Minimum Volume Enclosing Ellipsoid Problem

Instructor: Prof.Jiawang Nie
Author: Xiaoyan He

May 2023

Abstract

This paper delves into the fascinating, yet complex, realm of computational geometry, with a particular focus on the Minimum Volume Enclosing Ellipsoid (MVEE) problem. This problem concerns finding the smallest volume ellipsoid that encapsulates a given finite set of points within a multi-dimensional Euclidean space, a challenge with far-reaching implications across fields such as machine learning, pattern recognition, and optimization.

We commence with a comprehensive introduction to the MVEE problem, elucidating its mathematical intricacies and its significance in both theoretical and practical applications. Then we review several ways to solve the MVEE problem. We also discuss the advantages and disadvantages of each method, and compare their performance . Finally, we conclude with a brief discussion of the applications of the MVEE problem in many fields, including machine learning, pattern recognition, and optimization.

1 Introduction

The Minimum Volume Enclosing Ellipsoid (MVEE) problem is a fundamental problem in computational geometry. It concerns finding the smallest volume ellipsoid that encapsulates a given finite set of points within a multi-dimensional Euclidean space. This seemingly straightforward task is deceptively complex, with solutions that require traversing the depths of high-dimensional geometry and delving into the realms of advanced algorithms.

The MVEE problem has found a myriad of applications in diverse scientific fields. In machine learning, it has informed support vector machines and other classification algorithms. In pattern recognition, it aids in distinguishing and categorizing different data clusters. In optimization, the ellipsoid method, which uses concepts similar to the MVEE problem, serves as a powerful tool for linear programming.

This paper will delve into the intricacies of the MVEE problem, exploring its mathematical foundations and its applications in various fields. We will also review several methods to solve the MVEE problem.

2 Problem Statement

The MVEE problem concerns finding the smallest volume ellipsoid that encapsulates a given finite set of points within a multi-dimensional Euclidean space. More formally, given a set of n points

$\{x_1, x_2, \ldots, x_n\}$ in $\mathbb{R}^d$, an ellipsoid can be defined by a center $c \in \mathbb{R}^d$ and a positive definite matrix $A \in \mathbb{R}^{d \times d}$, as follows:

$$E(c, A) = \{x \in \mathbb{R}^d : \|(x - c)^T A^{-1}(x - c)\|_2 \leq 1\} \tag{1}$$

The MVEE problem is then to find c and A that minimize the volume of E subject to the constraint that all points x_i are contained within E. The volume of E is given by the following formula:

$$\text{Vol}(E) = \frac{\pi^{d/2}}{\Gamma(d/2 + 1)} \sqrt{\det(A)} \tag{2}$$

where Γ is the gamma function, d is the dimension of the space, and $\det(A)$ is the determinant of the inverse of A. Note that the volume of E is proportional to $\sqrt{\det(A)}$, so minimizing the volume of E is equivalent to minimizing $\det(A)$. Thus, the problem can be expressed as the following optimization problem:

$$\begin{aligned} \underset{c,A}{\text{minimize}} \quad & \det(A) \\ \text{subject to} \quad & \|(x_i - c)^T A^{-1}(x_i - c)\|_2 \leq 1, \quad i = 1, \ldots, n \\ & A \succ 0 \end{aligned} \tag{3}$$

However, in order to simplify the problem, we can indeed minimize the trace of A instead of the determinant of A, since trace is linear and determinant is not. Thus, the problem can be expressed as the following optimization problem:

$$\begin{aligned} \underset{c,A}{\text{minimize}} \quad & \text{tr}(A) \\ \text{subject to} \quad & \|(x_i - c)^T A^{-1}(x_i - c)\|_2 \leq 1, \quad i = 1, \ldots, n \\ & A \succ 0 \end{aligned} \tag{4}$$

3 Shur Complement

In order to solve the MVEE problem, we need to further formulate the problem into a linear conic optimization problem. To do so, we first need to introduce the Shur Complement Lemma, which is a useful tool in this problem.

