Clustering
via
Minimum Volume Ellipsoids

Romy Shioda * Levent Tunçel†

May 25, 2005, revised: January 20, 2006

Abstract

We propose minimum volume ellipsoids (MVE) clustering as an alternative clustering technique to k-means for data clusters with ellipsoidal shapes and explore its value and practicality. MVE clustering allocates data points into clusters in a way that minimizes the geometric mean of the 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 "ellipsoidal" distributions and compare its performance with the k-means clustering algorithm as well as the MCLUST algorithm (which is based on a maximum likelihood EM algorithm) available in the statistical package R.

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 criterion 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. [16] 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 [15].

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 [8]. 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.

MVE 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 [10] and [13]. [5] developed significant amount of theory and [1] proposed a heuristic algorithm based on the eigenvalue decomposition. The current best complexity bound is due to [6]. Other interesting theoretical algorithms are presented in [7] and [12]. Recently, many advances have been made to provide more practical algorithms with good foundations in convex optimization [18, 17, 15, 9]. We extend the algorithm developed by [15] 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 [14] and [3]). 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 point out that our techniques can be very sensitive to the existence of outliers. While this

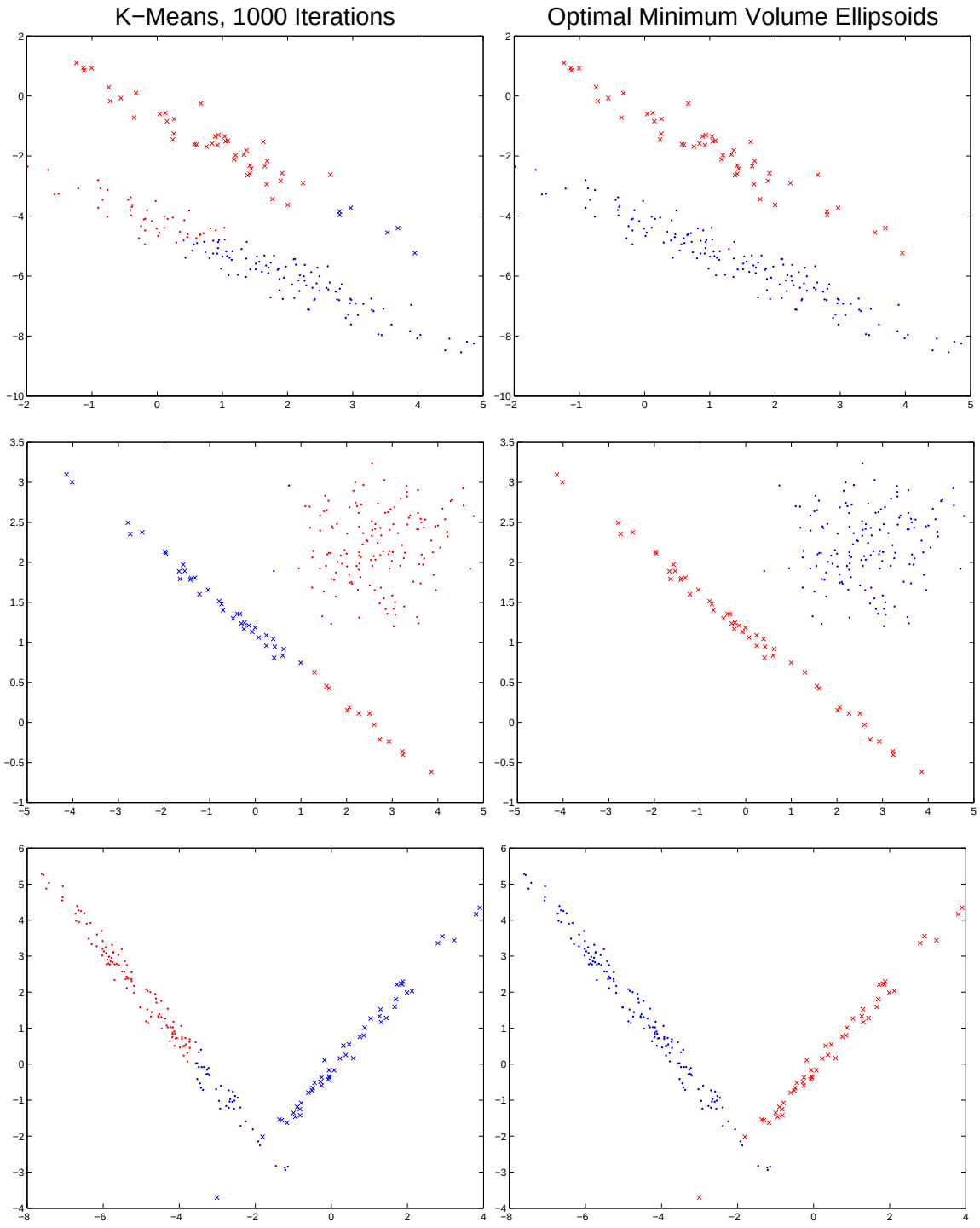

Figure 1: Three examples of two bivariate Gaussian distributions. The original distribution is represented with an "o" (150 points) and "x" (50 points). The clustering algorithms (k-means with 1000 iterations on the left and optimal minimum volume ellipsoids on the right) clustered the points into red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

by itself suggest that we can use the sensitivity of our techniques for detecting outliers, our algorithms (though theoretically sound) would not be practical to adapt for outlier detection. Investigation of heuristic algorithms using approximate MVEs may be a fruitful direction for future research.

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, uniform ellipsoidal distribution and an asymmetric ellipsoidal distribution. We compare its performance with the k-means clustering algorithm and the model-based EM algorithm, MCLUST.

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 highlights additional interpretations of our model and some pitfalls to avoid. Section 6 describes several implementation strategies for both of the branch-and-bound algorithms. Section 7 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 8 describes several alternative nonconvex formulations of the problem. Section 9 describes the computational experimentations and illustrates the results from applying MVE clustering compared to k-means clustering and MCLUST. Section 10 shows analogous computational results for the prior information case described in Section 7. 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). We use the trace inner-product in this space, given $A, B \in \mathbb{S}^d$,

$$\langle A, B \rangle := \text{trace}(AB).$$

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 geometric mean of the volumes 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.,

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

However, as in [15] we define it by

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

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

The 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 $-\sum_{j=1}^{K} \ln\det(\boldsymbol{M}_j) \geq UB$, 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 6.

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

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

where the positive definiteness of $\boldsymbol{M}_j$ is enforced by the domain of the objective function. Clearly, the above problem is separable across j, thus we solve for the minimum volume covering ellipsoid separately for each cluster using the Dual Reduced Newton algorithm proposed by [15].

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.

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 branch-and-bound 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:

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

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

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

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

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

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

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.

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. This upperbound for Cluster j and point $\boldsymbol{a}_r$ is

$$\left[(d+1)\left(\|\boldsymbol{z}_j\|+1\right)\lambda_{\max}+\|\boldsymbol{z}_j\|\right]^2,$$

where $\|\boldsymbol{z}_j\|$ (the center of the ellipsoid for cluster j) can be bounded and

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

(the maximum absolute value of the affine combination coefficients for point $\boldsymbol{a}_r$ based on $(d+1)$ affinely independent points known to belong to Cluster j—without the knowledge of such $(d+1)$ points, finding a good bound on $\lambda_{\max}$ can be hard).

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 us 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 $\mu > 0$):

$$\begin{aligned}
\min \quad & -\sum_{j=1}^{K} \ln\det(\boldsymbol{M}_j) - \mu \sum_{i=1}^{n}\sum_{j=1}^{K} \left[\ln(t_{ij}) + \ln(\beta_{ij})\right] \\
\text{subject to:} \quad & (\boldsymbol{M}_j\boldsymbol{a}_i - \boldsymbol{z}_j)^T(\boldsymbol{M}_j\boldsymbol{a}_i - \boldsymbol{z}_j) + (R_j - 1)\beta_{ij} + t_{ij} = R_j, \quad \forall i \notin \bigcup_{j=1}^{K} C_j, \forall j; \\
& (\boldsymbol{M}_j\boldsymbol{a}_i - \boldsymbol{z}_j)^T(\boldsymbol{M}_j\boldsymbol{a}_i - \boldsymbol{z}_j) + t_{ij} = 1, \quad \forall i \in C_j, \forall j; \\
& \sum_{j=1}^{K} \beta_{ij} = 1, \quad \forall i \notin \bigcup_{j=1}^{K} C_j.
\end{aligned}$$

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 [15], 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 [15]) 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 alternative search direction that removes the usage of pseudo-inverses in 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 $\tilde{u} > 0$, $\hat{u} > 0$, $\tilde{t} > 0$, $\hat{t} > 0$, $\beta \in (0,1)^{n_0 K}$, $v \in \mathbb{R}^n$ 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 [15]. 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 Further Interpretations and Some Pitfalls to Avoid

Our objective function is to minimize a convex function of $\boldsymbol{M}_1, \boldsymbol{M}_2, \ldots, \boldsymbol{M}_K$. This is significant for the following reasons: First, we would like to arrive at a good convex optimization problem ("good" meaning efficiently solvable and a good approximation of the initial non-convex problem). To find such a relaxation, we usually find an outer approximation to the non-convex feasible solution set. Secondly, if the objective function is also non-convex, we need to approximate (in our case, over-estimate the objective function by a convex function). Because of our choice of the objective function, we get to skip this second step of approximation. Thus, for any convex relaxation of the feasible region, we immediately have a corresponding convex optimization problem whose objective value is a lower bound on the optimal objective value of the original problem (1).

In our implementation and experiments, we assumed that for every Cluster j, there are at least $(d+1)$ affinely independent points $\boldsymbol{a}_i$ which belong to that Cluster j. When there is only one cluster (the MVE problem addressed in [15]), this assumption is clearly reasonable and can be efficiently checked in practice, before any optimization method is employed to compute the MVE or an approximation of it. The situation is different in our more general setting of multiple clusters, since the assignment of points to clusters is part of our decision variable. Moreover, without any additional knowledge about the data, this assumption is not as easily verifiable, a priori. Therefore, the users of our techniques will have to be familiar enough with their data (or ensure that the assumption holds by a suitable preprocessing step) so that they will know a priori that every cluster is "full-dimensional" (the users will also have to decide what affine independence means numerically —i.e., to what accuracy). For instance, the setting of Section 6 describes a "learning" situation in which such a strong assumption can be verified easily.

In our branch-and-bound algorithms, we do not evaluate the objective function until every cluster has at least $(d+1)$ affinely independent points assigned to it by branching (i.e., if this condition fails at a node of the branch-and-bound tree, we branch further; if the condition fails at a leaf node, we eliminate the node). Therefore, our branch-and-bound algorithms minimize the geometric mean of the volumes of MVEs where every MVE is full-dimensional and with finite volume.

