j^*\right) \leq R_j,$$

where $\boldsymbol{M}_j^*$ and $\boldsymbol{z}_j^*$ denote the parameters (initially unknown) of the minimum volume ellipsoid for Cluster j. With this notation, our naïve formulation is:

$$\text{Min} \quad -\sum_{j=1}^{K} \ln\left(\det\left(\boldsymbol{M}_j\right)\right) \tag{2}$$

$$\text{s.t.} \quad \left(\boldsymbol{M}_j\boldsymbol{a}_i - \boldsymbol{z}_j\right)^T\left(\boldsymbol{M}_j\boldsymbol{a}_i - \boldsymbol{z}_j\right) \leq \beta_{ij} + R_j\left(1-\beta_{ij}\right), \forall i \in \{1,2,\ldots,n\}, \forall j \in \{1,2,\ldots,K\}; \tag{3}$$

$$\sum_{j=1}^{K} \beta_{ij} = 1, \forall i \in \{1,2,\ldots,n\}; \tag{4}$$

$$\beta_{ij} \geq 0, \quad \text{and integer}, \forall i \in \{1,2,\ldots,n\}, \forall j \in \{1,2,\ldots,K\}; \tag{5}$$

$$\boldsymbol{M}_j \in \mathbb{S}^d, \boldsymbol{z}_j \in \mathbb{R}^d, \quad \forall j \in \{1,2,\ldots,K\}. \tag{6}$$

Note that the constraints $\boldsymbol{M}_j \succ 0$ for every $j \in \{1,2,\ldots,K\}$ are implied by the domain of the objective function.

A very important drawback of this formulation is R_j. While it is reasonable to assume that the optimal solution will have the property that $\boldsymbol{M}_j^* \succ 0$ for every j and that all $\boldsymbol{z}_j^*$ lie in a box (which can be computed from $\boldsymbol{a}_i$), it is much less reasonable to assume certain orientations of the ellipsoids or small ranges for the eigenvalues of all $\boldsymbol{M}_j^*$ a priori. Without proper handling of this parameter R_j, we do not expect that the above formulation will be effective, especially for large n. Indeed, if the best value for R_j (the smallest value which makes the above formulation valid) is too large, this might indicate some kind of intrinsic difficulty with the data. Appendix A illustrates how we can find a theoretical upperbound for the values of R_j.

We propose to solve the above semidefinite mixed-integer optimization problem via branch-and-bound where the convex relaxation would be solved at each node. Each branching step would assign an unassigned data point to one of the K clusters. At an intermediary node, if C_j are the indices of the points assigned to Cluster j, the subproblem that is solved at the node is the following continuous

convex semidefinite programming problem:

$$\min \quad -\sum_{j=1}^{K} \ln\det(\boldsymbol{M}_j) \tag{7}$$

$$\text{s.t.:} \quad (\boldsymbol{M}_j\boldsymbol{a}_i - \boldsymbol{z}_j)^T(\boldsymbol{M}_j\boldsymbol{a}_i - \boldsymbol{z}_j) + (R_j - 1)\beta_{ij} \le R_j, \quad \forall i \notin \bigcup_{j=1}^{K} C_j, \forall j; \tag{8}$$

$$(\boldsymbol{M}_j\boldsymbol{a}_i - \boldsymbol{z}_j)^T(\boldsymbol{M}_j\boldsymbol{a}_i - \boldsymbol{z}_j) \le 1, \quad \forall i \in C_j, \forall j; \tag{9}$$

$$\sum_{j=1}^{K} \beta_{ij} = 1, \quad \forall i \notin \bigcup_{j=1}^{K} C_j; \tag{10}$$

$$\beta_{ij} \ge 0, \quad \forall i, \forall j; \tag{11}$$

$$\boldsymbol{M}_j \in \mathbb{S}^d, \boldsymbol{z}_j \in \mathbb{R}^d, \quad \forall j; \tag{12}$$

where the constraints (9) are the ellipsoid inclusion constraint for the assigned points and constraints (8) are the ellipsoid inclusion constraints for the unassigned points. Note that the relaxed integer assignment variables β_{ij} exist only for the unassigned data points $\boldsymbol{a}_i$, i.e, $i \notin \bigcup_{j=1}^{K} C_j$. This formulation is the same as setting $\beta_{ij} = 1$ for all $i \in C_j$ in the relaxation of the original problem.

The subsequent sections discuss how we solve the above problem by solving the necessary and sufficient optimality conditions of the problem via Newton's method. Subsection 4.1 derives the optimality conditions for the subproblem and Subsection 4.2 illustrates the Newton directions for the optimality conditions.

4.1 Optimality Conditions for the Relaxations

Let's relax the constraint "β_{ij} is integer for every $i \notin \bigcup_{j=1}^{K} C_j$ and $j \in \{1, 2, \ldots, K\}$." To apply an interior-point-method, we consider the following formulation (parameterized by $\mu > 0$):

$$\min \quad -\sum_{j=1}^{K} \ln\det(\boldsymbol{M}_j) - \mu \sum_{i=1}^{n}\sum_{j=1}^{K} [\ln(t_{ij}) + \ln(\beta_{ij})]$$

$$\text{subject to:} \quad (\boldsymbol{M}_j\boldsymbol{a}_i - \boldsymbol{z}_j)^T(\boldsymbol{M}_j\boldsymbol{a}_i - \boldsymbol{z}_j) + (R_j - 1)\beta_{ij} + t_{ij} = R_j, \quad \forall i \notin \bigcup_{j=1}^{K} C_j, \forall j;$$

$$(\boldsymbol{M}_j\boldsymbol{a}_i - \boldsymbol{z}_j)^T(\boldsymbol{M}_j\boldsymbol{a}_i - \boldsymbol{z}_j) + t_{ij} = 1, \quad \forall i \in C_j, \forall j;$$

$$\sum_{j=1}^{K} \beta_{ij} = 1, \quad \forall i \notin \bigcup_{j=1}^{K} C_j.$$

We denote by t_{ij} the slack variables in the ellipsoidal inclusion constraints. We use u_{ij} for the dual variables corresponding to the same constraints. The dual variables v_i are for the linear equations on

β_{ij} ensuring that every point is included in some cluster. Now, we can list the (necessary and sufficient) optimality conditions for the above problem:

$$\begin{aligned}
\sum_{i=1}^{n} u_{ij}\left[(\boldsymbol{M}_j\boldsymbol{a}_i - \boldsymbol{z}_j)\boldsymbol{a}_i^T + \boldsymbol{a}_i(\boldsymbol{M}_j\boldsymbol{a}_i - \boldsymbol{z}_j)^T\right] - \boldsymbol{M}_j^{-1} &= \boldsymbol{0}, \forall j \in \{1,2,\ldots,K\};\\
\sum_{i=1}^{n} u_{ij}(\boldsymbol{z}_j - \boldsymbol{M}_j\boldsymbol{a}_i) &= \boldsymbol{0}, \forall j \in \{1,2,\ldots,K\};\\
(R_j - 1)u_{ij} + v_i - \frac{\mu}{\beta_{ij}} &= 0, \forall i \notin \bigcup_{j=1}^{K} C_j, j \in \{1,2,\ldots,K\};\\
u_{ij} - \frac{\mu}{t_{ij}} &= 0, \forall i \in \{1,2,\ldots,n\}, j \in \{1,2,\ldots,K\};\\
(\boldsymbol{M}_j\boldsymbol{a}_i - \boldsymbol{z}_j)^T(\boldsymbol{M}_j\boldsymbol{a}_i - \boldsymbol{z}_j) + t_{ij} + (R_j - 1)\beta_{ij} &= R_j, \forall i \notin \bigcup_{j=1}^{K}, j \in \{1,2,\ldots,K\};\\
\sum_{j=1}^{K} \beta_{ij} &= 1, \forall i \notin \bigcup_{j=1}^{K} C_j;\\
\boldsymbol{M}_j \succ \boldsymbol{O}, t_{ij} > 0, u_{ij} > 0, \beta_{ij} > 0, & \quad \forall i, \forall j.
\end{aligned}$$

Suppose n_0 is the number of unassigned points, i.e., $n_0 = n - \sum_{k=1}^{K} |C_k|$, and n_j is the number of points assigned to Cluster j, i.e., $n_j = |C_j|$. We denote by $\boldsymbol{e}$ the vector of all ones, $\boldsymbol{A}_j \in \mathbb{R}^{d \times n_0+n_j}$ denotes the matrix of $\boldsymbol{a}_i$'s that can be assigned to Cluster j, i.e., the first n_0 columns of $\boldsymbol{A}_j$ are comprised of $\boldsymbol{a}_i$ for $i \notin \bigcup_{k=1}^{K} C_k$ and the last n_k columns of $\boldsymbol{A}_j$ are comprised of $\boldsymbol{a}_i$ for $i \in C_j$. Similar to (Sun & Freund, 2004), we can eliminate the variables $\boldsymbol{M}_j$ and $\boldsymbol{z}_j$ in the above formulation. Let $\boldsymbol{u}_j \in \mathbb{R}^{n_0+n_j}$ denote the vector of variables u_{ij}, the first n_0 elements corresponding to $i \notin \bigcup_{k=1}^{K} C_k$ and the last n_j elements corresponding to $i \in C_j$. Then, $\boldsymbol{M}_j$, $\boldsymbol{z}_j$ and $\boldsymbol{u}_j$ satisfying the above optimality conditions (for some $\boldsymbol{\beta}$, $\boldsymbol{v}$) have the following relationships:

$$\boldsymbol{M}_j = \frac{1}{\sqrt{2}}\left[\left(\boldsymbol{A}_j \mathrm{Diag}(\boldsymbol{u}_j)\boldsymbol{A}_j^T\right) - \frac{\boldsymbol{A}_j\boldsymbol{u}_j\boldsymbol{u}_j^T\boldsymbol{A}_j^T}{\boldsymbol{e}^T\boldsymbol{u}_j}\right]^{-1/2},$$