Lemma 1 (Shur Complement Lemma) *Let M be a real symmetric matrix partitioned as follows:*

$$M = \begin{bmatrix} A & B \\ B^T & C \end{bmatrix} \tag{5}$$

where both A, C are symmetric square matrices. Then assuming C is positive definite following statements are equivalent:

1. $M \succeq 0$
2. $C - BA^{-1}B^T$, *is positive semidefinite.*

Proof 3.1 *Recall the fact that a symmetric matrix M is positive semidefinite if and only if for $\forall x$, $x^T Mx \geq 0$. Then partition the vector x into (y, z) and we have that $\forall (y, z)$*

$$g(y,z) = \begin{bmatrix} y^T & z^T \end{bmatrix} \begin{bmatrix} A & B \\ B^T & C \end{bmatrix} \begin{bmatrix} y \\ z \end{bmatrix} = y^T Ay + 2y^T Bz + z^T Cz \geq 0 \tag{6}$$

$g(y, z)$ is a quadratic function of y and z that is always nonnegative. Then fix z and we have the function $f(y)$ that is only about y and also bounded below. Then we can compute the global minimum of $f(y)$ by taking the gradient of $f(y)$ and set it to zero. Then we have

$$\nabla_y f(y) = 2Ay + 2Bz = 0 \Rightarrow y = -A^{-1}Bz \tag{7}$$

Plug this result back into $g(y, z)$ and we have

$$g(y,z) = z^T (C - B^T A^{-1} B)z \geq 0 \tag{8}$$

Since z is an arbitrary, we have $C - B^T A^{-1} B \succeq 0$.
Conversely, we can use the similar method that by the fact $g(y, z)$ always have a global minimum and set the gradient to zero and solve the equation. We can prove that $M \succeq 0$.

With this lemma, we can now formulate the MVEE problem into a linear conic optimization problem. By subsitution C as 1 and B as $(x_i - c)$ we have following:

$$\begin{bmatrix} A & (x_i - c) \\ (x_i - c)^T & 1 \end{bmatrix} \succeq 0 \iff 1 - (x_i - c)^T A^{-1} (x_i - c) \geq 0 \tag{9}$$

Then we can rewrite the MVEE problem in the form of linear conic programming as following:

$$\begin{aligned} \underset{c,A}{\text{maximize}} \quad & -\operatorname{tr}(A) \\ \text{subject to} \quad & \begin{bmatrix} A & (x_i - c) \\ (x_i - c)^T & 1 \end{bmatrix} \succeq 0, \quad i = 1, \ldots, n \\ & A \succ 0 \end{aligned} \tag{10}$$

This is the dual form of the MVEE problem, with this form we then can tranfrom it into code and solve it.

4 Solution Provided by SeDuMi

SeDuMi is a MATLAB toolbox for optimization over symmetric cones. It was developed by Jos F. Sturm and Stephen Boyd at Stanford University. SeDuMi is a free software and can be downloaded from `http://sedumi.ie.lehigh.edu/` .

In SeDuMi, the objective function must be expressed as a linear function as $b^T y$, where b is a vector and y is a vector of variables. Thus we need to convert the $\operatorname{tr}(A)$ to a linear function. Since the matrix A is symmetric, then we can convert it into a vector with $d(d+1)/2$ elements as following:

$$\begin{bmatrix} y_1 & y_2 & \cdots & y_d \\ y_2 & y_{d+1} & \cdots & y_{2d-1} \\ \vdots & \vdots & \ddots & \vdots \\ y_d & y_{2d-1} & \cdots & y_{d(d+1)/2} \end{bmatrix} \rightarrow \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_{d(d+1)/2} \end{bmatrix} \tag{11}$$

Since we also have a center c in the problem, we can also add it into the vector y as following:

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_{d(d+1)/2} \\ c_1 \\ c_2 \\ \vdots \\ c_d \end{bmatrix} \tag{12}$$

Then we can convert the problem into the form of SeDuMi as following:

$$\begin{aligned} \underset{y}{\text{maximize}} \quad & -b^T y \\ \text{subject to} \quad & \begin{bmatrix} y_1 & y_2 & \cdots & y_d \\ y_2 & y_{d+1} & \cdots & y_{2d-1} \\ \vdots & \vdots & \ddots & \vdots \\ y_d & y_{2d-1} & \cdots & y_{d(d+1)/2} \end{bmatrix} \succeq 0 \\ & \begin{bmatrix} y_1 & y_2 & \cdots & y_d & x_{i1}-c_1 \\ y_2 & y_{d+1} & \cdots & y_{2d-1} & x_{i2}-c_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ y_d & y_{2d-1} & \cdots & y_{d(d+1)/2} & x_{id}-c_d \\ x_{i1}-c_1 & x_{i2}-c_2 & \dots & x_{id}-c_d & 1 \end{bmatrix} \succeq 0, \quad i = 1, \dots, n \end{aligned} \tag{13}$$

Then we can use SeDuMi to solve this problem and get the optimal solution. The basic ideas are listed as following:

1. Randomly generate some points.
2. Set K.s for this problem.
3. Construct A, b, c for this problem then convert them into the form that can be understanded by SeDuMi.
4. Convert the optimal solution back to the form of ellipsoid, then plot the randomly generated points and the ellipsoid.