Note that if the number of points $\boldsymbol{a}_i$ is extremely large and/or the assumption of affine independence is nearly violated, some numerical difficulties might occur in our algorithms. Also, under such conditions, our branch-and-bound algorithms might favor solutions in which one or more clusters are covered by MVEs with extremely skinny ellipsoids. However, some of these problems can be circumvented by the introduction of various convex constraints. For example, we can add the linear inequality

$$\sum_{j=1}^{K} \text{trace}\,(\boldsymbol{M}_j) \leq \bar{R},$$

for a large enough constant $\bar{R}$ (or we can add a convex quadratic constraint which serves a similar purpose). Since the matrices $\boldsymbol{M}_j$ are symmetric and positive definite, such linear inequality ensures that all the eigenvalues of all the matrices are in the interval $(0, \bar{R})$. However, introduction of such a constant $\bar{R}$ must be done very carefully (again, additional knowledge about the data is necessary). We will have a discussion of similar constants later in Appendix A.

To discuss our choice of the objective function, suppose we have only two clusters, $K = 2$, and we have a complete assignment of points to the clusters leading to the MVEs: E_1 and E_2. Furthermore, suppose that we have another complete assignment of the points to the clusters leading to the MVEs E_1' and E_2' such that

$$\text{vol}\left(E_1'\right) = 2\text{vol}\left(E_1\right).$$

Then, the second assignment is better than the first if and only if

$$\text{vol}\left(E_2'\right) < \frac{1}{2}\text{vol}\left(E_2\right).$$

Thus, our objective function considers relative "blow-up factors" of the ellipsoids rather than their actual volumes. On the one hand, this kind of invariance property can be sometimes desirable (when the real distributions have covariance matrices with different magnitudes—say $\|\Sigma_1\| \ll \|\Sigma_2\|$, our objective function will treat these two distributions on more equal ground—doubling the volume of the first ellipsoid is equivalent to doubling the volume of the second ellipsoid).

On the other hand, if the users want to minimize the *sum* of the volumes of MVEs, then this can be easily handled by our Pure Branch-and-Bound Method (Section 3) after modifying the objective function; however, we could no longer use our Convex Relaxation Branch-and-Bound Method (Section 4) which is tailored for minimizing the geometric mean of the volumes of MVEs.

6 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.

6.1 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).

6.2 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 [15] 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, etc., 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.

6.3 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 [15]. 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.

6.4 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.

6.5 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 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.

7 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 10 presents some computational results of our methods when provided with these representative points a priori.

8 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 alternative formulations that we explored.

8.1 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.

8.2 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

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

8.3 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).

8.4 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:

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

9 Computational Experiments I

We tested our MVE algorithms against the k-means algorithm and MCLUST's EMclust routine, a model-based clustering method that uses an EM algorithm for Gaussian mixture models [4]. Both

k-means and MVE algorithms were implemented in MATLAB ®[11] and MCLUST is implemented in R.

We randomly generated data points from a Gaussian distribution, a uniform ellipsoidal distribution, and an asymmetric ellipsoidal distribution. The uniform ellipsoidal distribution has points uniformly distributed in a given ellipsoid. We used the MATLAB ® code developed by J. Berkardt[2] to generate these points. An example of this distribution is shown in Figure 2. The asymmetric ellipsoidal distribution is an off-centered version of the Gaussian distribution, where the center of mass of the data points do not correspond to the center of the ellipsoid. An example of this distribution is shown in Figure 3. For both of these distributions, the positive semi-definite matrix and the center of the ellipsoids were generated randomly. Similarly, the covariance matrix and the mean were randomly generated for the Gaussian distribution. In all of our experiments, we had $K = 2$ clusters and $d = 2$ and $d = 5$ (the dimension).

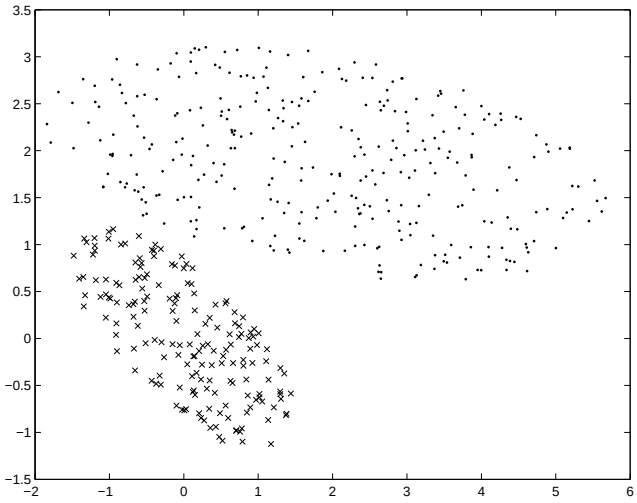

Figure 2: Points from two uniform ellipsoidal distribution

Tables 1 and 3 illustrate the accuracy of k-means with 1000 different starting points, MCLUST, the optimal MVE algorithm and the sampling heuristic of the optimal MVE algorithm on the Gaussian data points with $d = 2$ and $d = 5$, respectively. We included the latter heuristic from Subsection 6.1, 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

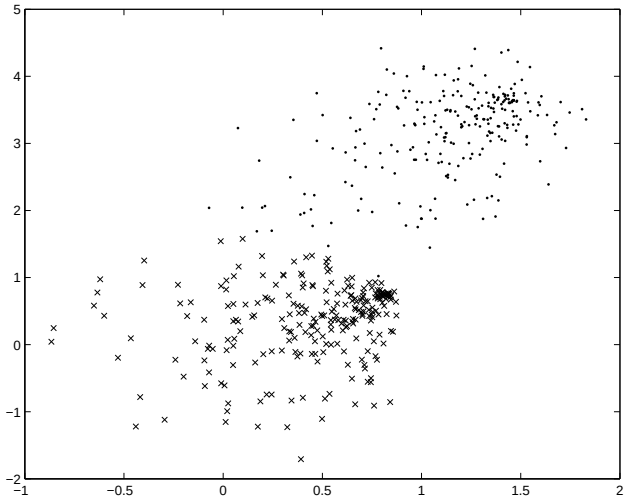

Figure 3: Points from two asymmetric ellipsoidal distribution

from the first Gaussian distribution and the column "n_2" is the number of points generated from the second Gaussian distribution. Three types of distributions were generated for a given number of points, as denoted by "E", "U", and "M" in the problem name. "E" refers to equal sizes, where $n_1 = n_2$. "U" refers to unequal sizes, where $n_1 \neq n_2$. "M" refers to two clusters of differing covariance matrices, where one cluster's covariance matrix has entry magnitudes five times larger than the other.

The number of points that are wrongly assigned (out of the $n_1 + n_2$ points) by k-means, MCLUST, MVE and the sampling heuristic are shown in column "Train" under "k-means","MCLUST", "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 MCLUST, a new point is assigned to a cluster according to the Bayesian information (maximum log-likelihood) criterion using the fitted model parameters [4]. 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, where a data point is assigned to the cluster of maximum likelihood, given the mean and covariance matrix of the true distributions. This value is presented as the benchmark, though it does not necessarily produce the fewest assignment errors as shown in the experiments.

This measure of robustness, along with the use of maximum likelihood as the assignment criteria, may optimistically bias the result of MCLUST which is based on maximum likelihood estimation. A potential alternative is to compare the various approaches in terms of how well they estimated the geometric centers of the "distributions" and the "covariance matrices" of the "distributions." These can be easily obtained from our optimal solutions (from $\boldsymbol{z}_j$'s and $\boldsymbol{M}_j$'s). One could then look at the norms of the difference of the true center and $\boldsymbol{z}_j$ and compare the true covariance matrix Σ_j to the information provided by the eigenvectors (and the eigenvalues) of $\boldsymbol{M}_j$ properly scaled. Such investigations are left for future work.

Tables 2 and 4 illustrate the total running times of the different algorithms for the Gaussian data points with $d = 2$ and $d = 5$. 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 column labelled "MCLUST" is the total CPU time for MCLUST. 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.

Analogous to the Gaussian distribution results, Tables 5 and 7 correspond to the accuracy of the different methods using the uniform distribution with $d = 2$ and $d = 5$, respectively. The corresponding running times are shown in Tables 6 and 8, respectively. Tables 9 and 11 show the accuracy of the methods using the asymmetric data with $d = 2$ and $d = 5$, respectively. The corresponding running times are shown in Tables 10 and 12, respectively. The assignment for the test set was done as in the Gaussian data for the uniform and asymmetric data. In addition, the column "True" is analogous to before, where the true center and shape of the ellipsoid are given.

9.1 Results

This section summarizes the results from Tables 1–12.

Gaussian data:

For the Gaussian data sets, it is clear from Tables 1–4 that MCLUST is the superior clustering technique, both in terms of accuracy and computation time. This is not surprising since MCLUST is tailored to detect Gaussian distributions, but the degree of its success is still impressive. In comparing MVE and k-means, MVE dominates k-means in most cases (the exceptional cases are discussed in the following paragraph). Relative to MCLUST and MVE, it appears as though k-means has more difficulty clustering the unequal clusters ("U" problems) and clusters with differing magnitudes ("M" problems), though this appears to be problem dependent. In almost all cases, the Sample MVE algorithm performs as well (and sometimes better) than the MVE algorithm in terms of accuracy.

We further investigate the problems that MVE performed especially poorly, namely G100d2M#2 and G500d2M#1. For Problem G100d2M#2, Figure 4 illustrates the true cluster membership and the cluster membership resulting from MVE. This appears to be a case where minimizing the product of the volumes of ellipsoids may not be appropriate, since one cluster has near negligible volume. Such a situation may be avoided by modifying the objective to minimize the sum of the volumes, as discussed in Section 5. For Problem G500d2M#1, Figure 5 illustrates the true cluster membership and the cluster membership resulting from MVE. This is clearly a case where incorporating density information gives significant advantage to the clustering method. MCLUST considers the positions and densities of the points when building a cluster, whereas MVE only considers the shape of the cluster's border. Thus, if there are known spatial distributions (such as Gaussian), a method such as MCLUST would be more appropriate than MVE.