$$\boldsymbol{z}_j = \frac{\boldsymbol{M}_j\boldsymbol{A}_j\boldsymbol{u}_j}{\boldsymbol{e}^T\boldsymbol{u}_j},$$

where $\mathrm{Diag}(\boldsymbol{x})$ is a diagonal matrix with $\boldsymbol{x}$ on the diagonal. Let $h_{ij}(\boldsymbol{u}_j)$ ($h_{ij} : \mathbb{R}^n \to \mathbb{R}$) be a nonlinear function of u_{ij} that corresponds to the value of $(\boldsymbol{M}_j\boldsymbol{a}_i - \boldsymbol{z}_j)^T(\boldsymbol{M}_j\boldsymbol{a}_i - \boldsymbol{z}_j)$ with the above relations substituted, i.e.,

$$h_{ij}(\boldsymbol{u}_j) = \frac{1}{2}\left(\boldsymbol{a}_i - \frac{\boldsymbol{A}_j\boldsymbol{u}_j}{\boldsymbol{e}^T\boldsymbol{u}_j}\right)^T\left[\boldsymbol{A}_j\mathrm{Diag}(\boldsymbol{u}_j)\boldsymbol{A}_j^T - \frac{\boldsymbol{A}_j\boldsymbol{u}_j\boldsymbol{u}_j^T\boldsymbol{A}_j^T}{\boldsymbol{e}^T\boldsymbol{u}_j}\right]^{-1}\left(\boldsymbol{a}_i - \frac{\boldsymbol{A}_j\boldsymbol{u}_j}{\boldsymbol{e}^T\boldsymbol{u}_j}\right).$$

Then the necessary and sufficient optimality conditions reduce to the following:

$$h_{ij}(\boldsymbol{u}_j) + (R_j - 1)\beta_{ij} + t_{ij} = R_j, \quad \forall i \notin \bigcup_{j=1}^{K} C_j, j \in \{1, 2, \ldots, K\}; \tag{13}$$

$$h_{ij}(\boldsymbol{u}_j) + t_{ij} = 1, \quad \forall i \in C_j, j \in \{1, 2, \ldots, K\}; \tag{14}$$

$$u_{ij}t_{ij} = \mu, \quad \forall i \in \{1, 2, \ldots, n\}, j \in \{1, 2, \ldots, K\}; \tag{15}$$

$$(R_j - 1)u_{ij}\beta_{ij} + v_i\beta_{ij} = \mu, \quad \forall i \notin \bigcup_{j=1}^{K} C_j, j \in \{1, 2, \ldots, K\}; \tag{16}$$

$$\sum_{j=1}^{K} \beta_{ij} = 1, \quad \forall i \notin \bigcup_{j=1}^{K} C_j; \tag{17}$$

$$u_{ij} > 0, \beta_{ij} > 0, t_{ij} > 0, \quad \forall i, \forall j. \tag{18}$$

4.2 Newton-type Direction for the Convex Relaxation

Given μ and u_{ij}, t_{ij}, v_i and β_{ij}, we compute a Newton-type direction for the optimality conditions (13)-(18). We first introduce some additional notation. To distinguish between assigned and unassigned points, let $\tilde{\boldsymbol{u}}_j$ be the vector of u_{ij} that are not assigned, $\hat{\boldsymbol{u}}_j$ be the vector of u_{ij} that have been assigned to Cluster j, $\tilde{\boldsymbol{t}}_j$ be the vector of t_{ij} that are not assigned, and $\hat{\boldsymbol{t}}_j$ be the vector of t_{ij} that have been assigned to Cluster j. Further, let $\boldsymbol{\beta}_j$ be the vector of β_{ij} and $\boldsymbol{v}$ be the vector of v_i. Let $\boldsymbol{H}_j$ denote the matrix of first derivatives of $h_{ij}(\boldsymbol{u}_j)$ with respect to u_{ij}. This matrix can be partitioned into four submatrices:

$$\boldsymbol{H}_j := \nabla h_j(\boldsymbol{u}_j) =: \begin{bmatrix} \boldsymbol{H}_{(1)\tilde{\boldsymbol{u}}_j} & \boldsymbol{H}_{(1)\hat{\boldsymbol{u}}_j} \\ \boldsymbol{H}_{(2)\tilde{\boldsymbol{u}}_j} & \boldsymbol{H}_{(2)\hat{\boldsymbol{u}}_j} \end{bmatrix}.$$

$\boldsymbol{H}_{(1)\tilde{\boldsymbol{u}}_j}$ is the submatrix of $\boldsymbol{H}$ corresponding to (13) and $\tilde{\boldsymbol{u}}_j$, $\boldsymbol{H}_{(1)\hat{\boldsymbol{u}}_j}$ corresponds to (13) and $\hat{\boldsymbol{u}}_j$, $\boldsymbol{H}_{(2)\tilde{\boldsymbol{u}}_j}$ corresponds to (14) and $\tilde{\boldsymbol{u}}_j$, and $\boldsymbol{H}_{(2)\hat{\boldsymbol{u}}_j}$ corresponds to (14) and $\hat{\boldsymbol{u}}_j$.

The Newton-type search direction $(\Delta\tilde{\boldsymbol{u}}_j, \Delta\hat{\boldsymbol{u}}_j, \Delta\tilde{\boldsymbol{t}}_j, \Delta\hat{\boldsymbol{t}}_j, \Delta\boldsymbol{\beta}_j, \Delta\boldsymbol{v})$, $j \in \{1, 2, \ldots, K\}$, at point $(\tilde{\boldsymbol{u}}_j, \hat{\boldsymbol{u}}_j, \tilde{\boldsymbol{t}}_j, \hat{\boldsymbol{t}}_j, \boldsymbol{\beta}_j, \boldsymbol{v})$, $j \in \{1, 2, \ldots, K\}$, is the solution to the following linear system:

$$H_{(1)\tilde{\boldsymbol{u}}_j}\Delta\tilde{\boldsymbol{u}}_j + H_{(1)\hat{\boldsymbol{u}}_j}\Delta\hat{\boldsymbol{u}}_j + (R_j - 1)\Delta\boldsymbol{\beta}_j + \Delta\tilde{\boldsymbol{t}}_j = \tilde{\boldsymbol{r}}_{1,j}, \forall j \in \{1, \ldots, K\}, \tag{19}$$

$$H_{(2)\tilde{\boldsymbol{u}}_j}\Delta\tilde{\boldsymbol{u}}_j + H_{(2)\hat{\boldsymbol{u}}_j}\Delta\hat{\boldsymbol{u}}_j + \Delta\hat{\boldsymbol{t}}_j = \boldsymbol{r}_{1,j}, \forall j \in \{1, \ldots, K\}, \tag{20}$$

$$\tilde{\boldsymbol{U}}_j\Delta\tilde{\boldsymbol{t}}_j + \tilde{\boldsymbol{T}}_j\Delta\tilde{\boldsymbol{u}}_j = \tilde{\boldsymbol{r}}_{2,j}, \forall j \in \{1, \ldots, K\}, \tag{21}$$

$$\hat{\boldsymbol{U}}_j\Delta\hat{\boldsymbol{t}}_j + \hat{\boldsymbol{T}}_j\Delta\hat{\boldsymbol{u}}_j = \boldsymbol{r}_{2,j}, \forall j \in \{1, \ldots, K\}, \tag{22}$$

$$(R_j - 1)\boldsymbol{B}_j\Delta\tilde{\boldsymbol{u}}_j + ((R_j - 1)\tilde{\boldsymbol{U}}_j + \boldsymbol{V})\Delta\boldsymbol{\beta}_j + \boldsymbol{B}_j\Delta\boldsymbol{v} = \tilde{\boldsymbol{r}}_{3,j}, \forall j \in \{1, \ldots, K\}, \tag{23}$$

$$\sum_{j=1}^{K} \Delta\boldsymbol{\beta}_j = \tilde{\boldsymbol{r}}_4, \tag{24}$$

where

$$\begin{aligned}
\tilde{\boldsymbol{U}}_j &:= \mathrm{Diag}(\tilde{\boldsymbol{u}}_j), \quad \forall j \in \{1,\ldots,K\},\\
\hat{\boldsymbol{U}}_j &:= \mathrm{Diag}(\hat{\boldsymbol{u}}_j), \quad \forall j \in \{1,\ldots,K\},\\
\tilde{\boldsymbol{T}}_j &:= \mathrm{Diag}(\tilde{\boldsymbol{t}}_j), \quad \forall j \in \{1,\ldots,K\},\\
\hat{\boldsymbol{T}}_j &:= \mathrm{Diag}(\hat{\boldsymbol{t}}_j), \quad \forall j \in \{1,\ldots,K\},\\
\boldsymbol{B_j} &:= \mathrm{Diag}(\boldsymbol{\beta}_j), \quad \forall j \in \{1,\ldots,K\},\\
\boldsymbol{V} &:= \mathrm{Diag}(\boldsymbol{v}),\\
\tilde{\boldsymbol{r}}_{1,j} &:= R_j\boldsymbol{e} - h_j(\tilde{\boldsymbol{u}}_j) - (R_j-1)\boldsymbol{\beta}_j - \tilde{\boldsymbol{t}}_j, \quad \forall j \in \{1,\ldots,K\},\\
\boldsymbol{r}_{1,j} &:= \boldsymbol{e} - h_j(\hat{\boldsymbol{u}}_j) - \hat{\boldsymbol{t}}_j, \quad \forall j \in \{1,\ldots,K\},\\
\tilde{\boldsymbol{r}}_{2,j} &:= \mu\boldsymbol{e} - \tilde{\boldsymbol{U}}_j\tilde{\boldsymbol{t}}_j, \quad \forall j \in \{1,\ldots,K\},\\
\boldsymbol{r}_{2,j} &:= \mu\boldsymbol{e} - \hat{\boldsymbol{U}}_j\hat{\boldsymbol{t}}_j, \quad \forall j \in \{1,\ldots,K\},\\
\tilde{\boldsymbol{r}}_{3,j} &:= \mu\boldsymbol{e} - (R_j-1)\tilde{\boldsymbol{U}}_j\boldsymbol{\beta}_j - \boldsymbol{V}\boldsymbol{\beta}_j, \quad \forall j \in \{1,\ldots,K\},\\
\tilde{\boldsymbol{r}}_4 &:= \boldsymbol{e} - \sum_{j=1}^{K}\boldsymbol{\beta}_j.
\end{aligned}$$