In this paper, I generate 20 points in 3-dimensional space and use SeDuMi to solve the problem. The result is shown in Figure 1.

This method's main advantage is that its objective function is convex and can be solved directly by any convex optimization solver. However, the disadvantage of this method is that it is not very accurate when the dimension of the space is high and inefficient when the number of points is large.

5 Solution Provided by Iterative Method

Although in last section we provide a method that can directly solve the MVEE problem, we indeed weaken the objective function by converting it into a linear function. To solve the problem

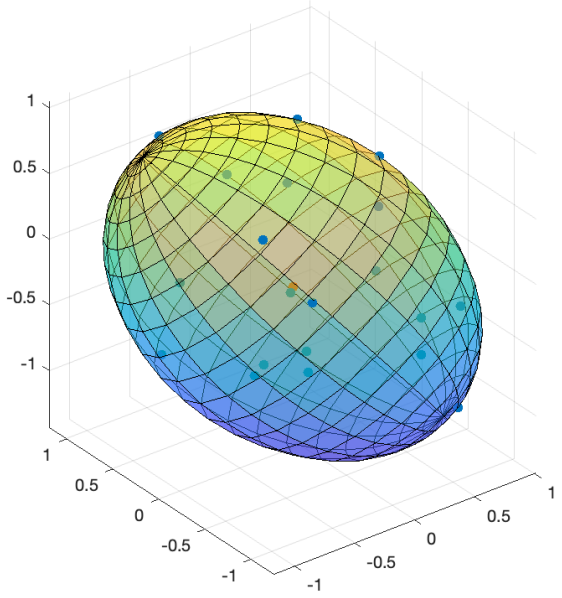


Figure 1: Result of SeDuMi

more precisely, we should try to solve the optimization problem with the original objective function. The problem of the original objective function is that a determinant of a matrix is not a convex function, thus it's hard for us to solve the problem by deterministic method. Thus, in this section, we focus on the iterative method. Unlike deterministic algorithms that aim to find an exact solution, iterative methods progressively refine an initial approximation of the MVEE based on a sequence of computations, aiming to converge to the optimal solution. One of the benefits of using iterative methods to solve the MVEE problem is that they can handle larger, higher-dimensional datasets more efficiently compared to other methods.

Recall that we can define the ellipsoid as $E(c, A) = \{x \in \mathbb{R}^d : (x - c)^T A(x - c) \leq 1\}$. By defining the ellipsoid in this way, the volume of the ellipsoid is $\frac{\pi^{d/2}}{\Gamma(d/2+1)}\sqrt{\det(A^{-1})}$ that leads to following prime problem:

$$\begin{aligned} \underset{c,A}{\text{minimize}} \quad & \sqrt{\det(A^{-1})} \\ \text{subject to} \quad & (x_i - c)^T A(x_i - c) \leq 1, \quad i = 1, \ldots, n \\ & A \succ 0 \end{aligned} \tag{14}$$

And by applying logarithm to the objective function (since logarithm is monotonic), we can get the following:

$$\begin{aligned} \underset{c,A}{\text{minimize}} \quad & -\log(\det(A)) \\ \text{subject to} \quad & (x_i - c)^T A(x_i - c) \leq 1, \quad i = 1, \ldots, n \\ & A \succ 0 \end{aligned} \tag{15}$$

The simplest iterative method is gradient descent method. Gradient descent is a first-order iterative

optimization algorithm for finding a local minimum of a differentiable function. To find a local minimum of a function using gradient descent, one takes steps proportional to the negative of the gradient (or approximate gradient) of the function at the current point.

The basic idea of gradient descent method is listed as following:

1. Initialize the center c and the matrix A, set the learning rate α and maximum iteration number.
2. Loop until the optimal solution is found or the maximum iteration number is reached.
3. For each iteration, calculate the gradient of the objective function with respect to c and A.
4. Update c and A with the gradient and learning rate.