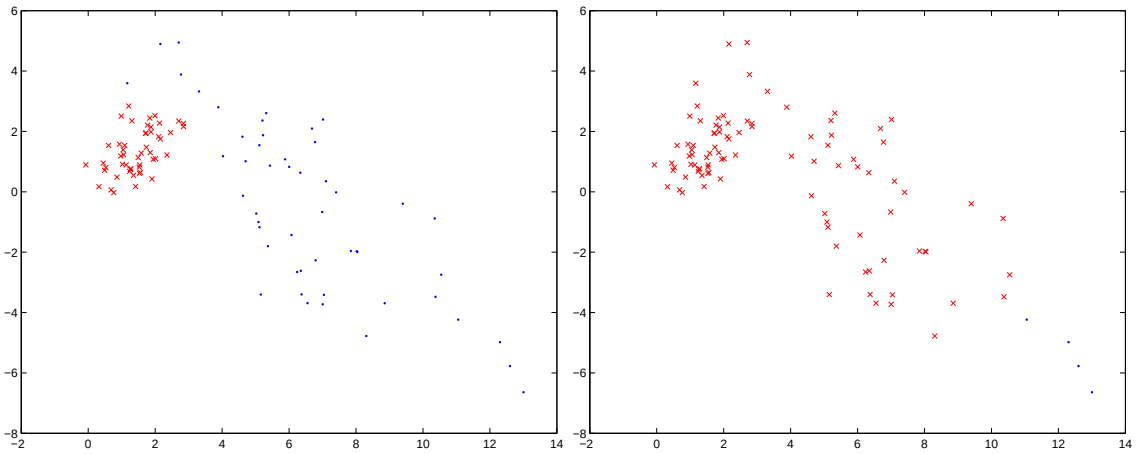

Figure 4: Illustration of Problem G100d2M#2. The left plot shows the true cluster memberships and the right plot shows the cluster membership resulting from MVE.

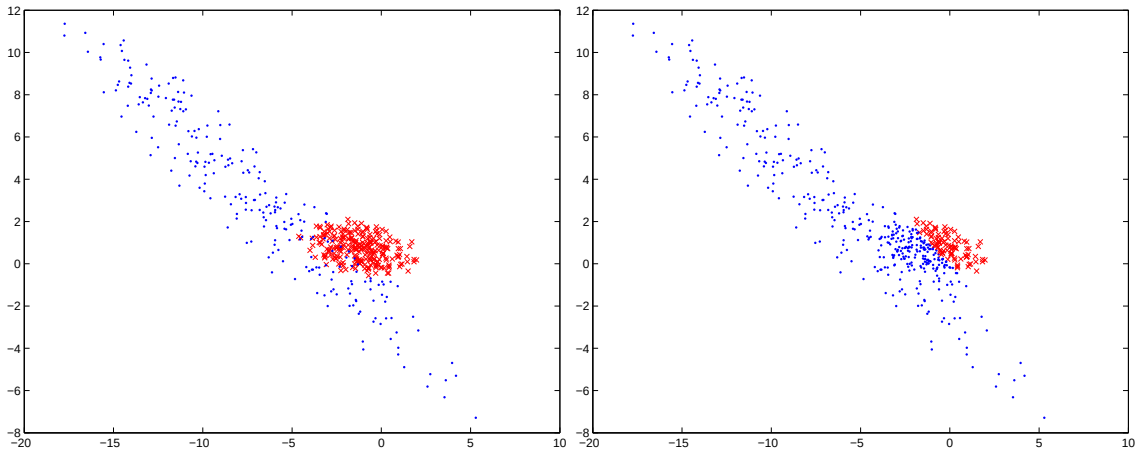

Figure 5: Illustration of Problem G500d2M#1. The left plot shows the true cluster memberships and the right plot shows the cluster membership resulting from MVE.

In terms of computation time, again MCLUST is clearly the dominant method. MVE is faster than k-means for $d = 2$ (Table 2), though this is for k-means repeated 1000 times. For $d = 5$, the running time for MVE increases dramatically. Almost half of the problems cannot be solved to provable optimality within 7200 CPU seconds. This explosion in computation time from $d = 2$ and $d = 5$ is mainly due to the "curse of dimensionality". The key to MVE's tractability for $d = 2$ was due to the elimination of interior points, as described in Section 6.4. For $d = 2$, a majority of the points did not need to be branched upon because they were contained in the interior of at least one of the ellipsoids. However, as d increases, the sparseness of the points increase for a given n. Thus, fewer points can be eliminated as interior points, requiring us to branch on significantly more points. With the increased height of the branch-and-bound tree, the number of nodes and thus the total running time increase exponentially. However, it appears that the "sub-optimal" (at least not provably optimal) solutions are not too poor in terms accuracy (Table 3). Also, as before, the Sample MVE appears to provide good solutions, with faster running times. Perhaps for higher dimensional data sets, solving MVE to provable optimality is not a tractable computation, but sub-optimal solutions might still provide good predictions.

Uniform data:

MVE performs well under the uniform data, losing to MCLUST on just two problems: U100d2#2 and U500d2#2. Figure 6 illustrates the true cluster membership and the cluster membership from MVE of Problem U100d2#2. The MVE clustering does not appear unreasonable from simple inspection. Since the data is derived from the uniform distribution, it is not clear how MCLUST utilized density

information to capture the training set more accurately. Perhaps this data set contains too few data points from each cluster to fully capture the shape of the clusters. For Problem U500d2#2, Figure 7 illustrates the true cluster membership and the MVE's cluster assignments. The errors in assignments are largely due to the overlapped region of the two clusters. For MVE, assignment of points in that region to one cluster or the other is due to the order in which the points were selected in the branch-and-bound algorithm. Perhaps points contained in both ellipsoids should be treated differently, instead of being given a hard assignment.

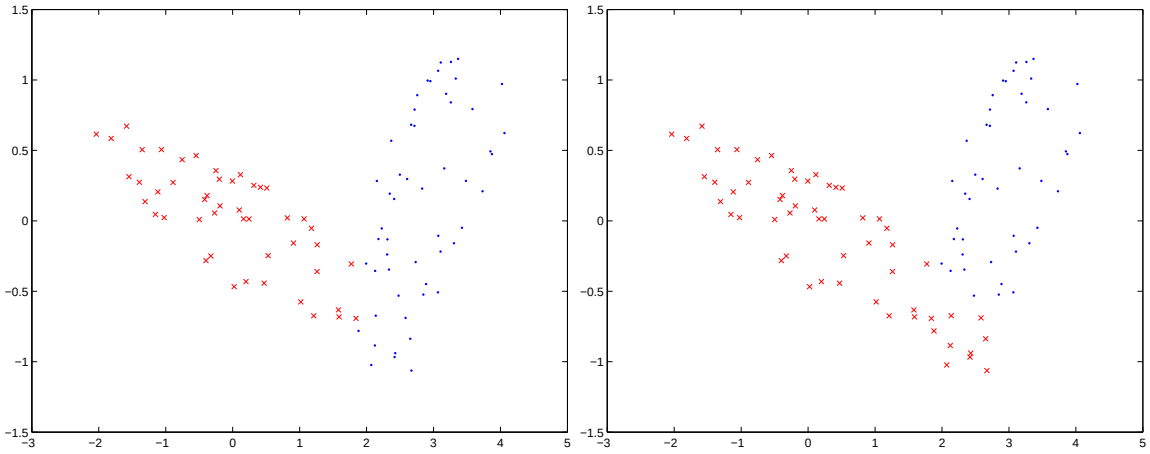

Figure 6: Illustration of Problem U100d2#2. The left plot shows the true cluster memberships and the right plot shows the cluster membership resulting from MVE.

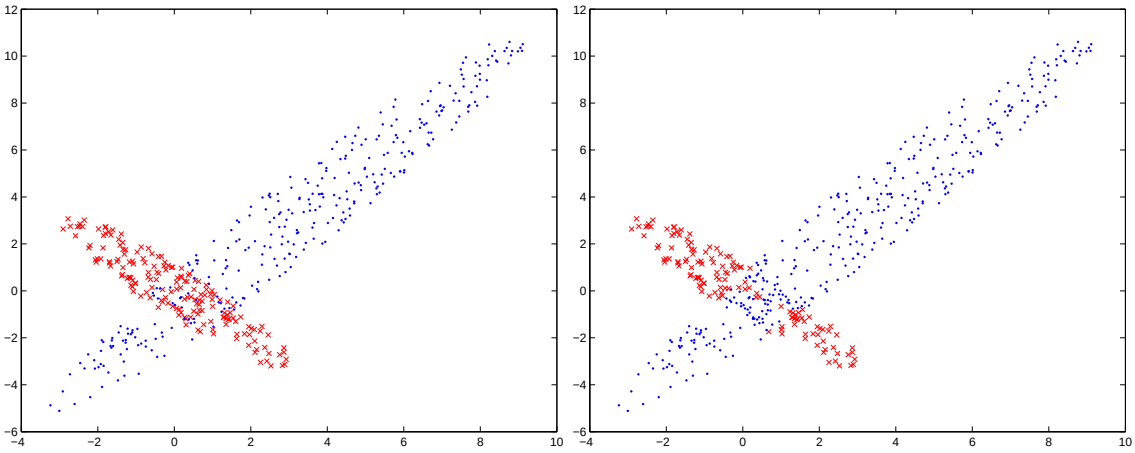

Figure 7: Illustration of Problem U500d2#2. The left plot shows the true cluster memberships and the right plot shows cluster membership resulting from MVE.

Again, the Sample MVE algorithm appears to perform well, both in terms of accuracy and running

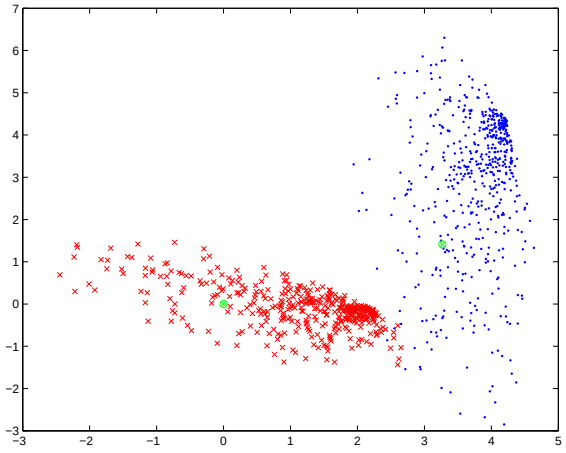

Figure 8: Illustration of Problem A1000d2#1.

time. In the $d = 5$ case, the Sample MVE resulted in either the optimal assignment or the best assignment within 7200CPU seconds. All of the Sample MVE times were under 7200CPU seconds.

MVE could not solve two of the problems to provable optimality within the time limit, but both cases resulted in few assignment erros (one having 0 mistakes, and other other having fewer mistakes than MCLUST in the training set).

Asymmetric Data:
Similar to the uniform data, MVE results in fewer assignment errors with the asymmetric data than both k-means and MCLUST. For $d = 5$, both MCLUST and MVE has no errors for all the problems tested. For $d = 2$, there are cases where Sample MVE performs rather poorly (e.g., A100d2#1 and A500d2#2). These are the cases where the sampled data points did not effectively capture the shape of the two clusters. This is improved by the inclusion of prior information as discussed in Section 7.