After some reorganizing, we can write the above system solely in terms of $\Delta\tilde{\boldsymbol{u}}_j$ and $\Delta\hat{\boldsymbol{u}}_j$: for every $j \in \{1,2,\ldots,K\}$,

$$\left(\boldsymbol{H}_{(1)\tilde{\boldsymbol{u}}_j} - \boldsymbol{D}_{\tilde{t}_j} - \boldsymbol{D}_{\beta_j}\right)\Delta\tilde{\boldsymbol{u}}_j + \boldsymbol{H}_{(1)\hat{\boldsymbol{u}}_j}\Delta\hat{\boldsymbol{u}}_j + \sum_{l\neq j}\boldsymbol{D}_{j,l}\Delta\tilde{\boldsymbol{u}}_l = \boldsymbol{c}_j, \tag{25}$$

$$\boldsymbol{H}_{(2)\tilde{\boldsymbol{u}}_j}\Delta\tilde{\boldsymbol{u}}_j + \left(\boldsymbol{H}_{(2)\hat{\boldsymbol{u}}_j} - \boldsymbol{D}_{\hat{t}_j}\right)\Delta\hat{\boldsymbol{u}}_j = \boldsymbol{r}_{1,j} - \hat{\boldsymbol{U}}_j^{-1}\boldsymbol{r}_{2,j}, \tag{26}$$

where $\boldsymbol{D}_{\tilde{t}_j}$, $\boldsymbol{D}_{\beta_j}$, and $\boldsymbol{D}_{\hat{t}_j}$ are diagonal matrices resulting from back substituting $\tilde{\boldsymbol{t}}_{\boldsymbol{j}}$, $\boldsymbol{\beta_j}$ and $\hat{\boldsymbol{t}}_{\boldsymbol{j}}$, respectively, and $\boldsymbol{D}_{j,l}$ is a diagonal matrix resulting from (24). Let $\boldsymbol{S}_j := ((R_j-1)\tilde{\boldsymbol{U}}_j + \boldsymbol{V})^\dagger\boldsymbol{B}_j$. (We use $\boldsymbol{A}^\dagger$ to denote the *pseudo-inverse of* $\boldsymbol{A}$.) Then,

$$\boldsymbol{D}_{\tilde{t}_j} := \tilde{\boldsymbol{U}}_j^{-1}\tilde{\boldsymbol{T}}, \tag{27}$$

$$\boldsymbol{D}_{\hat{t}_j} := \hat{\boldsymbol{U}}_j^{-1}\hat{\boldsymbol{T}}, \tag{28}$$

$$\boldsymbol{D}_{\beta_j} := (R_j-1)^2\left(\boldsymbol{B}_j - \boldsymbol{S}_j(\sum_j \boldsymbol{S}_j)^\dagger\right), \tag{29}$$

$$\boldsymbol{D}_{j,l} := (R_j-1)^2\boldsymbol{S}_j\boldsymbol{S}_l(\sum_j \boldsymbol{S}_j)^\dagger, \tag{30}$$

$$\boldsymbol{c}_j := \tilde{\boldsymbol{r}}_{\boldsymbol{1,j}} - \tilde{\boldsymbol{U}}_j^{-1}\tilde{\boldsymbol{r}}_2 - (R_j-1)\boldsymbol{S}_j(\sum_\ell \boldsymbol{S}_\ell)^\dagger\left(\boldsymbol{B}_j^{-1}(\sum_\ell \boldsymbol{S}_\ell)\tilde{\boldsymbol{r}}_3 - \sum_\ell \boldsymbol{S}_\ell\boldsymbol{B}^{-1}\tilde{\boldsymbol{r}}_{3,\ell} - \tilde{\boldsymbol{r}}_4\right). \tag{31}$$

Linear Algebra for each Iteration

Since we do not have any explicit control over the values of $\boldsymbol{V}$, it seems conceivable that $\left[(R_j-1)\tilde{\boldsymbol{U}}_j+\boldsymbol{V}\right]$ is singular for some j for some iterate. Moreover, the linear system (19)-(24) may be singular. It is known that H_j is negative semidefinite (Proposition 5 of (Sun & Freund, 2004)) which implies

$$H_j - \begin{pmatrix} \tilde{\boldsymbol{U}}_j^{-1}\tilde{\boldsymbol{T}}_j & 0 \\ 0 & \hat{\boldsymbol{U}}_j^{-1}\hat{\boldsymbol{T}}_j \end{pmatrix} \prec 0.$$

We would like to have a linear system whose nonsingularity is based on the above fact.

Note that we cannot ensure the nonsingularity of the current system based on this fact alone: Suppose that the above matrix is $-(R_j-1)I$ for some iterate for every $j \in \{1,2,\ldots,K\}$. We may also choose positive values for $\boldsymbol{u}$, $\boldsymbol{\beta}$ and appropriate value for $\boldsymbol{v}$ such that the system (19)-(24) when reduced in terms of the variables $\Delta\tilde{\boldsymbol{u}}$, $\Delta\boldsymbol{\beta}$ and $\Delta\boldsymbol{v}$ has the coefficient matrix

$$\left(\begin{array}{cccc|cccc|c} -(R_1-1)I & 0 & \cdots & 0 & (R_1-1)I & 0 & \cdots & 0 & 0 \\ 0 & -(R_2-1)I & \cdots & 0 & 0 & (R_2-1)I & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & -(R_K-1)I & 0 & 0 & \cdots & (R_K-1)I & 0 \\ \hline (R_1-1)I & 0 & \cdots & 0 & -(R_1-1)I & 0 & \cdots & 0 & I \\ 0 & (R_2-1)I & \cdots & 0 & 0 & -(R_2-1)I & \cdots & 0 & I \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \cdots & (R_K-1)I & 0 & 0 & \cdots & -(R_K-1)I & I \\ \hline 0 & 0 & \cdots & 0 & I & I & \cdots & I & 0 \end{array}\right),$$

where the first column block corresponds to $\Delta\tilde{\boldsymbol{u}}$, second column block corresponds to $\Delta\boldsymbol{\beta}$ and the last column block to $\Delta\boldsymbol{v}$. For $K \geq 2$, the above matrix is clearly singular (add the first block row to the second, then the second block row has linearly dependent rows).

4.3 Alternative Search Direction

Although the singularity of the matrix $\left[(R_j-1)\tilde{\boldsymbol{U}}_j+\boldsymbol{V}\right]$ arose rarely in practice, we present an alternate search direction that removes the usage of pseudo-inverses in the previous Subsection 4.2. Note that (16) is

$$(R_j-1)u_{ij}+v_i = \frac{\mu}{\beta_{ij}}.$$

Instead of multiplying both sides by β_{ij} and then linearizing, we can directly linearize the above. We use the first order approximation:

$$\frac{1}{\beta_{ij}+\Delta\beta_{ij}} \approx \frac{1}{\beta_{ij}} - \frac{\Delta\beta_{ij}}{\beta_{ij}^2}, \text{ for } \Delta\beta_{ij} \text{ small.}$$

Analogous derivations of search directions for interior-point methods have led to interesting algorithms.

The system (19)-(24) stays the same except (23) becomes

$$(R_j - 1)\Delta\tilde{\boldsymbol{u}}_j + \mu \boldsymbol{B}_j^{-2}\Delta\boldsymbol{\beta}_j + \Delta\boldsymbol{v} \;=\; \tilde{\boldsymbol{r}}_{5,j}, \tag{32}$$

where

$$\tilde{\boldsymbol{r}}_{5,j} := \mu \boldsymbol{B}_j^{-1}\boldsymbol{e} - (R_j - 1)\tilde{\boldsymbol{u}}_j - \boldsymbol{v}.$$

We define

$$\tilde{\boldsymbol{H}}_j := \left[H_{(1)\hat{\boldsymbol{u}}_j}\left(H_{(2)\hat{\boldsymbol{u}}_j} - \hat{\boldsymbol{U}}_j^{-1}\hat{\boldsymbol{T}}_j\right)^{-1} H_{(2)\tilde{\boldsymbol{u}}_j} - \left(H_{(1)\tilde{\boldsymbol{u}}_j} - \tilde{\boldsymbol{U}}_j^{-1}\tilde{\boldsymbol{T}}_j\right)\right].$$

Proposition 4.1 *Let $\tilde{\boldsymbol{u}} > 0$, $\hat{\boldsymbol{u}} > 0$, $\tilde{\boldsymbol{t}} > 0$, $\hat{\boldsymbol{t}} > 0$, $\beta \in (0,1)^{n_0 K}$, $v \in \mathbb{R}^n$ and $R_j > 1$, for every $j \in \{1, 2, \ldots, K\}$ be given. Then the system given by (19), (20), (21), (22), (32) and (24) has a unique solution.*

Proof. Note that $\tilde{\boldsymbol{H}}_j$ is the Schur complement of the $(1,1)$ block of

$$\begin{bmatrix} -H_{(1)\tilde{\boldsymbol{u}}_j} + \tilde{\boldsymbol{U}}_j^{-1}\tilde{\boldsymbol{T}}_j & -H_{(1)\hat{\boldsymbol{u}}_j} \\ -H_{(2)\tilde{\boldsymbol{u}}_j} & -H_{(2)\hat{\boldsymbol{u}}_j} + \hat{\boldsymbol{U}}_j^{-1}\hat{\boldsymbol{T}}_j \end{bmatrix}.$$