However, it's notable to mention that the gradient descent method is not guaranteed to converge to the global minimum since the objective function is not convex as the figure 2 shown below.

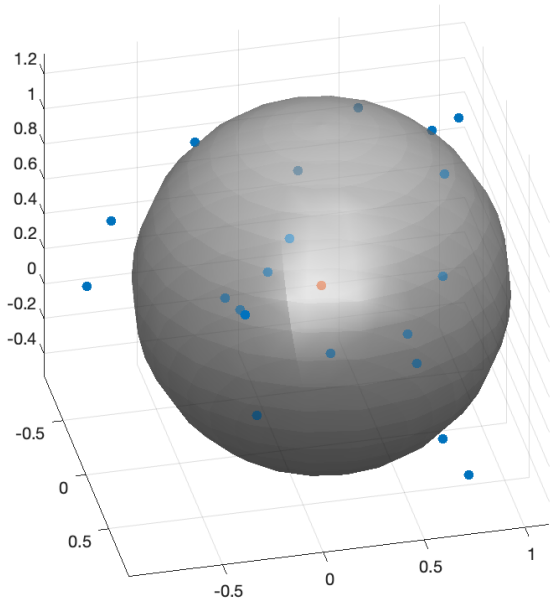


Figure 2: Result of Gradient Descent Method

Similar problem also exists in the Newton's method. To encounter such problem, a common iterative method is called Khachiyan algorithm. It's a polynomial-time algorithm for solving the MVEE problem. The basic idea is to start with an initial ellipsoid that covers all the points in the dataset and then iteratively update this ellipsoid to make it smaller while still enclosing all the points. The algorithm terminates when the change in the volume of the ellipsoid is smaller than a predefined tolerance. And consider a set of points $P = \{x_1, \ldots, x_n\} \subset \mathbb{R}^d$ whose affine hull is $\mathbb{R}^d$. If the matrix P is not centrally symmetric, we can add an additional row of ones to the bottom of P to lift it into $\mathbb{R}^{d+1}$, that is $P' = \{\pm x'_1, \ldots, \pm x'_n\}$ where $x'_i = (x_i, 1)^T$ that turns out P' is centrally symmetric. Denote the solution to the set P as $MVEE(P)$, then we have $MVEE(P) = MVEE(P') \cap \mathcal{H}$ where

$\mathcal{H} = \{x \in \mathbb{R}^{d+1} : x_{d+1} = 1\}$. Since we construct P' to be centrally symmetric, we can assume that $MVEE(P')$ is centered at the origin. Thus, we can raise the primal of optimization problem as following:

$$\begin{aligned} \underset{A}{\text{minimize}} \quad & -\log(\det(A)) \\ \text{subject to} \quad & x_i^T A x_i \leq 1, \quad i = 1, \ldots, n \\ & A \succ 0, \quad A \in \mathbb{R}^{d+1 \times d+1} \end{aligned} \tag{16}$$

And the dual problem is:

$$\begin{aligned} \underset{u}{\text{maximize}} \quad & \log det(V(u)) \\ \text{subject to} \quad & e^T u = 1 \\ & u \geq 0 \end{aligned} \tag{17}$$

where $u \in \mathbb{R}^n$ is the decision variable and $V(u) = \sum_{i=1}^{n} u_i x_i x_i^T$ is a positive definite matrix. Let u^* be the optimal solution to the dual problem, then let U^* be the diagonal matrix with diagonal entries u^*. Let matrix P be the $d \times n$ matrix whose columns are the points in P, then the optimal solution gives us the approximation of minimum volume ellipsoid as $E(c^*, A^*) = \{x \in \mathbb{R}^d : (x - c^*)^T A^* (x - c^*) \leq 1\}$ where $c^* = PU^*$ and $A^* = \frac{1}{d}(PU^*P^T - Pu^*(Pu^*)^T)^{-1}$. The algorithm is listed as following:

Algorithm 1 Minimum Volume Enclosing Ellipsoid using Khachiyan's Algorithm

Require: P, *tolerance* ▷ P is a $d \times N$ matrix where each column is a data point

Ensure: A, c

Initialize Q as P with an additional row of ones at the end

Initialize u as a uniform weight vector of size N (number of data points)

while Change in u is greater than *tolerance* **do**

Compute X as a matrix formed by the weighted sum of the outer product of each column of Q