Some of the assignment errors under column "True" in Table 9 are relatively high compared to MCLUST and MVE. An example of this is Problem A1000d2#1, illustrated in Figure 8, where the green (x) mark indicates the center of the ellipsoid. As mentioned earlier, the "True" assignment uses the center of the ellipsoid, not the center of mass, to assign test points. Given that the points are assigned using maximum likelihood, the number of assignment errors is not surprising for A1000d2#1, where the center of mass of one cluster is close to the boundary of another. In such cases, nearest neighbor assignment might have produced better assignments in the test data. However, it would be difficult to determine the appropriate approach without any prior distributional information.

From the computational results thus far, it appears that MCLUST is the superior method when there are ellipsoidal distributions in the data set. It is able to use relative positions of the points in the Euclidean space to accurately detect its original distribution. MVE does not consider densities of the data points—it only looks for the boundary points of ellipsoids that yield the total minimum volume. If the data points do not have ellipsoidal distributions, such as the uniform and asymmetric distributions, then MVE appears to perform well with respect to MCLUST. In these cases, the previous shortcoming of MVE becomes an asset, since an ellipsoidal distribution assumption can throw off a method such as MCLUST. Perhaps a practical solution is to test the data set first with MCLUST, then use its result as an initial solution (root heuristic) to the MVE algorithm. Thus, if the data set comes from an ellipsoidal distribution, MCLUST will provide the solution efficiently. In other cases, MVE may provide improvement in the cluster assignments.

10 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 7. MCLUST was not tested in these experiments since it was not clear how to incorporate prior information from its software interface. Tables 13 and 15 illustrate the accuracy of the clustering algorithms on Gaussian data with central information for $d = 2$ and $d = 5$, respectively. The corresponding running times are in Tables 14 and 16, respectively. For uniform data, Tables 21 and 25 show the accuracy of the algorithms for $d = 2$ and $d = 5$, respectively, and the corresponding running times are shown in Tables 22 and 26, respectively. For the asymmetric data, Tables 29 and 33 show the accuracy of the algorithms for $d = 2$ and $d = 5$, respectively, and the corresponding running times are shown in Tables 30 and 34, respectively. For each cluster, d^2 points near the center of each cluster are given. In the case of the asymmetric data, points near the center of mass of the points are given instead of the center of the ellipsoids.

Tables 17 and 19 illustrate the accuracy and running times of the clustering methods on Gaussian data for $d = 2$ and $d = 5$, respectively. For uniform data, Tables 23 and 27 show the accuracy of the algorithms for $d = 2$ and $d = 5$, respectively, and the corresponding running times are shown in Tables 24 and 28, respectively. For the asymmetric data, Tables 31 and 35 show the accuracy of the algorithms for $d = 2$ and $d = 5$, respectively, and the corresponding running times are shown in Tables 32 and 36, respectively. For each cluster, d^2 points with the lowest "probability" of being in any other clusters were given.

10.1 Results

The following gives an overview of the results shown in Tables 13–36.

Gaussian Data:

For $d = 2$, the central and extreme prior information does not uniformly improve the accuracy for MVE. In some cases, prior central information improves accuracy (e.g. G100d2M#2). Also, extreme information improves accuracy in some cases (e.g., G100d2E#2 and G500d2M#1), but slightly worsens in others. The accuracy of k-means seems to improve with prior central information but not significantly for extreme information. The main impact of the prior information is in the computation time. For both central and extreme information, the total running time for MVE greatly improved with prior information. The running times for k-means and Sample MVE remain the same, as expected.

For $d = 5$, the accuracy of MVE improves often due to the faster computation time that allows for provable optimality within the time limit. The major improvement is seen with extreme information (Table 19). With central information, six of the previously sub-optimal solutions were solved to provable optimality. With extreme information, 12 of the previously sub-optimal problems were solved to provable optimality, leaving just one problem that could not be solved to optimality within the time limit. The extreme information provides data points on the boundary of the clusters, thus it is not surprising that this information helps MVE capture the optimal ellipsoid shapes early on, thus speeding up the total running time.

Uniform Data:

For $d = 2$, both prior central and extreme information appear to slightly improve the accuracy of MVE, but not significantly. This is not surprising since the assignment errors of the original clustering (Table 5) was already low. Since MVE captures the correct shape of the clusters without the prior information, these additional information do not seem to contribute much in terms of accuracy. The accuracy of the Sample MVE improves for some cases, but not for all.

For $d = 5$, the accuracy level also remains relatively the same with prior information for both MVE and k-means. However, the running times for MVE improve significantly, especially with extreme information. Capturing the correct boundary points seem to have a great impact for the branch-and-bound algorithm, which can fathom significantly more nodes corresponding to interior points. The central information does not provide the boundary points, thus the running times are the same or sometimes worse. The performances of the k-means algorithm and Sample MVE stay relatively the same.

Asymmetric Data: For $d = 2$, the accuracy remains relatively the same for both k-means and MVE. However, the Sample MVE improves significantly with both prior information, especially for A500d2#1. Again, the major impact for both central and extreme information is on the total running time of MVE. The running time for Sample MVE and k-means remain the same. For $d = 5$, there is no change in accuracy with prior information, which is not surprising since MVE had no assignment errors with the original data. Again, the major impact is seen in computation time with extreme information.

For all three data types, the major impact of using prior information is in the running times of MVE. In many cases, extreme information helped MVE find the correct cluster, thereby eliminating many of the interior points early on.

11 Conclusion

We proposed using minimum volume ellipsoids (MVE) as a clustering criterion 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 ellipsoidal 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. MCLUST was clearly the best method for Gaussian data, but MVE performed relatively well for the non-Gaussian distributions. 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 ellipsoid-shaped clusters. However, there are clearly several follow-up research directions that must be explored. For example, the computational experiments were conducted for $d = 2$ and $d = 5$. 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 6.4, which becomes more difficult with larger dimensions and thus significantly increases 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 8, yet these formulations are currently not tractable. 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 finding 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, this sampling heuristic would be a viable technique since provable optimality may not be critical.

Our experiments showed that with Gaussian data, the MCLUST package is indeed outstanding as expected. A careful look at our design of our experiments will reveal that we treated maximum likelihood clustering as the "ideal clustering" that we try to match in performance. One important difference between MCLUST and our MVE approach is that MCLUST tries to capture the "distribution" whereas MVE is completely geometric and is invariant under the local densities of the points (except of course for the sampling heuristic). If the application indicates that there is an underlying distribution, then MCLUST would be a better choice. However, in applications where there may not be a distribution, or one tends to apply a non-parametric technique, our MVE algorithms may be a great alternative. As we pointed out in Section 9, using MCLUST as the root heuristic in our MVE algorithms may be the current best overall approach. Also, our empirical experiments have shown that the sampling 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**		**MCLUST**		**MVE**		**Sample MVE**		**True**
			Train	Test	Train	Test	Train	Test	Train	Test	Test
G100d2E#1	50	50	6	3	1	2	4	2	4*	2	2
G100d2E#2	50	50	14	12	4	2	14	5	4	4	3
G100d2E#3	50	50	16	21	0	0	0	0	0*	0	0
G100d2U#1	20	80	12	13	0	0	0	0	0*	0	0
G100d2U#2	20	80	3	0	2	0	1	0	8	14	0
G100d2U#3	20	80	31	23	1	1	1	1	15	8	1
G100d2M#1	50	50	10	3	0	0	0	2	8	2	2
G100d2M#2	50	50	14	13	0	8	46	49	42	36	12
G100d2M#3	50	50	26	27	3	5	12	7	27	25	7
G500d2E#1	250	250	62	35	1	11	3	12	2	11	11
G500d2E#2	250	250	70	55	0	0	0	0	0*	0	0
G500d2E#3	250	250	91	90	48	56	60	52	59	67	44
G500d2U#1	350	150	173	153	2	4	3	3	3	4	3
G500d2U#2	350	150	73	98	4	10	9	10	13	11	14
G500d2U#3	350	150	135	111	3	4	21	3	21*	3	5
G500d2M#1	250	250	122	115	30	38	158	142	109	129	60
G500d2M#2	250	250	73	63	7	13	33	18	9	15	19
G500d2M#3	250	250	95	92	0	2	0	2	0*	2	2
G1000d2E#1	500	500	47	48	18	43	36	40	35	46	42
G1000d2E#2	500	500	190	168	34	29	63	45	63*	45	32
G1000d2E#3	500	500	126	111	21	28	20	31	20*	31	29
G1000d2U#1	800	200	307	278	56	56	42	92	42*	92	91
G1000d2U#2	800	200	213	152	12	5	11	15	11*	15	17
G1000d2U#3	800	200	137	150	17	25	36	33	23	29	28
G1000d2M#1	500	500	151	151	0	2	0	12	9	30	12
G1000d2M#2	500	500	296	315	1	7	0	21	27	25	21
G1000d2M#3	500	500	208	183	63	59	45	55	76	79	50

Table 1: Wrongly Assigned points for k-means, MCLUST, MVE, and Sample MVE on Gaussian Data with $d = 2$.

Problem	**k-means**	**MCLUST**	**MVE**H1	**MVE**H2	**MVE**H3	**Sample MVE**
G100d2E#1	23.24	0.08	26.45	10.87	3.97	0.70
G100d2E#2	17.34	0.07	15.23	8.21	4.40	1.26
G100d2E#3	28.45	0.07	11.26	10.50	3.28	1.04
G100d2U#1	29.02	0.07	4.12	4.12	4.12	1.30
G100d2U#2	46.94	0.10	52.30	50.66	12.16	0.93
G100d2U#3	31.24	0.08	29.10	17.38	11.10	0.78
G100d2M#1	23.51	0.06	25.12	16.73	11.33	1.14
G100d2M#2	23.72	0.08	29.26	28.91	28.22	1.05
G100d2M#3	29.11	0.07	14.67	14.51	11.79	1.16
G500d2E#1	353.56	0.61	134.72	115.57	54.65	20.61
G500d2E#2	173.42	0.65	168.56	50.84	16.52	15.11
G500d2E#3	209.09	0.62	140.34	60.46	56.07	20.09
G500d2U#1	213.95	0.57	163.52	34.54	17.61	18.64
G500d2U#2	253.83	0.59	70.46	23.16	8.17	25.60
G500d2U#3	162.51	0.60	73.23	50.34	9.22	16.94
G500d2M#1	201.03	0.59	193.45	178.62	108.13	24.59
G500d2M#2	262.35	0.56	172.68	20.93	18.60	27.72
G500d2M#3	226.02	0.58	159.73	23.75	9.22	19.18
G1000d2E#1	671.39	2.62	495.16	206.77	106.15	127.89
G1000d2E#2	612.62	2.82	284.35	175.95	116.53	118.54
G1000d2E#3	545.46	2.64	253.34	110.53	41.74	113.93
G1000d2U#1	430.33	2.49	129.63	124.25	68.60	154.19
G1000d2U#2	658.26	2.73	360.34	123.51	54.90	171.43
G1000d2U#3	1000.45	2.84	260.35	235.06	73.42	147.60
G1000d2M#1	861.53	2.36	693.63	321.52	305.09	124.88
G1000d2M#2	644.69	2.61	489.06	435.23	337.16	182.49
G1000d2M#3	655.99	2.55	524.03	262.43	223.66	129.80

Table 2: Running Times for k-means, MCLUST, MVE, and Sample MVE on Gaussian Data with $d = 2$.