The positive definiteness of the above matrix was already established (Sun & Freund, 2004). Therefore, $\tilde{\boldsymbol{H}}_j$ is positive definite. Using (15), we have

$$\Delta\hat{\boldsymbol{t}}_j = \hat{\boldsymbol{U}}_j^{-1}\boldsymbol{r}_{2,j} - \hat{\boldsymbol{U}}_j^{-1}\hat{\boldsymbol{T}}_j\Delta\hat{\boldsymbol{u}}_j.$$

Using (20) we obtain

$$\Delta\hat{\boldsymbol{u}}_j \;=\; \left(\boldsymbol{H}_{(2)\hat{\boldsymbol{u}}_j} - \hat{\boldsymbol{U}}_j^{-1}\hat{\boldsymbol{T}}_j\right)^{-1}\left[\boldsymbol{r}_{1,j} - \hat{\boldsymbol{U}}_j^{-1}\boldsymbol{r}_{2,j} - \boldsymbol{H}_{(2)\tilde{\boldsymbol{u}}_j}\Delta\tilde{\boldsymbol{u}}_j\right]. \tag{33}$$

Now, using the last identity with (14) and (12) we arrive at

$$\Delta\tilde{\boldsymbol{u}}_j \;=\; \tilde{\boldsymbol{H}}_j^{-1}\left[(R_j-1)\Delta\boldsymbol{\beta}_j + \tilde{\boldsymbol{U}}_j^{-1}\tilde{\boldsymbol{r}}_{2,j} + H_{(1)\hat{\boldsymbol{u}}_j}\left(H_{(2)\hat{\boldsymbol{u}}_j} - \hat{\boldsymbol{U}}_j^{-1}\hat{\boldsymbol{T}}_j\right)^{-1}\left(\boldsymbol{r}_{1,j} - \hat{\boldsymbol{U}}_j^{-1}\boldsymbol{r}_{2,j}\right) - \tilde{\boldsymbol{r}}_{1,j}\right] \tag{34}$$

Define

$$\tilde{\boldsymbol{r}}_{6,j} := -\tilde{\boldsymbol{H}}_j^{-1}\left[\tilde{\boldsymbol{U}}_j^{-1}\tilde{\boldsymbol{r}}_{2,j} + H_{(1)\hat{\boldsymbol{u}}_j}\left(H_{(2)\hat{\boldsymbol{u}}_j} - \hat{\boldsymbol{U}}_j^{-1}\hat{\boldsymbol{T}}_j\right)^{-1}\left(\boldsymbol{r}_{1,j} - \hat{\boldsymbol{U}}_j^{-1}\boldsymbol{r}_{2,j}\right) - \tilde{\boldsymbol{r}}_{1,j}\right].$$

Substituting into (32) we obtain

$$\left[(R_j-1)^2\tilde{\boldsymbol{H}}_j^{-1} + \mu\boldsymbol{B}_j^{-2}\right]\Delta\boldsymbol{\beta} + \Delta\boldsymbol{v} \;=\; \tilde{\boldsymbol{r}}_{5,j} + (R_j-1)\tilde{\boldsymbol{r}}_{6,j}. \tag{35}$$

Note that since $\tilde{\boldsymbol{H}}_j$ and $\boldsymbol{B}_j$ are positive definite and $R_j > 1$, $\mu > 0$, we have that

$$\boldsymbol{L}_j := \left[(R_j-1)^2\tilde{\boldsymbol{H}}_j^{-1} + \mu\boldsymbol{B}_j^{-2}\right] \succ 0.$$

Thus, using (17), we arrive at

$$\Delta\boldsymbol{v} \;=\; \left(\sum_{j=1}^{K}\boldsymbol{L}_j^{-1}\right)^{-1}\left[-\tilde{\boldsymbol{r}}_4 + \sum_{j=1}^{K}\boldsymbol{L}_j^{-1}\left(\tilde{\boldsymbol{r}}_{5,j} + (R_j-1)\tilde{\boldsymbol{r}}_{6,j}\right)\right]. \tag{36}$$

(Positive definiteness of $\boldsymbol{L}_j$ implies the positive definiteness and hence the nonsingularity of $\left(\sum_{j=1}^{K}\boldsymbol{L}_j^{-1}\right)$.) Substituting back, we conclude

$$\Delta\boldsymbol{\beta}_j \;=\; \boldsymbol{L}_j^{-1}\left(\tilde{\boldsymbol{r}}_{5,j} + (R_j-1)\tilde{\boldsymbol{r}}_{6,j} - \Delta\boldsymbol{v}\right) \tag{37}$$

and $\Delta\tilde{\boldsymbol{u}}_j$ and $\Delta\hat{\boldsymbol{u}}_j$ are given by (34) and (35) respectively. The above computation also proves the uniqueness of the solution. □

For the numerical computation, we consider other ways of solving the linear system. Instead of solving for $\Delta\boldsymbol{v}$ first, it might be better to eliminate $\Delta\boldsymbol{v}$, $\Delta\boldsymbol{\beta}$ and reduce the system to a linear system involving only $\Delta\tilde{\boldsymbol{u}}$. We derive,

$$\Delta\boldsymbol{v} = \left(\sum_{\ell=1}^{K}\boldsymbol{B}_\ell^2\right)^{-1}\left[-\mu\tilde{\boldsymbol{r}}_4 + \sum_{\ell=1}^{K}\boldsymbol{B}_\ell^2\left(\tilde{\boldsymbol{r}}_{5,\ell} - (R_\ell-1)\Delta\tilde{\boldsymbol{u}}_\ell\right)\right]$$

and

$$\Delta\boldsymbol{\beta}_j = \frac{1}{(R_j-1)}\tilde{\boldsymbol{H}}_j\Delta\tilde{\boldsymbol{u}}_j + \frac{1}{(R_j-1)}\left[\tilde{\boldsymbol{r}}_{1,j} - \tilde{\boldsymbol{U}}_j^{-1}\tilde{\boldsymbol{r}}_{2,j} - \boldsymbol{r}_{7,j}\right],$$

where

$$\boldsymbol{r}_{7,j} := H_{(1)\hat{\boldsymbol{u}}_j}\left(H_{(2)\hat{\boldsymbol{u}}_j} - \hat{\boldsymbol{U}}_j^{-1}\hat{\boldsymbol{T}}_j^{-1}\right)^{-1}\left(\boldsymbol{r}_{1,j} - \hat{\boldsymbol{U}}_j^{-1}\boldsymbol{r}_{2,j}\right).$$

Substituting back into (32) we obtain

$$\left(\frac{\mu}{R_j-1}\boldsymbol{B}_j^{-2}\tilde{\boldsymbol{H}}_j + (R_j-1)(I - \tilde{\boldsymbol{B}}\boldsymbol{B}_j^2)\right)\Delta\tilde{\boldsymbol{u}}_j - (R_j-1)\tilde{\boldsymbol{B}}\sum_{\ell\neq j}\boldsymbol{B}_\ell^2\Delta\tilde{\boldsymbol{u}}_\ell$$

$$= \tilde{\boldsymbol{r}}_{5,j} - \frac{\mu}{R_j - 1}\boldsymbol{B}_j^{-2}\left(\tilde{\boldsymbol{r}}_{1,j} - \tilde{\boldsymbol{U}}_j^{-1}\tilde{\boldsymbol{r}}_{2,j} - \boldsymbol{r}_{7,j}\right) - \tilde{\boldsymbol{B}}\left(\sum_{\ell=1}^{K}\boldsymbol{B}_\ell^2\tilde{\boldsymbol{r}}_{5,j} - \mu\tilde{\boldsymbol{r}}_4\right), \quad j \in \{1, ..., K\},$$

where $\tilde{\boldsymbol{B}} := \left(\sum_{\ell=1}^{K}\boldsymbol{B}_\ell\right)^{-1}$.

5 Implementation Strategies for Branch-and-Bound

The computational performance of the branch-and-bound algorithm can depend heavily on the implementation. The following are the different strategies we tested – some pertaining to both Pure Branch-and-Bound and Convex Relaxation Branch-and-Bound, and some only to one of the approaches. This section illustrates some of the different strategies we tested.

5.1 Root Heuristic

A heuristic was run at the root node to find a good initial upperbound on the optimal objective value for both branch-and-bound versions. We tested three different methods for comparison:

1. **Total Volume times K**: As a benchmark, we used a trivial upperbound by calculating the volume of the minimum volume covering ellipsoid that covered all n points and multiplied it by K.

2. **k-means Heuristic**: The k-means algorithm using Euclidean distances was run with 1000 different starting points. We used two different criteria for the best solution – one corresponding to the minimum mean Euclidean distances from the center of the clusters (the traditional objective function for k-means) and the other corresponding to the minimum total volume of the covering ellipsoids associated to each cluster. We tested other variants of the k-means algorithm, such as using Mahalanobis distances with updating covariance matrices, but the traditional version of k-means gave the best solutions overall.

3. **Sampling Heuristic**: Unlike the k-means heuristic, the minimum volume ellipsoids are dependent only on the boundary points. Thus, sampling a small number of points and running either of the branch-and-bound algorithms gives solutions that are often close to optimal. Since we are assuming there are no outliers, we selected the sample points to include all of the boundary points of the covering ellipsoid that covers all n points, plus additional interior points. For the latter set of points, we randomly selected pn additional points from a uniform distribution, where $p \in (0, 1)$ is a user-defined value (e.g. $p = 0.1$ indicates that 10% of the data points were selected).

5.2 Branching Strategies

Different branching strategies in the branch-and-bound tree have significant impact in the total computation time of both branch-and-bound versions. We tested different strategies for node searches, variable selection and branching direction.

Node Search: We use depth-first-search for our node search strategy. Although best-bound search and other hybrid node search strategies are typically better in total computation time, depth-first-search often requires significantly less memory. For $K = 2$, we can implement this with constant memory usage.