Compute M as a vector where each entry is the diagonal entry of an $N \times N$ matrix obtained by $Q^T \times X^{-1} \times Q$

Find j, the index of the maximum entry of M

Compute *step_size* using the formula $(\max(M) - d - 1)/((d + 1) * (\max(M) - 1))$

Update u as a linear combination of the old u and a vector that has value *step_size* at the j-th entry and zero otherwise

Update the error as the norm of the difference between the old u and the updated u

end while

Compute U, a diagonal matrix formed by u

Compute A as the shape matrix of the minimum volume enclosing ellipsoid

Compute c as the center of the minimum volume enclosing ellipsoid

return A, c

The result of Khachiyan's algorithm is shown in figure 3. It's obvious that the result is much better than the result provide by gradient descent method.

One of the main advantages of Khachiyan's Algorithm is its efficiency. The algorithm has a polynomial time complexity, This makes it a favorable choice for large-scale MVEE problems where other algorithms may be prohibitively expensive. Moreover, the algorithm is guaranteed to converge

to an ellipsoid that is at most a factor of $(1+\epsilon)$ away from the optimal solution, where ϵ is a small user-defined error term. This gives users a balance between the speed and accuracy of the algorithm.

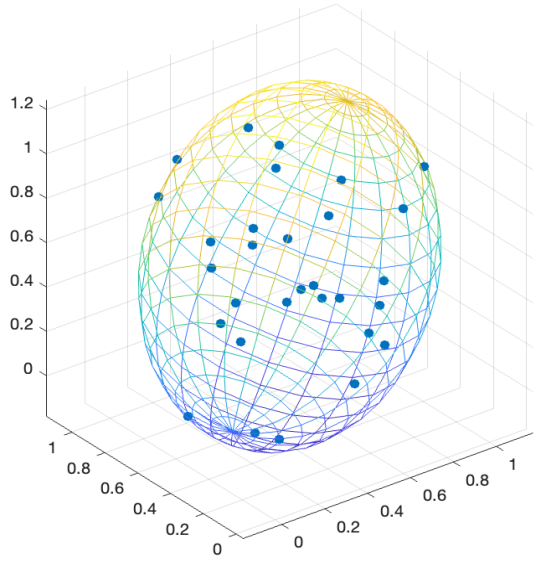


Figure 3: Result of Khachiyan's Algorithm

6 Comparison of the two algorithms

In the section above, we have introduced two different algorithms to solve the minimum volume enclosing ellipsoid problem. In this section, we will compare the two algorithms in terms of their efficiency when dealing with different size and dimension of data set.

In those two figures shown below, we first fix the dimension of the data set to be 3, and then we vary the number of data points up to 10000. For figure 4, we can see that for both algorithms, the time it takes to solve the problem increases as the number of data points increases. But the time it takes for Khachiyan's algorithm is less than the time it takes for the algorithm that uses SeDuMi. This is because Khachiyan's uses a tolerance to control the number of iterations. But it's worth noting that the time it takes for Khachiyan's algorithm will increase as the we set a smaller tolerance.

By figure 5, we can observe that for a smaller tolerance value, although initally the time it takes for Khachiyan's algorithm is less than the time it takes for the algorithm that uses SeDuMi, as the number of data points increases, the time it takes for Khachiyan's algorithm will be greater and increases faster than the time it takes for the algorithm that uses SeDuMi.

In figure 6, we fix the number of data points to be 1000, and then we vary the dimension of the dataset up to 10. We can observe that the increase of the dimensionality doesn't have a significant impact on the time it takes for Khachiyan's algorithm, while for the algorithm that uses SeDuMi, the

time it takes increases as the dimensionality increases.