Problem	n_1	n_2	**k-means**		**MCLUST**		**MVE**		**Sample MVE**		**True**
			Train	Test	Train	Test	Train	Test	Train	Test	Test
G500d5E#1	250	250	2	0	0	0	0	0	0	0	0
G500d5E#2	250	250	0	0	0	0	0	0	0*	0	0
G500d5E#3	250	250	1	0	0	0	0	0	0*	0	0
G500d5U#1	350	150	12	14	0	0	0	0	0*	0	0
G500d5U#2	350	150	74	40	0	0	0	0	0*	0	0
G500d5U#3	350	150	18	27	2	0	32	17	32*	17	1
G500d5M#1	250	250	59	57	0	0	0	0	0*	0	0
G500d5M#2	250	250	14	11	0	0	0	0	2	0	0
G500d5M#3	250	250	14	18	1	0	40	16	17	20	14
G1000d5E#1	500	500	0	0	0	0	0	0	0	0	0
G1000d5E#2	500	500	0	0	0	0	0	0	0*	0	0
G1000d5E#3	500	500	55	59	8	13	27	12	27*	12	11
G1000d5U#1	800	200	20	33	0	2	0	2	0*	2	2
G1000d5U#2	800	200	67	108	20	26	22	32	20	32	37
G1000d5U#3	800	200	32	21	4	1	6	3	6*	3	1
G1000d5M#1	500	500	69	65	0	0	11	0	1	0	0
G1000d5M#2	500	500	89	98	0	0	14	9	14*	9	1
G1000d5M#3	500	500	0	0	0	0	0	0	0*	0	0
G2000d5E#1	1000	1000	79	111	2	6	7	7	9	9	8
G2000d5E#2	1000	1000	16	36	0	1	1	5	1*	5	0
G2000d5E#3	1000	1000	195	314	7	8	4	8	4*	8	8
G2000d5U#1	600	1400	17	26	0	0	0	0	0*	0	0
G2000d5U#2	600	1400	63	75	0	0	17	13	17*	13	14
G2000d5U#3	600	1400	34	45	0	0	0	0	0*	0	0
G2000d5M#1	1000	1000	33	78	0	0	0	0	0*	0	0
G2000d5M#2	1000	1000	377	345	0	1	2	1	2*	1	0
G2000d5M#3	1000	1000	212	202	7	10	27	48	27*	48	43

Table 3: Wrongly Assigned points for k-means, MCLUST, MVE, and Sample MVE on Gaussian Data with $d = 5$.

Problem	**k-means**	**MCLUST**	**MVE**H3	**Sample MVE**
G500d5E#1	130.02	0.80	2016.94	265.29
G500d5E#2	105.24	0.69	216.91	866.30
G500d5E#3	115.51	0.78	3526.81	452.47
G500d5U#1	203.23	0.67	3931.80	444.94
G500d5U#2	304.01	0.83	301.47	168.73
G500d5U#3	275.95	0.79	7200+	1008.56
G500d5M#1	209.39	0.68	912.55	186.97
G500d5M#2	161.93	0.79	4845.53	204.09
G500d5M#3	178.52	0.69	7200+	1264.77
G1000d5E#1	242.49	3.23	3693.77	795.75
G1000d5E#2	249.13	3.21	2716.56	880.27
G1000d5E#3	589.35	3.37	7200+	1660.31
G1000d5U#1	1041.51	3.06	4374.86	692.39
G1000d5U#2	2110.43	2.97	2042.68	1326.89
G1000d5U#3	1244.91	3.10	7200+	937.92
G1000d5M#1	473.94	2.94	1933.71	1012.65
G1000d5M#2	578.00	3.11	7200+	1198.86
G1000d5M#3	284.31	2.90	235.87	885.13
G2000d5E#1	1636.10	16.08	7200+	1996.61
G2000d5E#2	1643.15	15.28	7200+	7200+
G2000d5E#3	2856.74	15.25	7200+	2873.70
G2000d5U#1	1952.23	15.57	1539.37	1640.25
G2000d5U#2	2195.20	16.18	7200+	4084.89
G2000d5U#3	1583.60	15.28	7200+	7200+
G2000d5M#1	1474.02	14.85	7200+	2200.97
G2000d5M#2	2212.93	15.16	7200+	5943.29
G2000d5M#3	2312.11	14.78	7200+	3811.58

Table 4: Running Times for k-means, MCLUST, MVE, and Sample MVE on Gaussian Data with $d = 5$ and a time limit of 7200 CPU seconds .

Problem	n_1	n_2	**k-means**		**MCLUST**		**MVE**		**Sample MVE**		**True**
			Train	Test	Train	Test	Train	Test	Train	Test	Test
U100d2#1	31	69	23	30	27	30	0	0	0*	0	3
U100d2#2	47	53	4	3	1	10	9	6	22	14	3
U100d2#3	45	55	12	7	0	2	0	0	37	52	0
U500d2#1	287	213	77	74	0	0	0	0	3	1	8
U500d2#2	174	326	152	166	35	29	59	41	39	40	48
U500d2#3	175	325	145	135	8	0	0	0	5	2	7
U1000d2#1	300	700	306	323	259	261	244	248	256	267	200
U1000d2#2	409	591	35	21	28	42	9	13	15	17	10
U1000d2#3	423	577	151	142	1	1	0	0	1	1	30

Table 5: Wrongly Assigned points for k-means, MCLUST, MVE, and Sample MVE on Uniform Data with $d = 2$.

Problem	**k-means**	**MCLUST**	**MVE**H1	**MVE**H2	**MVE**H3	**Sample MVE**
U100d2#1	20.23	0.09	12.42	7.24	7.40	1.43
U100d2#2	18.62	0.08	20.42	17.99	18.65	1.26
U100d2#3	28.24	0.09	20.51	18.23	16.10	0.87
U500d2#1	241.35	0.63	70.44	45.43	43.50	23.36
U500d2#2	232.68	0.60	120.24	90.15	86.03	18.71
U500d2#3	254.76	0.69	179.23	100.02	90.91	24.93
U1000d2#1	405.30	2.53	589.53	521.05	505.36	237.00
U1000d2#2	517.35	2.54	203.06	123.51	105.06	146.19
U1000d2#3	564.26	2.53	287.02	257.64	256.35	126.76

Table 6: Running Times for k-means, MCLUST, MVE, and Sample MVE on Uniform Data with $d = 2$.

Problem	n_1	n_2	**k-means**		**MCLUST**		**MVE**		**Sample MVE**		**True**
			Train	Test	Train	Test	Train	Test	Train	Test	Test
U500d5#1	238	262	30	39	0	0	0	0	0*	0	0
U500d5#2	184	316	0	0	0	0	0	0	0*	0	0
U500d5#3	217	283	170	166	0	0	0	0	0*	0	0
U1000d5#1	266	734	205	195	8	2	7	12	7*	12	2
U1000d5#2	537	463	30	27	0	0	0	0	0*	0	0
U1000d5#3	413	587	53	55	0	0	0	0	0*	0	0
U2000d5#1	441	1559	328	290	0	0	1	0	1*	0	0
U2000d5#2	1265	735	25	26	0	0	0	0	0*	0	0
U2000d5#3	1072	928	167	158	0	0	0	0	0*	0	0

Table 7: Wrongly Assigned points for k-means, MCLUST, MVE, and Sample MVE on Uniform Data with $d = 5$.

Problem	**k-means**	**MCLUST**	**MVE**H3	**Sample MVE**
U500d5#1	150.54	0.80	7200+	1041.92
U500d5#2	124.41	0.67	216.94	113.77
U500d5#3	304.97	0.82	638.19	199.59
U1000d5#1	374.46	3.46	7200+	725.32
U1000d5#2	415.27	3.15	1768.85	344.08
U1000d5#3	400.76	3.18	1146.30	428.72
U2000d5#1	1671.37	15.77	3444.22	4383.29
U2000d5#2	1417.49	14.95	955.20	1763.69
U2000d5#3	2440.71	15.26	4407.05	3397.38

Table 8: Running Times for k-means, MCLUST, MVE, and Sample MVE on Uniform Data with $d = 5$ and a time limit of 7200 CPU seconds .

Problem	n_1	n_2	**k-means**		**MCLUST**		**MVE**		**Sample MVE**		**True**
			Train	Test	Train	Test	Train	Test	Train	Test	Test
A100d2#1	56	44	0	0	1	0	0	0	43	44	2
A100d2#2	57	43	0	1	10	9	0	1	6	9	3
A100d2#3	32	68	16	11	5	7	0	0	1	0	0
A500d2#1	352	148	6	1	2	13	0	0	147	156	1
A500d2#2	247	253	8	16	35	22	2	12	2	12	8
A500d2#3	245	255	3	5	0	1	0	0	0*	0	0
A1000d2#1	418	582	97	80	59	42	6	7	14	2	65
A1000d2#2	394	606	93	85	2	4	2	2	1	2	5
A1000d2#3	565	435	37	25	13	11	3	7	4	7	41

Table 9: Wrongly Assigned points for k-means, MCLUST, MVE, and Sample MVE on Asymmetric Data with $d = 2$.

Problem	**k-means**	**MCLUST**	**MVE**H1	**MVE**H2	**MVE**H3	**Sample MVE**
A100d2#1	14.35	0.07	20.51	16.34	15.74	1.36
A100d2#2	24.03	0.19	20.32	19.02	19.86	1.26
A100d2#3	17.41	0.08	10.24	6.82	6.73	0.99
A500d2#1	184.30	0.60	80.35	60.25	52.54	36.63
A500d2#2	127.35	0.60	79.02	59.30	57.16	23.83
A500d2#3	92.26	0.55	60.35	35.33	29.99	31.16
A1000d2#1	490.21	2.38	125.23	93.52	90.63	134.20
A1000d2#2	432.62	2.89	143.63	40.35	36.65	129.34
A1000d2#3	352.22	2.66	241.53	80.36	78.09	142.86

Table 10: Running Times for k-means, MCLUST, MVE, and Sample MVE on Asymmetric Data with $d = 2$.