Branching Variable Selection: The efficacy of a variable selection strategy is closely dependent on the branching order strategy as well. Because we are currently using depth-first-search only, these branching strategies are geared towards faster fathoming of the nodes. For both of the branch-and-bound strategies, we want to be able to assign the boundary points first so that many of the interior points would not have to be branched on. Out of many varieties, the following strategies worked best overall:

1. **Maximum Distance**: The unassigned point that is "farthest" from a cluster is branched. The distance is measured in terms of the Mahalanobis distance with respect to the cluster center. In Pure Branch-and-Bound, we calculate the optimal $\boldsymbol{M}_j$ and $\boldsymbol{z}_j$ for the assigned points, C_j. The farthest unassigned point would be $\boldsymbol{a}_r$ where $r = \text{ argmax }_{i \in \{1,\ldots,n\} \setminus \bigcup_{k=1}^{K} C_k} \|\boldsymbol{M}_j \boldsymbol{a}_i - \boldsymbol{z}_j\|^2$. In Relaxation Branch-and-Bound, if we want to choose an unassigned point to assign to Cluster j, we choose the point $\boldsymbol{a}_r$ if $r = \text{argmax}_{i \in \{1,\ldots,n\} \setminus \bigcup_{k=1}^{K} C_k} \{\beta_{i,j} + R(1 - \beta_{i,j}) - \tilde{t}_{i,j}\}$.
2. **Maximum Angle**: Together with the above strategy, we want the assigned points to be as spread out as possible. Thus, we use the measure proposed by (Sun & Freund, 2004) which chooses the point that has the largest angle from the current assigned points in E_j, i.e., we choose an unassigned point $\boldsymbol{a}_r$ to assign to Cluster j if $\sum_{i \in C_j} (\boldsymbol{M}_j \boldsymbol{a}_i - \boldsymbol{z}_j)^T (\boldsymbol{M}_j \boldsymbol{a}_r - \boldsymbol{z}_j) < 0$. If none of the points satisfies this criteria, we simply choose the point with the maximum distance.

From computational testing thus far, the combination of the two strategies seems to work best for fathoming the most nodes overall.

Branching Direction: We tested two branching strategies for both branch-and-bound versions:

1. **Constant Order**: In this strategy, when a data point is branched upon, it is first assigned to Cluster 1, then Cluster 2, et cetera, in consecutive order. The branch-and-bound tree would, thus, be building up the first cluster first, then the second, etc.
2. **Farthest Cluster**: Using the maximum distance strategy for variable selection, the cluster corresponding to the largest Mahalanobis distance for all of the unassigned points is branched on. The clusters will grow more uniformly compared to the Constant Order strategy.

Although the Farthest Cluster strategy seemed more intuitive, computational experimentation showed that one did not have significant advantages over the other in terms of the total number of nodes explored.

5.3 Warm Starting

In Convex Relaxation Branch-and-Bound, the relaxation problem at each node starts at a given starting value for $u_{i,j}$ and $\beta_{i,j}$. Unlike the pivoting-based methods, the Newton direction has no clear theoretical nor practical choice for a starting point given the optimal solution of the parent node. Thus, we tested different choices for the starting point:

1. **Solve from Scratch**: The default strategy is to solve from scratch, i.e., to not utilize the solution of the parent node. We use the starting point suggested by (Sun & Freund, 2004). We initialize with $u_{i,j} = \frac{d}{2n}$ and $\beta_{i.j} = \frac{1}{K}$.
2. **Resolve from Previous Point**: This resolve strategy starts with the solution of the parent node. Initializing at the optimal $u_{i,j}$ and $\beta_{i,j}$ leads to numerical instability, so we use an intermediary $u_{i,j}$ and $\beta_{i,j}$.

Experimentation thus far has not shown that the resolve strategy gives us any computational advantage. In most cases, the numerical instability of resolve leads to more iterations (e.g., 20 iterations versus 300) to solve the relaxation problem. The occasional cases where resolve did benefit does not seem to compensate for this overall.

5.4 Interior Point Elimination

At intermediary nodes, if an unassigned Point $\boldsymbol{a}_i$ is in the convex hull of the assigned points of Cluster j, then Point i must be in Cluster j for all subsequent nodes of that subtree. Detecting these interior points and assigning them to clusters can significantly reduce the size of our branch-and-bound tree. Also,

once they are considered interior points, they are not considered in the computation of the ellipsoids at each node. Especially for the Convex Relaxation Branch-and-Bound, eliminating these points as interior points can significantly improve the solution time of the relaxation problem. We tested several methods for detecting these interior points:

1. **Constructing the Convex Hull**: (Convex Relaxation Version Only) We use the `convhulln` procedure in Matlab (based on QHull ©, developed by the National Science and Technology Research Center for Computation and Visualization of Geometric Structures, University of Minnesota) to find all the facets that define the convex hull of the assigned points.

2. **Covering Ellipsoid**: For each cluster, we compute the minimum volume covering ellipsoid of the points assigned to that cluster. For the Pure Branch-and-Bound Version, these covering ellipsoids are simply those that are computed at each node. If an unassigned Point $\boldsymbol{a}_i$ is contained in that ellipsoid, we assign Point i to that cluster. For the Convex Relaxation Branch-and-Bound, the minimum volume covering ellipsoid is only an approximation of the convex hull, so it is possible that such a point will not be contained in the convex hull. Thus, this strategy is a heuristic for finding the interior points. In the Pure Branch-and-Bound Version, since the unassigned points do not affect the solution at each node, points that are previously assigned as interior points but become exterior points can be labeled unassigned again.

In larger dimensions ($d > 5$), we approximated the convex hull using the minimum volume ellipsoids that covered all the assigned points in that cluster.

5.5 Node Fathoming and Updating R_j

Typically, a node is fathomed in the branch-and-bound tree if the objective value of the relaxation is worse than the current upperbound. However, in the Convex Relaxation Branch-and-Bound, this objective value may not be a valid lowerbound to the optimal mixed-integer objective if our R_j is too small. We considered updating the value of R_j in the Convex Relaxation Branch-and-Bound as follows:

Test with $maxR$: We first check whether the current node should be fathomed by solving the relaxation problem using $R_j = maxR$, where $maxR$ is the theoretical upperbound for R_j of Appendix A. If the new objective value is less than the upperbound, then we cannot fathom the node. At this point, we have three strategies for updating R_j:

(a) Keep $R_j = maxR$,

(b) Use binary search to find an $R_j < maxR$ that also gives an objective value less than the upperbound. We use geometric mean as the midpoint.

(c) Double the value of R_j from the current value until the objective value is less than the upperbound.

We want R_j to be as small as possible for both numerical stability and for computational efficiency. Thus, we prefer updating R_j via binary search or by doubling. Although there is no theoretical basis for doubling, it worked best in practice. A key problem with this updating scheme is that the often $maxR$ is so large that the relaxation is not solvable due to numerical instability, or the solution value is unreliable. For overall computation time and numerical stability, the best approach would be to find better bounds on R_j or keep the value of R_j constant throughout.

6 A Refinement of the Model and the Assumptions

In many applications, we have some prior knowledge of the clusters. A plausible approach is similar to the one taken in the area of learning theory. Instead of clustering points without any prior information, we assume that for each cluster, we are given a small set of representative points. Let C_j^0 denote the indices of points that are known (a priori) to belong to Cluster j, where $C_j^0 \subset \{1, \ldots, n\}$ and $|C_j^0| \ll n$. We can use these representative points to approximate a value for R_j and incorporate them into our original problem as follows:

1. For each Cluster j, compute the smallest volume ellipsoid E_j containing those points $\boldsymbol{a}_i \in C_j^0$.

2. Approximate R_j,

   $R_j := d(\text{the smallest value } R \geq 2 \text{ such that } R \text{ expansion of } E_j \text{ contains all points } \boldsymbol{a}_i, \forall i \in \{1, \ldots, n\}).$

3. Initiate branch-and-bound with the points $\boldsymbol{a}_i$, $i \in C_j^0$ already branched to Cluster j ,$j = 1, ..., K$, at the root node.

The way it is stated above, our problem is related to *transduction* in *learning theory*. That is, given a set of points together with their cluster assignment (*the training set*) and an unassigned *test set*, the problem is to predict the cluster assignments of the points in the test set. Section 9 presents some computational results of our methods when provided with these representative points a priori.

7 Alternative Formulations

The main shortcoming of the formulation in Section 4 is the use of the parameter R_j. Since our initial approximations of R_j are often very conservative, it often results in poor relaxations. Large value of R_j also results in numerical instability and inaccuracy. The following are alternate formulations that we explored.

7.1 Reciprocal Formulation

We can consider

$$(\boldsymbol{M}_j\boldsymbol{a}_i - \boldsymbol{z}_j)^T (\boldsymbol{M}_j\boldsymbol{a}_i - \boldsymbol{z}_j) - \frac{1}{\beta_{ij}} + t_{ij} = 0,$$

in place of Constraint (3). However, the resulting relaxation is nonconvex.