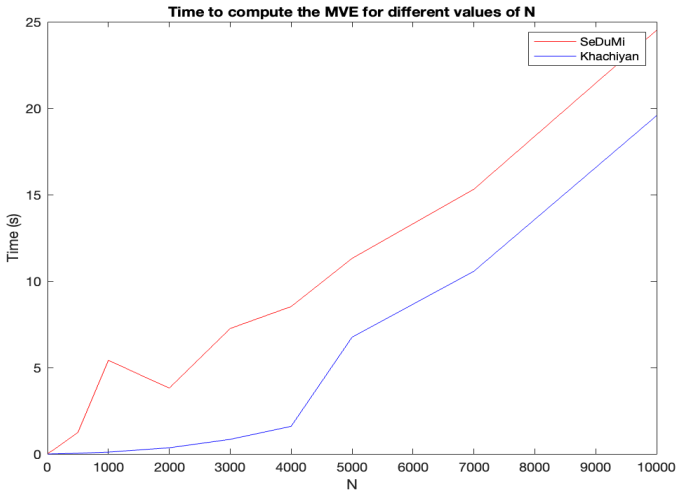


Figure 4: Set the tolerance of Khachiyan's Algorithm as 0.01

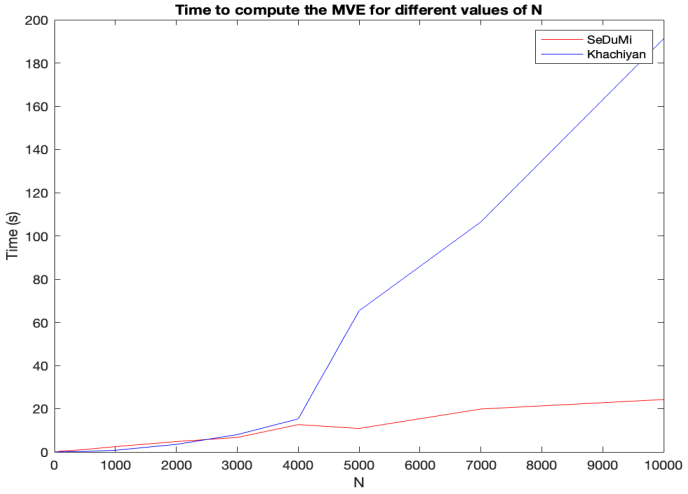
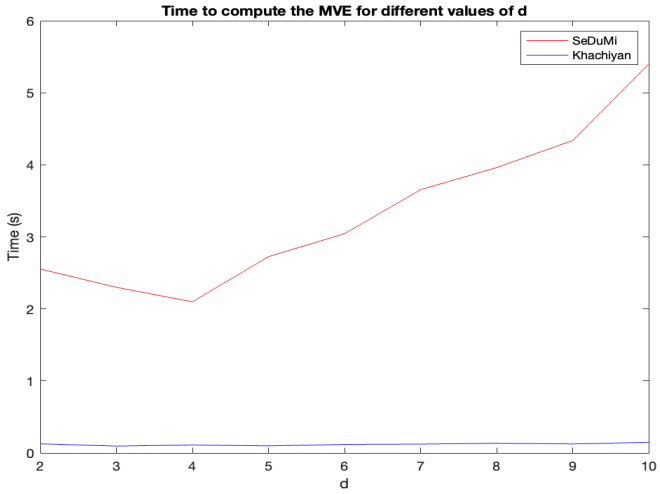


Figure 5: Set the tolerance of Khachiyan's Algorithm as 0.001



7 Applications of MVEE

The MVEE is particularly useful for detecting outliers in multivariate data. Outlier detection is a critical step in data preprocessing, particularly in machine learning, because outliers can significantly affect the performance of learning algorithms. Suppose we have a set of n observations (data points) in a d-dimensional space, represented as $P = p_1, p_2, ..., p_n$, where each p_i is a d-dimensional vector.

We aim to find the MVEE E that encloses all points in P or, equivalently, the smallest ellipsoid centered at $c \in \mathbb{R}^d$ with shape matrix $A \in \mathbb{R}^{d \times d}$ such that all p_i satisfy the inequality

$$(p_i - c)^T A^{-1} (p_i - c) \leq 1 \tag{18}$$

Once the MVEE is found, we can then use it to detect outliers in new observations. Let's say we have a new observation p_{new}. We can determine if p_{new} is an outlier by checking if it lies outside the MVEE. In other words, p_{new} is considered an outlier if

$$(p_{new} - c)^T A^{-1} (p_{new} - c) > 1 \tag{19}$$

If this inequality is satisfied, then p_{new} lies outside the MVEE and hence can be considered an outlier. The beauty of this approach is that it extends naturally to multivariate data and it is particularly useful in scenarios where we expect our normal data to be clustered around certain regions (captured by the MVEE), with outliers falling outside these regions.