Problem	n_1	n_2	**k-means**		**MCLUST**		**MVE**		**Sample MVE**		**True**
			Train	Test	Train	Test	Train	Test	Train	Test	Test
A500d5#1	390	110	4	5	0	0	0	0	0*	0	0
A500d5#2	234	266	3	0	0	0	0	0	0*	0	0
A500d5#3	312	188	0	0	0	0	0	0	0*	0	0
A1000d5#1	349	651	15	12	0	0	0	0	0*	0	0
A1000d5#2	530	470	1	0	0	0	0	0	0*	0	0
A1000d5#3	499	501	2	1	0	0	0	0	0*	0	0
A2000d5#1	662	1338	14	96	0	0	0	0	0*	0	0
A2000d5#2	657	1343	2	2	0	0	0	0	0*	0	0
A2000d5#3	1386	614	14	7	0	0	0	0	0*	0	0

Table 11: Wrongly Assigned points for k-means, MCLUST, MVE, and Sample MVE on Asymmetric Data with $d = 5$.

Problem	**k-means**	**MCLUST**	**MVE**H3	**Sample MVE**
A500d5#1	157.40	0.79	162.83	115.31
A500d5#2	173.27	0.67	678.18	467.40
A500d5#3	124.41	0.76	442.89	119.93
A1000d5#1	487.98	3.25	7200+	1377.69
A1000d5#2	298.23	3.07	1493.73	618.51
A1000d5#3	293.14	3.11	3039.47	639.40
A2000d5#1	1053.22	15.20	3340.94	6796.64
A2000d5#2	989.03	15.08	1821.74	1931.05
A2000d5#3	1429.52	15.31	7200+	7200+

Table 12: Running Times for k-means, MCLUST, MVE, and Sample MVE on Asymmetric Data with $d = 5$ and a time limit of 7200 CPU seconds.

Problem	n_1	n_2	**k-means** Train	Test	**MVE** Train	Test	**Sample MVE** Train	Test	**True** Test
G100d2E#1	50	50	10	5	3	2	2	6	2
G100d2E#2	50	50	14	12	14	5	4	4	3
G100d2E#3	50	50	20	21	0	0	0*	0	0
G100d2U#1	20	80	12	13	0	0	0*	0	0
G100d2U#2	20	80	3	0	1	0	0	0	0
G100d2U#3	20	80	33	26	1	1	15	8	1
G100d2M#1	50	50	9	2	0	2	8	2	2
G100d2M#2	50	50	13	12	5	11	43	50	12
G100d2M#3	50	50	11	9	12	7	4	2	7
G500d2E#1	250	250	40	33	3	12	2	11	11
G500d2E#2	250	250	70	55	0	0	0*	0	0
G500d2E#3	250	250	91	90	43	52	59	67	44
G500d2U#1	350	150	183	144	3	3	3	4	3
G500d2U#2	350	150	78	102	9	10	13	11	14
G500d2U#3	350	150	135	111	21	3	21*	3	5
G500d2M#1	250	250	121	115	132	135	109	129	60
G500d2M#2	250	250	69	61	33	18	9	15	19
G500d2M#3	250	250	94	91	0	2	0*	2	2
G1000d2E#1	500	500	47	48	33	40	35	46	42
G1000d2E#2	500	500	190	168	66	45	66*	45	32
G1000d2E#3	500	500	125	111	21	31	21*	31	29
G1000d2U#1	800	200	312	283	47	92	44	92	91
G1000d2U#2	800	200	213	152	13	15	13*	15	17
G1000d2U#3	800	200	142	177	35	33	23	29	28
G1000d2M#1	500	500	148	151	0	12	9	30	12
G1000d2M#2	500	500	265	262	0	21	27	25	21
G1000d2M#3	500	500	206	180	56	55	76	79	50

Table 13: Wrongly Assigned points for k-means, MVE, and Sample MVE with Prior Central Information on Gaussian Data with $d = 2$.

Problem	**k-means**	**MVE**H1	**MVE**H2	**MVE**H3	**Sample MVE**
G100d2E#1	28.79	13.98	7.95	8.03	0.63
G100d2E#2	18.75	4.63	2.63	2.06	0.71
G100d2E#3	30.32	8.95	7.55	2.47	1.84
G100d2U#1	24.02	5.03	4.53	1.49	1.03
G100d2U#2	49.45	5.25	2.98	2.54	1.12
G100d2U#3	28.03	11.23	8.01	8.92	0.92
G100d2M#1	19.38	12.51	7.35	7.08	1.55
G100d2M#2	23.00	13.54	8.21	7.66	1.02
G100d2M#3	31.09	6.43	4.02	3.43	1.11
G500d2E#1	174.59	33.25	26.34	25.15	19.42
G500d2E#2	177.88	53.42	39.25	17.58	17.35
G500d2E#3	182.25	23.11	16.83	14.07	18.04
G500d2U#1	243.51	58.35	52.53	17.41	18.60
G500d2U#2	250.58	72.42	35.44	10.09	19.29
G500d2U#3	148.26	23.52	14.99	11.82	18.30
G500d2M#1	197.63	40.23	28.52	25.03	22.74
G500d2M#2	206.53	34.66	12.53	11.95	18.10
G500d2M#3	203.73	18.60	9.21	8.00	17.30
G1000d2E#1	594.47	122.51	90.24	76.67	126.70
G1000d2E#2	599.27	110.24	105.23	63.45	117.05
G1000d2E#3	635.75	88.20	47.24	36.81	108.19
G1000d2U#1	467.84	80.23	74.05	63.17	145.12
G1000d2U#2	686.26	126.43	81.24	51.18	167.08
G1000d2U#3	1156.50	73.00	72.31	59.54	140.58
G1000d2M#1	598.03	123.51	90.42	88.48	126.46
G1000d2M#2	647.20	172.09	151.25	127.15	182.30
G1000d2M#3	613.06	142.12	81.15	80.86	128.91

Table 14: Running Times for k-means, MVE, and Sample MVE with Prior Central Information on Gaussian Data with $d = 2$.

Problem	n_1	n_2	k-means		MVE		Sample MVE		True
			Train	Test	Train	Test	Train	Test	Test
G500d5E#1	250	250	3	5	0	0	0	0	0
G500d5E#2	250	250	0	0	0	0	0*	0	0
G500d5E#3	250	250	1	0	0	0	0	0	0
G500d5U#1	350	150	12	14	0	0	0*	0	0
G500d5U#2	350	150	75	40	0	0	0*	0	0
G500d5U#3	350	150	19	28	9	5	9*	5	1
G500d5M#1	250	250	52	58	0	0	0*	0	0
G500d5M#2	250	250	13	12	0	0	2	0	0
G500d5M#3	250	250	12	13	18	18	18*	18	14
G1000d5E#1	500	500	0	0	0	0	0*	0	0
G1000d5E#2	500	500	0	0	0	0	0*	0	0
G1000d5E#3	500	500	56	59	32	12	27	12	11
G1000d5U#1	800	200	23	31	0	2	0*	2	2
G1000d5U#2	800	200	68	100	22	32	21	32	37
G1000d5U#3	800	200	30	16	5	3	5*	3	1
G1000d5M#1	500	500	63	59	1	0	1*	0	0
G1000d5M#2	500	500	87	94	9	7	9*	7	1
G1000d5M#3	500	500	0	0	0	0	0*	0	0
G2000d5E#1	1000	1000	79	111	9	9	9*	9	8
G2000d5E#2	1000	1000	18	36	0	0	0*	0	0
G2000d5E#3	1000	1000	193	314	4	8	4*	8	8
G2000d5U#1	600	1400	18	26	0	0	0*	0	0
G2000d5U#2	600	1400	63	75	17	13	17*	13	14
G2000d5U#3	600	1400	34	45	0	0	0*	0	0
G2000d5M#1	1000	1000	32	64	0	0	0*	0	0
G2000d5M#2	1000	1000	371	345	0	0	2	1	0
G2000d5M#3	1000	1000	212	202	27	48	27*	48	43

Table 15: Wrongly Assigned points for k-means, MVE, and Sample MVE with Prior Central Information on Gaussian Data with $d = 5$.

Problem	**k-means**	**MVE**H3	**Sample MVE**
G500d5E#1	119.21	540.91	229.51
G500d5E#2	96.05	49.96	461.39
G500d5E#3	108.43	616.88	169.07
G500d5U#1	196.16	590.19	262.78
G500d5U#2	387.67	56.02	233.65
G500d5U#3	231.31	7200+	428.63
G500d5M#1	195.64	138.17	128.90
G500d5M#2	182.01	818.09	254.36
G500d5M#3	174.23	7200+	414.44
G1000d5E#1	244.05	2896.82	420.86
G1000d5E#2	278.22	926.84	1136.11
G1000d5E#3	602.06	7196.06	907.48
G1000d5U#1	1027.34	1786.53	488.54
G1000d5U#2	1388.24	422.51	525.43
G1000d5U#3	989.76	1996.61	676.19
G1000d5M#1	648.88	704.71	502.33
G1000d5M#2	567.37	7200+	1031.73
G1000d5M#3	284.05	209.45	877.61
G2000d5E#1	1687.66	7200+	1719.37
G2000d5E#2	1700.57	4247.95	4022.41
G2000d5E#3	2857.37	1540.36	4002.55
G2000d5U#1	1771.05	710.09	4319.08
G2000d5U#2	2056.74	7200+	2690.88
G2000d5U#3	1747.27	6510.16	5263.69
G2000d5M#1	1289.46	7200+	1893.24
G2000d5M#2	1925.27	5242.83	4911.93
G2000d5M#3	2408.14	7200+	2288.71

Table 16: Running Times for k-means, MVE, and Sample MVE with Prior Central Information on Gaussian Data with $d = 5$ and a time limit of 7200 CPU seconds .