7.2 Logarithmic Formulation

To protect the nice structure of the necessary conditions for local optimality, we may try

$$(\boldsymbol{M}_j\boldsymbol{a}_i - \boldsymbol{z}_j)^T (\boldsymbol{M}_j\boldsymbol{a}_i - \boldsymbol{z}_j) + \ln(\beta_{ij}) + t_{ij} = 1,$$

instead of Constraint (3). Again, the resulting relaxation is nonconvex. The necessary conditions for local optimality in the parameterized problems become

$$\begin{aligned}
h_{ij}(\boldsymbol{u}_j) + \ln(\beta_{ij}) + t_{ij} &= 1, \forall i \notin \bigcup_{j=1}^{K} C_j, j \in \{1,2,\ldots,K\};\\
h_{ij}(\boldsymbol{u}_j) + t_{ij} &= 1, \forall i \in C_j, j \in \{1,2,\ldots,K\};\\
u_{ij}t_{ij} &= \mu, \forall i \in \{1,2,\ldots,n\}, j \in \{1,2,\ldots,K\};\\
\tilde{u}_{ij} + v_i\beta_{ij} &= \mu, \forall i \notin \bigcup_{j=1}^{K} C_j, j \in \{1,2,\ldots,K\};\\
\sum_{j=1}^{k} \beta_{ij} &= 1, \forall i \notin \bigcup_{j=1}^{K} C_j;\\
u_{ij} > 0, \beta_{ij} > 0, t_{ij} &> 0, \qquad \forall i, \forall j.
\end{aligned}$$

7.3 Nonconvex Quadratic Formulation

Let y_{ij} denote some auxiliary variables. Then we can formulate the problem as follows.

$$\begin{array}{lrcll}
\text{Minimize} & -\sum_{j=1}^{K} \ln\left(\det\left(\boldsymbol{M}_j\right)\right) & & & \\
& \left(\boldsymbol{M}_j\boldsymbol{a}_i - \boldsymbol{z}_j\right)^T \left(\boldsymbol{M}_j\boldsymbol{a}_i - \boldsymbol{z}_j\right) & \leq & y_{ij}, & \forall i \notin \bigcup_{j=1}^{K} C_j, \forall j \in \{1,2,\ldots,K\}; \\
& \left(\boldsymbol{M}_j\boldsymbol{a}_i - \boldsymbol{z}_j\right)^T \left(\boldsymbol{M}_j\boldsymbol{a}_i - \boldsymbol{z}_j\right) & \leq & 1, & \forall i \in C_j, \forall j \in \{1,2,\ldots,K\}; \\
& \beta_{ij} y_{ij} & \leq & 1, & \forall i \notin \bigcup_{j=1}^{K} C_j, \forall j \in \{1,2,\ldots,K\}; \\
& \sum_{j=1}^{K} \beta_{ij} & = & 1, & \forall i \notin \bigcup_{j=1}^{K} C_j; \\
& \beta_{ij}^2 & = & \beta_{ij}, & \forall i \notin \bigcup_{j=1}^{K} C_j, \forall j \in \{1,2,\ldots,K\}.
\end{array}$$

This is a nonconvex quadratic optimization problem to which SDP based relaxation techniques can be applied. However, currently this approach seems hopeless for large scale problems.

We could also use a system of cubic inequalities

$$\beta_{ij}\left(\boldsymbol{M}_j\boldsymbol{a}_i - \boldsymbol{z}_j\right)^T \left(\boldsymbol{M}_j\boldsymbol{a}_i - \boldsymbol{z}_j\right) \leq 1.$$

There are many methods for obtaining convex relaxations of such systems (e.g., with auxiliary variables we can formulate these as quadratic inequalities). However, currently such approaches are not successful in problems with thousands of quadratic inequalities (which is our interest here).

7.4 Complementarity Based Formulation for Two Clusters

When $K = 2$, simpler, R_j-free formulations are possible. In particular, we let y_{i1}, y_{i2} denote the auxiliary variables so that if $i \in C_j$ then $y_{ij} = 0$. The formulation would be as follows:

$$\begin{array}{lrcll}
\text{Minimize} & -\sum_{j=1}^{K} \ln\left(\det\left(\boldsymbol{M}_j\right)\right) & & & \\
& \left(\boldsymbol{M}_j\boldsymbol{a}_i - \boldsymbol{z}_j\right)^T \left(\boldsymbol{M}_j\boldsymbol{a}_i - \boldsymbol{z}_j\right) & \leq & 1 + y_{i,j}, & \forall i \notin \bigcup_{j=1}^{K} C_j, \forall j \in \{1,2\}; \\
& \left(\boldsymbol{M}_j\boldsymbol{a}_i - \boldsymbol{z}_j\right)^T \left(\boldsymbol{M}_j\boldsymbol{a}_i - \boldsymbol{z}_j\right) & \leq & 1, & \forall i \in C_j, \forall j \in \{1,2\}; \\
& y_{i,1} y_{i,2} & \leq & 0, & \forall i \notin \bigcup_{j=1}^{K} C_j; \\
& y_{i,j} & \geq & 0, & \forall i \notin \bigcup_{j=1}^{K} C_j, \forall j \in \{1,2\}.
\end{array}$$

8 Computational Experiments I

We randomly generated $K = 2$ clusters from the Gaussian distribution for $d = 2$ (the dimension) and various values of n (the number of data points) and tested the performance of the k-means algorithm and the MVE algorithms.

Table 1 illustrates the accuracy of k-means with 1000 different starting points, the optimal MVE algorithm and the sampling heuristic of the optimal MVE algorithm. We included the latter heuristic from Subsection 5.1, that runs the optimal MVE algorithm on a sampled set of the points since it often gave accurate solutions efficiently. We used $p = 0.1$, i.e., we uniformly selected 10% of the points for the sampling heuristic. We measure the *accuracy* of the algorithms by their ability to capture the original Gaussian distributions.

The first column "Problem" is the problem name, the column "n_1" is the number of points generated from the first Gaussian distribution and the column "n_2" is the number of points generated from the second Gaussian distribution. The number of points that are wrongly assigned (out of the $n_1 + n_2$ points) by k-means, MVE and the sampling heuristic are shown in column "Train" under "k-means", "MVE", and "Sample MVE", respectively. In column "Sample MVE", entries with "*" indicate that the sampling heuristic found the optimal solution for that instance.

To test the robustness of these clusters, a new data set of the same size was generated from the original distributions. For the k-means cluster, a new point is assigned to the cluster with the closest center in terms of Euclidean distance. For the MVE algorithms, a new point is assigned to the cluster with the closest center in terms of the metric induced by the optimal ellipsoids. The number of new points that are incorrectly assigned are shown under column "Test". The last column "True" is the number of points that are wrongly assigned in the new test data set given the true distribution where a data point is assigned to the cluster of maximum likelihood. This value is presented as the benchmark lowerbound.

Table 2 illustrates the total running times of the different algorithms. A Linux desktop with Pentium 4 (2.4 GHz) and 1G of RAM was used for all computations. The column labeled "k-means" is the total CPU time for the k-means algorithm with 1000 iterations. The columns labeled "MVEH1", "MVEH2", and "MVEH3" are the total CPU times for the MVE Pure Branch-and-Bound algorithms of Section 3 using the root heuristic "Total Volume times K", k-means, and sampling, respectively. The column labeled "SampleMVE" is the total CPU times for the sampling heuristic. Note that the running times for MVEH2 and MVEH3 do *not* include the running times of the heuristics.

The running times of Convex Relaxation Branch-and-Bound of Section 4 are not shown since it was consistently worse than that of Pure Branch-and-Bound. For $n = 100$, the running times of both approaches were comparable, yet the convex relaxation version became significantly slower for larger n. Although the total number of branch-and-bound nodes explored is fewer for the convex relaxation due to its stronger bound, it did not compensate for the longer per node running time.

8.1 Results

In terms of accuracy in capturing the original Gaussian distribution, both MVE and sampling heuristic performed significantly better than the k-means algorithm for every instance. For example, for "500A" problems (500 points with asymmetric clusters) the percentages of wrongly assigned training points on average were 2% for MVE and 25% for k-means. For "1000A" problems, the training misclassification percentages were 3% for MVE and 22% for k-means, on average. Although the performance of MVE and the sampling heuristic did not seem to differ between equal size and different size (asymmetric) clusters, there was a clear disparity in the performance of the k-means algorithm. For example, for 500 data points, the average percentages of misclassification by k-means on the training and testing set were 15% and 13%, respectively, for equal clusters and 25% and 24%, respectively, for asymmetric clusters. For 1000 data points, the average percentages of misclassification by k-means on the training and testing set were 12% and 11%, respectively, for equal clusters and 22% and 20%, respectively, for asymmetric clusters. This is further empirical evidence that k-means suffers difficulty with asymmetric clusters as opposed to MVE.

Another interesting result is the performance of the sampling heuristic compared to the full implementation of MVE. Although only 10% of the total number of points were used, the accuracy of the sampling heuristic is nearly identical to the full MVE implementation. Surprisingly, the sample heuristic found the optimal solution in 50% of the problems. Also, there are cases where the accuracy of the Sample heuristic is better than MVE (e.g., P500E1, P500E3, P1000A3). Since the data was generated randomly from Gaussian distributions, it is possible that the optimal solution of our problem formulation in Section 2 may not capture the exact distribution association, as evident from the last column where there are wrongly assigned points even when the true distributions are known. Also, perhaps the sampling heuristic is less prone to the influence of outlier points.

The robustness performance of MVE is evident by comparing its results from the last column. In the testing set, both MVE and the sampling heuristic gave comparable results to the true distribution. Thus, the algorithms were able to successfully capture the original distribution of the data points.

As for running times, it appears that MVEH3 is the fastest over all MVE experiments due to its ability to find strong initial upperbounds. Again, the CPU times in Table 2 under "MVEH2" and "MVEH3" do not include the running time of the root heuristic. They show the total CPU seconds required for provable optimality after the root heuristics find the initial upperbounds. For example, for P100E1, the total running time for MVEH2 (MVE using solution from k-means as initial upperbound)

is $52.36 + 15.84 = 68.20$ CPU seconds. Similarly, the total running time for MVEH3 (MVE using solution from the sampling heuristic as initial upperbound) is $1.84 + 8.38 = 10.22$ CPU seconds. Also, the k-means running time includes the completion of all 1000 runs.

Even after incorporating the running times of the root heuristic, MVEH3 appears as the best method in terms of speed. The major exceptions are P1000E2 and P1000E3, though the sampling heuristic found the optimal solution in these problems. In these cases, the heuristic spent a longer time finding the optimal solution for the sampled points, which happend to include the boundary points of the ellipsoids. However, to show provable optimality for all the data points still took 240.55 CPU seconds for P1000E2 and 85.82 CPU seconds for P1000E3.