Also MVEE can be applied in Support Vector Machine (SVM) classification. In the traditional SVM approach, the aim is to find a hyperplane that maximally separates two classes of data points. The hyperplane is chosen to maximize the margin, which is the distance from the hyperplane to the nearest data point from either class.

Let's denote our two classes of data points as $P_+ = p_1, p_2, ..., p_n$ and $P_- = p_{n+1}, p_{n+2}, ..., p_m$, where each p_i is a d-dimensional vector. The hyperplane is defined by a weight vector $w \in \mathbb{R}^d$ and bias term $b \in \mathbb{R}$, and a data point $x \in \mathbb{R}^d$ is classified according to the sign of the decision function $f(x) = w^T x + b$.

Now, consider the case where the data are not linearly separable, or where a nonlinear decision boundary may be more appropriate. One possible approach is to use MVEE to fit separate ellipsoids to the two classes of data points. This would result in two ellipsoids, E_+ and E_-, where E_+ is defined by center $c_+ \in \mathbb{R}^d$ and shape matrix $A_+ \in \mathbb{R}^{d\times d}$, and similarly E_- is defined by c_- and A_-. Each p_i in P_+ would satisfy $(p_i - c_+)^T A_+(p_i - c_+) \leq 1$, and similarly for P_-.

A new data point $x \in \mathbb{R}^d$ could then be classified according to which ellipsoid it is closer to. One possible measure of "closeness" is the Mahalanobis distance, which for a point x with respect to an ellipsoid with center c and shape matrix A is given by $\sqrt{(x-c)^T A(x-c)}$. So, we could classify x according to the sign of the function

$$g(x) = \sqrt{(x - c_+)^T A_+(x - c_+)} - \sqrt{(x - c_-)^T A_-(x - c_-)} \tag{20}$$

In other words, x is assigned to class P_+ if $g(x) > 0$, and to class P_- otherwise. This approach allows for more flexibility in the decision boundary, potentially improving classification performance when the assumption of a linear decision boundary is not appropriate. Note, however, that the problem of finding the MVEE is more computationally intensive than finding a separating hyperplane, and this approach may be less efficient or practical for large datasets.

Other than those mentioned above, MVEE can also be applied in other fields, such as finance. In portfolio optimization and risk management, MVEE can be used to understand the structure of multivariate asset returns and model the boundaries of portfolio loss.

8 Conclusion

In this paper, we have discussed the MVEE problem and its applications. We have also implemented two algorithms to solve the MVEE problem, one is transform the original objective function of MVEE problem into a linear function and directly solve it by convex optimization solver, the other is method is Khachiyan's algorithm that iteratively updates the center and shape matrix of the ellipsoid until the change of the objective function is less than a tolerance. We have also compared the performance of these two algorithms. Finally, we have discussed the applications of MVEE in outlier detection and SVM. The Khachiyan's algorithm is more efficient in general case, while we need to pay attention to the value of tolerance, because if the tolerance is too small, the algorithm is hard to converge and takes more time to run. On the other hand, if the tolerance is too large, the algorithm may converge too early and the precision of the result is not good enough. Thus we need to choose a proper tolerance to make the algorithm converge in a reasonable time and get a good result.

In general, MVEE is a very useful tool in many fields, and the ways to solve MVEE problem are not limited to the two algorithms we have discussed in this paper. There are many other algorithms that can be used to solve MVEE problem such as Todd's algorithm, which is an update version of Khachiyan's algorithm. In the future, we can try to implement other algorithms to solve MVEE problem and compare their performance. We can also try to apply MVEE in other fields that not limited to the fields we have discussed in this paper.

9 Reference

[1] Boyd, Stephen, and Lieven Vandenberghe. Convex Optimization - Stanford University, Accessed 8 June 2023.

[2] Todd, Michael J. Minimum-Volume Ellipsoids — Siam Digital Library, Accessed 8 June 2023.

[3] Jambawalikar, Sachin. A Note on Approximate Minimum Volume Enclosing Ellipsoid of Ellipsoids ..., ieeexplore.ieee.org/document/4561253/. Accessed 8 June 2023.

[4] Källberg, Linus. "Minimum Enclosing Balls and Ellipsoids in General Dimensions: Semantic Scholar." Minimum Enclosing Balls and Ellipsoids in General Dimensions — Semantic Scholar, 1 Jan. 1970, http://www.diva-portal.org/smash/get/diva2:1366471/FULLTEXT01.pdf. Accessed 8 June 2023.