Problem	n_1	n_2	**k-means** Train	Test	**MVE** Train	Test	**Sample MVE** Train	Test	**True** Test
G100d2E#1	50	50	12	6	0	3	2	6	2
G100d2E#2	50	50	14	12	4	6	4*	4	3
G100d2E#3	50	50	16	9	0	0	0*	0	0
G100d2U#1	20	80	6	11	0	0	0*	0	0
G100d2U#2	20	80	4	5	1	0	0	0	0
G100d2U#3	20	80	28	22	1	1	9	3	1
G100d2M#1	50	50	9	2	9	2	9*	2	2
G100d2M#2	50	50	13	12	46	47	46*	49	12
G100d2M#3	50	50	8	7	4	2	4*	2	7
G500d2E#1	250	250	62	35	3	12	2	11	11
G500d2E#2	250	250	71	60	0	0	0*	0	0
G500d2E#3	250	250	94	92	77	52	62	69	44
G500d2U#1	350	150	183	81	4	3	3	4	3
G500d2U#2	350	150	79	102	5	10	13	11	14
G500d2U#3	350	150	141	119	17	3	17*	3	5
G500d2M#1	250	250	121	113	109	129	109*	129	60
G500d2M#2	250	250	69	61	34	18	9	15	19
G500d2M#3	250	250	94	91	0	2	0*	2	2
G1000d2E#1	500	500	47	48	27	40	35	46	42
G1000d2E#2	500	500	194	168	66	45	66*	45	32
G1000d2E#3	500	500	123	109	20	31	20*	31	29
G1000d2U#1	800	200	316	291	41	91	41*	92	91
G1000d2U#2	800	200	220	167	13	15	13*	15	17
G1000d2U#3	800	200	143	177	23	33	23	29	28
G1000d2M#1	500	500	149	151	0	12	9	30	12
G1000d2M#2	500	500	265	226	0	21	55	42	21
G1000d2M#3	500	500	206	180	45	55	76	79	50

Table 17: Wrongly assigned points for k-means, MVE, and Sample MVE with Prior Extreme Information on Gaussian Data with $d = 2$.

Problem	**k-means**	**MVEH1**	**MVEH2**	**MVEH3**	**Sample MVE**
G100d2E#1	30.21	11.42	7.07	6.12	1.01
G100d2E#2	17.62	4.22	2.04	1.73	0.67
G100d2E#3	26.52	4.92	5.91	0.95	0.88
G100d2U#1	41.73	8.91	8.24	4.09	1.47
G100d2U#2	42.63	8.51	4.92	4.11	1.09
G100d2U#3	36.04	9.42	7.25	3.92	0.92
G100d2M#1	25.42	12.12	8.41	3.65	1.59
G100d2M#2	21.36	6.12	5.23	4.61	1.01
G100d2M#3	27.75	5.24	2.31	1.47	0.63
G500d2E#1	14.78	30.14	23.51	22.95	17.18
G500d2E#2	243.04	100.42	44.25	11.82	16.42
G500d2E#3	243.39	16.31	12.01	11.89	18.86
G500d2U#1	281.07	107.99	30.51	9.70	23.98
G500d2U#2	222.42	85.93	33.62	14.46	29.60
G500d2U#3	225.00	185.34	65.35	11.19	23.21
G500d2M#1	203.34	95.35	32.51	16.44	23.90
G500d2M#2	226.62	87.02	42.52	12.32	22.02
G500d2M#3	308.20	30.25	10.25	9.14	22.21
G1000d2E#1	714.34	169.35	120.52	111.65	137.95
G1000d2E#2	742.61	120.24	105.23	85.17	122.38
G1000d2E#3	469.10	315.23	172.24	73.76	119.74
G1000d2U#1	450.66	72.35	40.24	47.47	144.50
G1000d2U#2	689.35	320.41	162.34	74.04	173.82
G1000d2U#3	855.29	50.36	47.34	42.83	138.57
G1000d2M#1	740.11	185.23	139.35	130.37	134.94
G1000d2M#2	665.15	223.41	200.14	183.53	190.41
G1000d2M#3	582.02	231.53	160.42	156.00	143.37

Table 18: Running Times for k-means, MVE, and Sample MVE with Prior Extreme Information on Gaussian Data with $d = 2$.

Problem	n_1	n_2	k-means		MVE		Sample MVE		True
			Train	Test	Train	Test	Train	Test	Test
G500d5E#1	250	250	3	6	0	0	0*	0	0
G500d5E#2	250	250	0	0	0	0	0*	0	0
G500d5E#3	250	250	1	0	0	0	0*	0	0
G500d5U#1	350	150	12	11	0	0	0*	0	0
G500d5U#2	350	150	80	35	0	0	0*	0	0
G500d5U#3	350	150	19	23	6	2	7	3	1
G500d5M#1	250	250	51	53	0	0	0*	0	0
G500d5M#2	250	250	13	10	0	0	2	0	0
G500d5M#3	250	250	11	9	17	14	23	23	14
G1000d5E#1	500	500	0	0	0	0	0*	0	0
G1000d5E#2	500	500	0	0	0	0	0*	0	0
G1000d5E#3	500	500	58	57	18	12	18*	12	11
G1000d5U#1	800	200	23	30	0	3	0*	2	2
G1000d5U#2	800	200	281	219	19	34	19*	32	37
G1000d5U#3	800	200	29	13	5	2	5*	3	1
G1000d5M#1	500	500	63	57	3	0	3*	0	0
G1000d5M#2	500	500	89	97	0	1	0*	1	1
G1000d5M#3	500	500	0	0	0	0	0*	0	0
G2000d5E#1	1000	1000	80	108	9	9	9*	9	8
G2000d5E#2	1000	1000	19	42	0	0	0*	0	0
G2000d5E#3	1000	1000	185	265	4	8	4*	8	8
G2000d5U#1	600	1400	18	38	0	0	0*	0	0
G2000d5U#2	600	1400	63	90	76	17	20	13	14
G2000d5U#3	600	1400	34	45	0	0	0*	0	0
G2000d5M#1	1000	1000	32	63	0	0	0*	0	0
G2000d5M#2	1000	1000	371	343	0	0	0*	0	0
G2000d5M#3	1000	1000	212	198	41	50	41*	51	43

Table 19: Wrongly Assigned points for k-means, MVE, and Sample MVE with Prior Extreme Information on Gaussian Data with $d = 5$.

Problem	**k-means**	**MVE**H3	**Sample MVE**
G500d5E#1	1.17	127.14	189.89
G500d5E#2	0.97	34.64	187.41
G500d5E#3	1.31	43.41	70.44
G500d5U#1	1.98	15.77	83.71
G500d5U#2	2.83	23.27	174.25
G500d5U#3	2.80	1736.35	206.78
G500d5M#1	1.97	74.96	104.82
G500d5M#2	1.84	534.16	119.06
G500d5M#3	1.50	2601.89	117.66
G1000d5E#1	2.70	528.38	672.17
G1000d5E#2	2.62	66.57	243.53
G1000d5E#3	6.10	350.00	310.12
G1000d5U#1	9.46	65.70	230.82
G1000d5U#2	17.41	103.65	364.83
G1000d5U#3	15.44	119.96	447.70
G1000d5M#1	5.75	121.96	311.45
G1000d5M#2	4.84	7200+	854.41
G1000d5M#3	2.92	91.00	403.12
G2000d5E#1	16.51	1902.89	1848.15
G2000d5E#2	18.66	2621.15	1807.88
G2000d5E#3	24.74	505.32	2077.00
G2000d5U#1	17.25	482.20	2186.56
G2000d5U#2	19.31	919.70	1638.89
G2000d5U#3	17.67	1964.28	2592.33
G2000d5M#1	13.90	3736.61	1401.29
G2000d5M#2	15.87	1078.74	4935.82
G2000d5M#3	27.48	2269.00	1287.78

Table 20: Running Times for k-means, MVE, and Sample MVE with Prior Extreme Infromation on Gaussian Data with $d = 5$ and a time limit of 7200 CPU seconds .

Problem	n_1	n_2	**k-means** Train	Test	**MVE** Train	Test	**Sample MVE** Train	Test	**True** Test
U100d2#1	31	69	19	19	0	0	0*	0	3
U100d2#2	47	53	8	11	10	8	5	3	3
U100d2#3	45	55	12	7	0	0	1	1	0
U500d2#1	287	213	77	74	0	0	3	1	8
U500d2#2	174	326	150	166	55	41	40	40	48
U500d2#3	175	325	143	135	0	0	5	2	7
U1000d2#1	300	700	306	323	167	203	211	227	200
U1000d2#2	409	591	34	21	8	13	15	17	10
U1000d2#3	423	577	150	142	0	0	1	1	30

Table 21: Wrongly Assigned points for k-means, MVE, and Sample MVE with Prior Central Information on Uniform Data with $d = 2$.

Problem	**k-means**	**MVE**H1	**MVE**H2	**MVE**H3	**Sample MVE**
U100d2#1	20.46	3.42	3.24	3.03	0.84
U100d2#2	28.24	6.03	4.62	4.11	0.74
U100d2#3	26.74	5.53	5.00	4.77	1.05
U500d2#1	223.55	103.40	56.30	53.93	21.88
U500d2#2	230.64	20.24	15.32	12.50	18.69
U500d2#3	230.58	94.53	70.34	64.87	21.66
U1000d2#1	429.64	213.42	163.63	142.32	184.66
U1000d2#2	558.04	450.52	230.23	212.18	132.65
U1000d2#3	571.96	252.35	120.53	107.36	127.93

Table 22: Running Times for k-means, MVE, and Sample MVE with Prior Central Information on Uniform Data with $d = 2$.

Problem	n_1	n_2	**k-means** Train	Test	**MVE** Train	Test	**Sample MVE** Train	Test	**True** Test
U100d2#1	31	69	20	19	0	0	0*	0	3
U100d2#2	47	53	10	15	9	6	6	5	3
U100d2#3	45	55	19	8	0	0	36	40	0
U500d2#1	287	213	77	74	0	0	0*	0	8
U500d2#2	174	326	150	166	59	41	42	40	48
U500d2#3	175	325	143	135	0	0	5	2	7
U1000d2#1	300	700	306	323	256	267	256*	267	200
U1000d2#2	409	591	29	21	8	13	8*	13	10
U1000d2#3	423	577	151	142	0	0	1	1	30

Table 23: Wrongly assigned points for k-means, MVE, and Sample MVE with Prior Extreme Information on Uniform Data with $d = 2$.

Problem	**k-means**	**MVE**H1	**MVE**H2	**MVE**H3	**Sample MVE**
U100d2#1	19.15	12.42	7.90	8.72	1.15
U100d2#2	26.20	12.41	9.02	8.96	1.01
U100d2#3	26.39	15.43	8.01	7.94	1.22
U500d2#1	159.76	70.52	50.25	45.82	22.76
U500d2#2	224.57	34.52	16.34	14.35	17.58
U500d2#3	257.62	142.08	70.51	64.53	25.92
U1000d2#1	414.45	300.25	232.51	211.87	239.43
U1000d2#2	702.26	210.12	124.13	98.92	133.27
U1000d2#3	636.74	260.53	260.25	228.30	121.52

Table 24: Running Times for k-means, MVE, and Sample MVE with Prior Extreme Information on Uniform Data with $d = 2$.