The advantage of finding strong initial upperbounds is apparent for n small, but its advantage seems to decrease with larger n. This is primarily due to the increased running time of the sampling heuristic. Comparing the running time between MVEH1 (MVE using basic root heuristic "Total Volume times K") and MVEH3 for P1000E2, P1000E3, P1000A1, and P1000A3, the strong initial upperbounds found by the sampling heuristic does not seem to compensate in faster total running time.

9 Computational Experiment II: Prior Information

This section illustrates the performance of the branch-and-bound algorithms and k-means algorithm when given representative points, as described in Section 6. Tables 3 and 4 illustrate the accuracy and running times of k-means, the optimal MVE algorithm and the sampling heuristic given prior knowledge regarding the central points of each cluster. For each cluster, d^2 points from the center of each cluster were given. Tables 5 and 6 illustrate the accuracy and running times of the k-means algorithm, MVE algorithm and sampling heuristic with prior information regarding extreme points for each cluster. For each cluster, d^2 points with the lowest probability of being in any other clusters were given.

9.1 Results

With both central and extreme information, the overall level of accuracy was very similar to that of Section 8 for both k-means and MVE algorithms. This is especially surprising for k-means since we expected it to reap significant advantages by having these prior information.

In the central information experiments, the misclassification accuracy of the sampling heuristic was slightly better overall for both training and testing sets compared to those without prior information. Also, the accuracy of k-means on P100A3 was significantly better than without prior information. Even

with theses changes, both the MVE and the sampling heuristic still performed consistently better than the k-means algorithm.

For the experiments with extreme information, the testing set accuracy of k-means was slightly better than with no prior information. For both MVE and the sampling heuristic, the accuracy of the training set was slightly better, but the testing set accuracy was virtually identical.

The significance of using prior information appears in the running times of the MVE algorithms. The running time for k-means is not greatly affected by the presence of central nor extreme prior information, as expected. However, there is a notable difference in the running times of MVEH1 and MVEH3 with prior information compared to those without prior information. For example, the average running time for MVEH1 for P1000A went from 548.13 CPU seconds to 188.38 CPU seconds with central information and to 290.28 CPU seconds with extreme information. Similarly, the average running time for MVEH3 for P1000A went from 136.33 CPU seconds to 119.10 CPU seconds with central information and 109.93 CPU seconds with extreme information.

10 Conclusion

We proposed using minimum volume ellipsoids (MVE) as a clustering criteria and modeled the problem as a mixed-integer semidefinite optimization problem. Compared to the popular k-means clustering algorithm, MVE is scale invariant, can handle non-spherical asymmetric clusters and can be solved to global optimality. We gave two solution approaches – one using pure branch-and-bound and the other using convex relaxation branch-and-bound. To improve the efficiency of the branching algorithm, we also considered several implementation strategies such as root heuristics, branching strategies, and interior point elimination. From computational experimentation on Gaussian distributions, we saw that the MVE approach was successful in capturing the original distribution of the data points and was far more accurate than the k-means algorithm. Notably, the sampling heuristic, which ran the branch-and-bound algorithm on only 10% of the data points, often found the optimal solution at a fraction of the total running time.

These promising results provide at least preliminary support for using MVE for Gaussian data. However, there are clearly several follow-up work that must be explored. For example, the computational experiments were only conducted for bivariate data. We saw that the branch-and-bound method became significantly impaired in higher dimensions due to the sparsity of the data points or "curse of dimensionality" suffered also by the k-means algorithm. The key challenge was eliminating interior

points, as discussed in Subsection 5.4, which becomes more difficult with larger dimensions and thus significantly increased the number of nodes explored in the branching.

Another extension is to explore alternative formulations with stronger relaxations. The Convex Relaxation Branch-and-Bound formulation of Section 4 gave stronger relaxations at each node than the Pure Branch-and-Bound approach of Section 3, but its advantage did not compensate for its longer per node running time. Alternative formulations were proposed in Section 7, yet these formulations are currently not tractible. For MVE to be a practical clustering algorithm for higher dimensions, we would need to find a stronger formulation that can greatly cut down the size of the branch-and-bound tree.

A very encouraging findings from our computational experimentation is the success of the sampling heuristic. Again, since ellipsoids are defined only by their boundary points, we only need to run the MVE clustering algorithm on the boundary points. Perhaps for data mining practitioners who deal with Gaussian or ellipsoidal data sets, this sampling heuristic is a viable technique since provable optimality may not be critical. Also, our empirical experiments have shown that the heuristic often finds solution equal to or close to the optimal solution. We hope that these preliminary results would spur interest in the intersecting fields of optimization, data mining and machine learning to further consider minimum volume ellipsoid as a clustering technique.

Problem	n_1	n_2	**k-means**		**MVE**		**Sample MVE**		**True**
			Train	Test	Train	Test	Train	Test	Test
P100E1	50	50	6	3	4	2	4*	2	2
P100E2	50	50	14	12	14	5	4	4	3
P100E3	50	50	16	21	0	0	0*	0	0
P100A1	20	80	12	13	0	0	0*	0	0
P100A2	20	80	3	0	1	0	8	14	0
P100A3	20	80	27	23	1	1	15	8	1
P500E1	250	250	64	45	3	12	2	11	11
P500E2	250	250	70	55	0	0	0*	0	0
P500E3	250	250	91	90	103	52	59	67	44
P500A1	350	150	173	153	3	3	3	4	3
P500A2	350	150	73	98	9	10	13	11	14
P500A3	350	150	135	111	21	3	21*	3	5
P1000E1	500	500	47	48	36	40	35	46	42
P1000E2	500	500	190	168	63	45	63*	45	32
P1000E3	500	500	126	111	20	31	20*	31	29
P1000A1	800	200	312	283	42	92	42*	92	91
P1000A2	800	200	213	152	11	15	11*	15	17
P1000A3	800	200	137	150	36	33	23	29	28

Table 1: Wrongly Assigned points for k-means, MVE, and Sample MVE

Problem	**k-means**	**MVE**H1	**MVE**H2	**MVE**H3	**Sample MVE**
P100E1	52.36	36.92	15.84	8.38	1.84
P100E2	39.07	22.10	11.31	9.31	2.65
P100E3	61.15	21.82	21.60	6.97	2.10
P100A1	70.58	23.18	23.16	8.46	2.56
P100A2	99.69	54.29	17.19	25.01	1.83
P100A3	61.32	23.45	22.61	22.63	1.53
P500E1	622.66	176.17	132.09	110.04	40.16
P500E2	323.43	42.54	40.34	33.46	28.18
P500E3	379.54	147.45	113.34	112.56	38.44
P500A1	422.93	68.69	65.25	35.48	36.90
P500A2	460.29	171.40	50.41	16.43	51.74
P500A3	311.62	162.95	34.54	18.46	33.38
P1000E1	1159.11	615.10	268.68	214.63	262.24
P1000E2	1014.76	283.33	266.39	240.55	247.29
P1000E3	922.85	235.06	113.78	85.82	236.58
P1000A1	636.73	293.75	207.49	141.03	316.59
P1000A2	1070.32	916.31	175.95	116.58	359.15
P1000A3	1819.76	434.33	404.51	151.37	302.22

Table 2: Running Times for k-means, MVE, and Sample MVE

Problem	n_1	n_2	**k-means**		**MVE**		**Sample MVE**		**True**
			Train	Test	Train	Test	Train	Test	Test
P100E1	50	50	10	5	3	2	2	6	2
P100E2	50	50	14	12	14	5	4	4	3
P100E3	50	50	20	21	0	0	0*	0	0
P100A1	20	80	11	13	0	0	0*	0	0
P100A2	20	80	3	0	1	0	0	0	0
P100A3	20	80	5	3	1	1	15	8	1
P500E1	250	250	64	45	3	12	2	11	11
P500E2	250	250	70	55	0	0	0*	0	0
P500E3	250	250	91	90	43	52	59	67	44
P500A1	350	150	183	144	3	3	3	4	3
P500A2	350	150	78	102	9	10	13	11	14
P500A3	350	150	135	111	21	3	21*	3	5
P1000E1	500	500	47	48	33	40	35	46	42
P1000E2	500	500	190	168	66	45	66*	45	32
P1000E3	500	500	125	111	21	31	21*	31	29
P1000A1	800	200	312	283	47	92	44	92	91
P1000A2	800	200	213	152	13	15	13*	15	17
P1000A3	800	200	142	177	35	33	23	29	28

Table 3: Wrongly Assigned points for k-means, MVE, and Sample MVE with Prior Central Information

Problem	**k-means**	**MVE**H1	**MVE**H2	**MVE**H3	**Sample MVE**
P100E1	70.79	26.92	15.88	16.82	1.30
P100E2	40.75	9.42	5.08	4.20	1.48
P100E3	60.13	16.90	15.51	5.04	3.75
P100A1	54.29	10.37	9.13	3.01	2.07
P100A2	97.59	10.33	5.36	5.20	2.28
P100A3	62.45	21.54	17.68	18.07	1.81
P500E1	347.48	66.44	53.43	49.45	37.38
P500E2	318.49	104.24	78.50	35.24	33.55
P500E3	374.37	46.32	33.39	28.02	35.30
P500A1	410.51	116.50	104.60	34.75	36.95
P500A2	475.94	142.68	69.76	20.29	38.41
P500A3	276.09	46.95	29.73	23.92	36.29
P1000E1	1155.78	255.28	181.05	152.60	261.58
P1000E2	1011.36	221.62	207.14	127.77	242.40
P1000E3	1072.22	177.59	95.95	73.38	221.14
P1000A1	831.93	161.32	147.12	127.34	294.67
P1000A2	1150.95	255.63	162.97	103.65	340.81
P1000A3	1760.00	148.19	145.73	126.31	288.42

Table 4: Running Times for k-means, MVE, and Sample MVE with Prior Central Information