Problem	n_1	n_2	**k-means** Train	Test	**MVE** Train	Test	**Sample MVE** Train	Test	**True** Test
U500d5#1	238	262	28	39	0	0	0*	0	0
U500d5#2	184	316	0	0	0	0	0*	0	0
U500d5#3	217	283	160	168	0	0	0*	0	0
U1000d5#1	266	734	204	195	7	12	7*	12	2
U1000d5#2	537	463	29	27	0	0	0*	0	0
U1000d5#3	413	587	52	55	0	0	0*	0	0
U2000d5#1	441	1559	328	290	1	0	1*	0	0
U2000d5#2	1265	735	24	26	0	0	0*	0	0
U2000d5#3	1072	928	167	158	0	0	0*	0	0

Table 25: Wrongly Assigned points for k-means, MVE, and Sample MVE with Prior Central Information on Uniform Data with $d = 5$.

Problem	**k-means**	**MVE**H3	**Sample MVE**
U500d5#1	164.60	2240.16	1363.19
U500d5#2	130.07	2106.00	352.69
U500d5#3	239.84	847.78	221.75
U1000d5#1	398.98	2546.12	1153.35
U1000d5#2	457.43	7200+	1003.97
U1000d5#3	434.76	1688.09	1171.74
U2000d5#1	1476.44	7200+	3844.33
U2000d5#2	1465.73	7200+	5629.75
U2000d5#3	2482.04	3139.16	2264.27

Table 26: Running Times for k-means, MVE, and Sample MVE with Prior Central Information on Uniform Data with $d = 5$ and a time limit of 7200 CPU seconds.

Problem	n_1	n_2	**k-means** Train	Test	**MVE** Train	Test	**Sample MVE** Train	Test	**True** Test
U500d5#1	238	262	25	36	0	0	0*	0	0
U500d5#2	184	316	0	0	0	0	0*	0	0
U500d5#3	217	283	161	159	0	0	0*	0	0
U1000d5#1	266	734	202	115	8	10	8*	12	2
U1000d5#2	537	463	26	19	0	0	0*	0	0
U1000d5#3	413	587	52	52	0	0	0*	0	0
U2000d5#1	441	1559	327	283	0	0	0*	0	0
U2000d5#2	1265	735	25	24	0	0	0*	0	0
U2000d5#3	1072	928	167	165	2	0	2*	0	0

Table 27: Wrongly assigned points for k-means, MVE, and Sample MVE with Prior Extreme Information on Uniform Data with $d = 5$.

Problem	**k-means**	**MVE**H3	**Sample MVE**
U500d5#1	154.37	187.65	118.55
U500d5#2	115.76	15.29	46.42
U500d5#3	326.04	12.91	53.48
U1000d5#1	453.00	82.48	257.41
U1000d5#2	535.66	147.33	325.88
U1000d5#3	476.98	106.38	481.87
U2000d5#1	1656.37	619.75	2511.85
U2000d5#2	1425.49	422.39	1995.81
U2000d5#3	2823.05	509.16	2033.69

Table 28: Running Times for k-means, MVE, and Sample MVE with Prior Extreme Information on Uniform Data with $d = 5$ and a time limit of 7200 CPU seconds.

Problem	n_1	n_2	**k-means** Train	Test	**MVE** Train	Test	**Sample MVE** Train	Test	**True** Test
A100d2#1	56	44	0	0	0	0	6	17	2
A100d2#2	57	43	1	0	0	1	7	12	3
A100d2#3	32	68	15	11	0	0	1	0	0
A500d2#1	352	148	6	1	0	0	1	2	1
A500d2#2	247	253	7	14	2	12	2	12	8
A500d2#3	245	255	3	5	0	0	0*	0	0
A1000d2#1	418	582	97	80	6	7	14	2	65
A1000d2#2	394	606	91	80	2	2	1	2	5
A1000d2#3	565	435	37	25	3	7	4	7	41

Table 29: Wrongly Assigned points for k-means, MVE, and Sample MVE with Prior Central Information on Asymmetric Data with $d = 2$.

Problem	**k-means**	**MVE**H1	**MVE**H2	**MVE**H3	**Sample MVE**
A100d2#1	13.66	7.34	5.93	5.34	0.93
A100d2#2	16.23	11.24	9.02	9.09	0.81
A100d2#3	20.04	10.53	9.15	9.03	1.23
A500d2#1	181.35	22.25	18.30	16.55	21.69
A500d2#2	123.42	50.25	36.02	30.07	22.72
A500d2#3	85.61	62.63	30.64	27.78	25.68
A1000d2#1	404.03	129.09	93.64	80.87	138.51
A1000d2#2	560.59	73.53	50.26	43.98	122.70
A1000d2#3	345.72	120.42	65.30	36.71	135.79

Table 30: Running Times for k-means, MVE, and Sample MVE with Prior Central Information on Asymmetric Data with $d = 2$.

Problem	n_1	n_2	**k-means** Train	Test	**MVE** Train	Test	**Sample MVE** Train	Test	**True** Test
A100d2#1	56	44	0	0	1	3	4	7	2
A100d2#2	57	43	13	9	0	0	7	12	3
A100d2#3	32	68	15	11	0	0	1	0	0
A500d2#1	352	148	9	2	0	0	2	6	1
A500d2#2	247	253	7	14	1	12	2	12	8
A500d2#3	245	255	3	6	0	0	0*	0	0
A1000d2#1	418	582	95	80	11	7	14	2	65
A1000d2#2	394	606	91	81	1	2	1*	2	5
A1000d2#3	565	435	42	28	4	7	7	8	41

Table 31: Wrongly assigned points for k-means, MVE, and Sample MVE with Prior Extreme Information on Asymmetric Data with $d = 2$.

Problem	**k-means**	**MVE**H1	**MVE**H2	**MVE**H3	**Sample MVE**
A100d2#1	14.77	2.31	1.25	1.13	0.85
A100d2#2	22.57	12.42	9.87	9.58	1.51
A100d2#3	16.32	4.32	2.98	2.69	1.06
A500d2#1	159.03	20.09	10.53	10.00	22.04
A500d2#2	131.62	31.24	20.01	19.07	23.77
A500d2#3	92.25	21.41	20.59	16.09	31.12
A1000d2#1	391.85	178.25	121.42	112.78	135.44
A1000d2#2	503.04	60.34	31.52	29.38	122.30
A1000d2#3	424.63	79.36	50.34	34.56	144.54

Table 32: Running Times for k-means, MVE, and Sample MVE with Prior Extreme Information on Asymmetric Data with $d = 2$.

Problem	n_1	n_2	**k-means** Train	Test	**MVE** Train	Test	**Sample MVE** Train	Test	**True** Test
A500d5#1	390	110	3	3	0	0	0*	0	0
A500d5#2	234	266	2	1	0	0	0*	0	0
A500d5#3	312	188	0	0	0	0	0*	0	0
A1000d5#1	349	651	15	12	0	0	0*	0	0
A1000d5#2	530	470	1	0	0	0	0*	0	0
A1000d5#3	499	501	2	1	0	0	0*	0	0
A2000d5#1	662	1338	13	95	0	0	0*	0	0
A2000d5#2	657	1343	2	2	0	0	0*	0	0
A2000d5#3	1386	614	13	7	0	0	0*	0	0

Table 33: Wrongly Assigned points for k-means, MVE, and Sample MVE with Prior Central Information on Asymmetric Data with $d = 5$.

Problem	**k-means**	**MVE**H3	**Sample MVE**
A500d5#1	142.99	168.84	288.53
A500d5#2	140.10	1610.66	471.95
A500d5#3	109.02	1488.79	216.28
A1000d5#1	442.46	1979.69	1209.55
A1000d5#2	287.50	3280.37	417.90
A1000d5#3	297.42	1391.85	1313.92
A2000d5#1	1209.70	4560.19	2912.55
A2000d5#2	960.05	4635.59	3214.10
A2000d5#3	1255.17	7200+	6316.71

Table 34: Running Times for k-means, MVE, and Sample MVE with Prior Central Information on Asymmetric Data with $d = 5$ and a time limit of 7200 CPU seconds.

Problem	n_1	n_2	**k-means** Train	Test	**MVE** Train	Test	**Sample MVE** Train	Test	**True** Test
A500d5#1	390	110	3	0	0	0	0*	0	0
A500d5#2	234	266	2	1	0	0	0*	0	0
A500d5#3	312	188	0	0	0	0	0*	0	0
A1000d5#1	349	651	13	13	0	0	0*	0	0
A1000d5#2	530	470	1	0	0	0	0*	0	0
A1000d5#3	499	501	2	1	0	0	0*	0	0
A2000d5#1	662	1338	13	87	0	0	0*	0	0
A2000d5#2	657	1343	2	1	0	0	0*	0	0
A2000d5#3	1386	614	14	6	0	0	0*	0	0

Table 35: Wrongly assigned points for k-means, MVE, and Sample MVE with Prior Extreme Information on Asymmetric Data with $d = 5$.

Problem	**k-means**	**MVE**H3	**Sample MVE**
A500d5#1	150.00	11.90	46.44
A500d5#2	159.81	33.03	206.72
A500d5#3	111.64	10.34	40.59
A1000d5#1	429.34	71.91	633.97
A1000d5#2	291.54	59.23	252.04
A1000d5#3	292.67	140.37	418.20
A2000d5#1	1214.17	526.26	2102.47
A2000d5#2	945.15	412.32	1639.48
A2000d5#3	1249.35	5127.23	2664.12

Table 36: Running Times for k-means, MVE, and Sample MVE with Prior Extreme Information on Asymmetric Data with $d = 5$ and a time limit of 7200 CPU seconds.

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

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

[2] J. Burkardt. Random_data. http://www.csit.fsu.edu/~ burkardt/m_src/random_data/random_data.html.

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

[4] C. Fraley and A.E. Raftery. Mclust:software for model-based clustering, discriminant analysis and density estimation. Technical Report no.415R, Department of Statistics, University of Washington, 2002.

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

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

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

[8] M. Kumar and J. Orlin. Volume-based clustering. In Preparation, 2003.

[9] P. Kumar and E. A. Yıldırım. Approximate minimum volume enclosing ellipsoids using core sets. manuscript, May 2003.

[10] K. Löwner. über monotone matrixfunktionen. *Mat. Z.*, 38:177–216, 1934.

[11] The MathWorks Inc. *MATLAB 6.5 Reference Guide*, 2003.

[12] Y. Nesterov and A.S. Nemirovskii. *Interior-Point Polynomial Algorithms in Convex Programming.* SIAM, Philadelphia, PA, 1994.

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

[14] P. J. Rousseeuw and A. M. Leroy. *Robust Regression and Outlier Detection.* Wiley, NY, NY, 1987.

[15] P. Sun and R. M. Freund. Computation of minimum volume covering ellipsoids. *Operations Research*, 52:690–706, 2004.

[16] M.J. Symons. Clustering criteria and multivariate normal mixtures. *Biometrics*, 37:35–43, 1981.

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

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