Problem	n_1	n_2	**k-means**		**MVE**		**Sample MVE**		**True**
			Train	Test	Train	Test	Train	Test	Test
P100E1	50	50	12	6	0	3	2	6	2
P100E2	50	50	14	12	4	6	4*	4	3
P100E3	50	50	20	9	0	0	0*	0	0
P100A1	20	80	6	11	0	0	0*	0	0
P100A2	20	80	3	0	1	0	0	0	0
P100A3	20	80	23	10	1	1	9	3	1
P500E1	250	250	40	33	3	12	2	11	11
P500E2	250	250	71	60	0	0	0*	0	0
P500E3	250	250	94	92	77	52	62	69	44
P500A1	350	150	173	153	4	3	3	4	3
P500A2	350	150	85	105	5	10	13	11	14
P500A3	350	150	141	119	17	3	17*	3	5
P1000E1	500	500	47	48	27	40	35	46	42
P1000E2	500	500	194	168	66	45	66*	45	32
P1000E3	500	500	123	109	20	31	20*	31	29
P1000A1	800	200	316	291	41	91	41*	92	91
P1000A2	800	200	220	167	13	15	13*	15	17
P1000A3	800	200	143	177	23	33	23	29	28

Table 5: Wrongly assigned points for k-means, MVE, and Sample MVE with Prior Extreme Information

Problem	**k-means**	**MVE**H1	**MVE**H2	**MVE**H3	**Sample MVE**
P100E1	68.80	22.92	15.15	12.81	2.05
P100E2	42.07	8.50	4.13	3.65	1.38
P100E3	65.12	11.17	10.36	1.93	1.78
P100A1	87.10	17.67	17.00	8.34	2.98
P100A2	81.78	16.93	9.33	8.43	2.18
P100A3	75.94	17.98	15.47	8.04	1.85
P500E1	375.05	61.86	46.17	44.79	32.41
P500E2	423.31	203.16	89.85	23.42	31.88
P500E3	446.05	32.63	24.04	23.48	36.87
P500A1	457.65	114.25	61.95	19.39	48.03
P500A2	400.93	169.55	79.34	29.33	59.90
P500A3	428.45	362.61	138.41	22.51	46.38
P1000E1	1119.17	326.57	247.26	223.68	280.47
P1000E2	1218.30	241.32	210.57	172.18	252.65
P1000E3	811.77	635.08	349.66	150.39	244.77
P1000A1	764.15	132.93	89.16	95.48	297.72
P1000A2	1143.96	630.80	331.93	148.34	354.43
P1000A3	1406.11	107.10	96.36	85.98	284.48

Table 6: Running Times for k-means, MVE, and Sample MVE with Prior Extreme Information

A Estimating R

Let $\bar{\boldsymbol{a}}_1, \bar{\boldsymbol{a}}_2, \ldots, \bar{\boldsymbol{a}}_{d+1}$ be affinely independent such that all these points belong to the same cluster, wlog, 1. Hence they all lie in the ellipsoid defined by $(\boldsymbol{M}_1, \boldsymbol{z}_1)$. We have

Lemma A.1 *Suppose $\bar{\boldsymbol{a}}_i \in \mathbb{R}^d$ lies in the ellipsoid defined by $(\boldsymbol{M}_1, \boldsymbol{z}_1)$. Then*

$$\|\boldsymbol{z}_1\| - 1 \leq \|\boldsymbol{M}_1 \bar{\boldsymbol{a}}_i\| \leq \|\boldsymbol{z}_1\| + 1.$$

Proof. Since $\bar{\boldsymbol{a}}_i$ lies in the ellipsoid defined by $(\boldsymbol{M}_1, \boldsymbol{z}_1)$, we have the following string of implications:

$$\begin{aligned} \left(\boldsymbol{M}_1 \bar{\boldsymbol{a}}_i - \boldsymbol{z}_1\right)^T \left(\boldsymbol{M}_1 \bar{\boldsymbol{a}}_i - \boldsymbol{z}_1\right) \leq 1 \quad &\Longleftrightarrow \quad \bar{\boldsymbol{a}}_i^T \boldsymbol{M}_1^2 \bar{\boldsymbol{a}}_i - 2\boldsymbol{z}_1^T \boldsymbol{M}_1 \bar{\boldsymbol{a}}_i + \boldsymbol{z}_1^T \boldsymbol{z}_1 - 1 \leq 0 \\ &\Rightarrow \quad \|\boldsymbol{M}_1 \bar{\boldsymbol{a}}_i\|^2 - 2\|\boldsymbol{z}_1\| \|\boldsymbol{M}_1 \bar{\boldsymbol{a}}_i\| + \|\boldsymbol{z}_1\|^2 - 1 \leq 0. \end{aligned}$$

Now, treating the last expression as a quadratic function of $\|\boldsymbol{M}_1 \bar{\boldsymbol{a}}_i\|$, we obtain that the following bounds are implied (for every $i \in \{1, 2, \ldots, d+1\}$):

$$\|\boldsymbol{z}_1\| - 1 \leq \|\boldsymbol{M}_1 \bar{\boldsymbol{a}}_i\| \leq \|\boldsymbol{z}_1\| + 1.$$

□

We can use this trivial lemma to get some rough bounds on R. For every ellipsoid $(\boldsymbol{M}_j, \boldsymbol{z}_j)$, we have such set of $\bar{\boldsymbol{a}}_i$. Suppose $\boldsymbol{a}_r = \sum_{i=1}^{d+1} \lambda_i \bar{\boldsymbol{a}}_i$, for $\boldsymbol{e}^T \boldsymbol{\lambda} = 1$. That is, $\boldsymbol{a}_r$ is expressed as an affine combination of $\bar{\boldsymbol{a}}_1, \bar{\boldsymbol{a}}_2, \ldots, \bar{\boldsymbol{a}}_{d+1}$. We can do this for every r, since the affine combinations of $\bar{\boldsymbol{a}}_1, \bar{\boldsymbol{a}}_2, \ldots, \bar{\boldsymbol{a}}_{d+1}$ span the whole space $\mathbb{R}^d$. Then

$$\begin{aligned} \left(\boldsymbol{M}_j \boldsymbol{a}_r - \boldsymbol{z}_j\right)^T \left(\boldsymbol{M}_j \boldsymbol{a}_r - \boldsymbol{z}_j\right) &\leq \left\| \sum_{i=1}^{d+1} \lambda_i \left(\boldsymbol{M}_j \bar{\boldsymbol{a}}_i\right) \right\|^2 + 2 \sum_{i=1}^{d+1} |\lambda_i| \, \|\boldsymbol{M}_j \bar{\boldsymbol{a}}_i\| \, \|\boldsymbol{z}_j\| + \|\boldsymbol{z}_j\|^2 \\ &\leq \sum_{i,\ell} |\lambda_i \lambda_\ell| \, \|\boldsymbol{M}_j \bar{\boldsymbol{a}}_i\| \, \|\boldsymbol{M}_j \bar{\boldsymbol{a}}_\ell\| + 2 \sum_{i=1}^{d+1} |\lambda_i| \, \|\boldsymbol{M}_j \bar{\boldsymbol{a}}_i\| \, \|\boldsymbol{z}_j\| + \|\boldsymbol{z}_j\|^2 \\ &\leq \sum_{i,\ell} |\lambda_i \lambda_\ell| \left(\|\boldsymbol{z}_j\| + 1\right)^2 + 2 \sum_{i=1}^{d+1} |\lambda_i| \left(\|\boldsymbol{z}_j\|^2 + \|\boldsymbol{z}_j\| \right) + \|\boldsymbol{z}_j\|^2 \\ &\leq \left[(d+1) \left(\|\boldsymbol{z}_j\| + 1\right) \lambda_{\max} + \|\boldsymbol{z}_j\| \right]^2, \end{aligned}$$

where $\lambda_{\max} := \max \{ |\lambda_i| : i \in \{1, 2, \ldots, d+1\} \}$.

References

Barnes, E. 1982. An algorithm for separating patterns by ellipsoids. *IBM J. Research and Development*, **26**, 759–764.

Dunagan, J., & Vempala, S. 2001. Optimal outlier removal in high-dimensional spaces. *ACM Symp. on Theory of Computing*, 627–636.

John, F. 1948. Extreme problems with inequalities as subsidiary conditions. *Studies and Essays for Courant Anniversary*, 187–204.

Khachiyan, L. 1996. Rounding polytopes in the real number model of computation. *Mathematics Operations Research*, **21**, 307–320.

Khachiyan, L., & Todd, M. J. 1993. On the complexity of approximating the maximal inscribed ellipsoid for a polytope. *Mathematicsal Programming*, **61**, 137–159.

Kumar, M., Orlin, J., & Patel, N. R. 2003. *Scale invariant clustering.* In Preparation.

Kumar, P., & Yıldırım, E. A. 2003 (May). *Approximate minimum volume enclosing ellipsoids using core sets.* manuscript.

Löwner, K. 1934. Über monotone matrixfunktionen. *Mat. Z.*, **38**, 177–216.

Nesterov, Y., & Nemirovskii, A.S. 1994. *Interior-Point Polynomial Algorithms in Convex Programming.* Philadelphia, PA: SIAM.

Rosen, J.B. 1965. Pattern separation by convex programming. *J. Math. Anal. Appl.*, **10**, 123–134.

Rousseeuw, P. J., & Leroy, A. M. 1987. *Robust Regression and Outlier Detection.* NY, NY: Wiley.

Sun, P., & Freund, R. M. 2004. Computation of minimum volume covering ellipsoids. *Operations Research*, **52**, 690–706.

Symons, M.J. 1981. Clustering Criteria and Multivariate Normal Mixtures. *Biometrics*, **37**, 35–43.

Toh, K. 1999. Primal-dual path-following algorithms for determinant maximization problems with linear matrix inequalities. *Comp. Opt. Appl.*, **14**, 309–330.

Zhang, Y., & Gao, L. 2004. On numerical solution of the maximum volume ellipsoid problem. *SIAM J. Optim.*, **14**, 53–76.