A Branch and Bound Method Solving the Max–Min Linear Discriminant Analysis Problem

Amir Beck* Raz Sharon †

March 19, 2023

Abstract

Fisher linear discriminant analysis (FLDA or LDA) is a well-known technique for dimension reduction and classification. The method was first formulated in 1936 by Fisher in the one–dimensional setting. In this paper, we will examine the LDA problem using a different objective function. Instead of maximizing the sum of all distances between all classes, we will define an objective function that will maximize the minimum separation among all distances between all classes. This leads to a difficult nonconvex optimization problem. We present a branch and bound method for the problem in the case where the reduction is to a one–dimensional space.

1 Introduction

1.1 Fisher's Linear Discriminant Analysis

The starting point of this paper is Fisher's linear discriminant analysis model, which we now recall. Assume we have a dataset in $\mathbb{R}^d$ which contains n samples from c classes labeled as $1, 2, \ldots, c$. Each sample $\mathbf{x}_i \in \mathbb{R}^d,\ i \in \{1, \ldots, n\}$ is associated with one class. We wish to project the dataset into a lower–dimensional space $\mathbb{R}^p,\ p \leq d$, where the different classes can be easily separated in some sense. We denote by $C^{(k)}$ the set containing all the indices of samples associated with class k. The number of samples in each class is $n_k \equiv \left|C^{(k)}\right|$. We are looking for a matrix $\mathbf{P} \in \mathbb{R}^{d\times p}$ such that $\mathbf{P}^T\mathbf{P} = \mathbf{I}_p$. The new representation of the data set in $\mathbb{R}^p$ is $\mathbf{y}_i := \mathbf{P}^T\mathbf{x}_i$. We denote by $\boldsymbol{\mu}^{(k)}$ the center of each class, and by $\boldsymbol{\mu}$ the center of the entire dataset:

$$\boldsymbol{\mu}^{(k)} := \frac{1}{n_k}\sum_{i\in C^{(k)}} \mathbf{x}_i, \quad \boldsymbol{\mu} := \frac{1}{n}\sum_{i=1}^{n}\mathbf{x}_i.$$

Similarly, the class centers and the entire sample center in the lower–dimensional space $\mathbb{R}^p$ are denoted by:

$$\tilde{\boldsymbol{\mu}}^{(k)} := \frac{1}{n_k}\sum_{i\in C^{(k)}} \mathbf{y}_i = \mathbf{P}^T\boldsymbol{\mu}^{(k)}, \quad \tilde{\boldsymbol{\mu}} := \frac{1}{n}\sum_{i=1}^{n}\mathbf{y}_i = \mathbf{P}^T\boldsymbol{\mu} = \sum_{i=1}^{c}\frac{n_i}{n}\tilde{\boldsymbol{\mu}}^{(k)}.$$

*School of Mathematical Sciences, Tel Aviv University, becka@tauex.tau.ac.il. The research of Amir Beck is partially supported by ISF grant no. 926/21.

†School of Mathematical Sciences, Tel Aviv University, razsharon@mail.tau.ac.il

We denote two important matrices—the between scatter matrix $\mathbf{S}_B$, and the within–class scatter matrix $\mathbf{S}_W$:

$$\mathbf{S}_W := \sum_{k=1}^{c} \mathbf{S}_W^{(k)}, \quad \mathbf{S}_B := \sum_{k=1}^{c} n_k \mathbf{S}_B^{(k)},$$

where

$$\mathbf{S}_W^{(k)} := \sum_{i \in C_k} \left(\mathbf{x}_i - \boldsymbol{\mu}^{(k)}\right) \left(\mathbf{x}_i - \boldsymbol{\mu}^{(k)}\right)^T, \quad \mathbf{S}_B^{(k)} := \left(\boldsymbol{\mu}^{(k)} - \boldsymbol{\mu}\right) \left(\boldsymbol{\mu}^{(k)} - \boldsymbol{\mu}\right)^T.$$

We wish to find a transformation into a lower–dimensional space $\mathbb{R}^p$ which maximizes the separation between classes. To do so, we must first formulate a measure for how good is the separation. We use the common measure function R which is the ratio of the variance between the classes and the variance within the classes:

$$R(\mathbf{P}) = \frac{\phi_B(\mathbf{P})}{\phi_W(\mathbf{P})}, \tag{1.1}$$

where

$$\phi_B(\mathbf{P}) := \sum_{k=1}^{c} \frac{n_k}{n} \|\tilde{\boldsymbol{\mu}}^{(k)} - \tilde{\boldsymbol{\mu}}\|_2^2, \quad \phi_W(\mathbf{P}) := \frac{1}{n} \sum_{k=1}^{c} \sum_{i \in C^{(k)}} \left\|\mathbf{y}_i - \tilde{\boldsymbol{\mu}}^{(k)}\right\|_2^2.$$

We can rewrite $\phi_B(\mathbf{P})$ and $\phi_W(\mathbf{P})$ more explicitly as

$$\begin{aligned}
\phi_W(\mathbf{P}) &= \frac{1}{n} \sum_{k=1}^{c} \sum_{i \in C^{(k)}} \left\|\mathbf{y}_i - \tilde{\boldsymbol{\mu}}^{(k)}\right\|_2^2 = \frac{1}{n} \sum_{k=1}^{c} \sum_{i \in C^{(k)}} \left\|\mathbf{P}^T \left(\mathbf{x}_i - \boldsymbol{\mu}^{(k)}\right)\right\|_2^2 \\
&= \frac{1}{n} \sum_{k=1}^{c} \sum_{i \in C^{(k)}} \operatorname{Tr}\left(\mathbf{P}^T \left(\mathbf{x}_i - \boldsymbol{\mu}^{(k)}\right) \left(\mathbf{x}_i - \boldsymbol{\mu}^{(k)}\right)^T \mathbf{P}\right) \\
&= \frac{1}{n} \operatorname{Tr}(\mathbf{P}^T \mathbf{S}_W \mathbf{P}), \\
\phi_B(\mathbf{P}) &= \frac{1}{n} \sum_{k=1}^{c} n_k \|\mathbf{P}^T \boldsymbol{\mu}^{(k)} - \mathbf{P}^T \boldsymbol{\mu}\|_2^2 = \frac{1}{n} \operatorname{Tr}\left(\mathbf{P}^T \left(\sum_{k=1}^{c} n_k \left(\boldsymbol{\mu}^{(k)} - \boldsymbol{\mu}\right) \left(\boldsymbol{\mu}^{(k)} - \boldsymbol{\mu}\right)^T\right) \mathbf{P}\right) \\
&= \frac{1}{n} \operatorname{Tr}\left(\mathbf{P}^T \mathbf{S}_B \mathbf{P}\right).
\end{aligned}$$

The goal is to maximize the ratio of the above two expressions under an orthogonality constraint on $\mathbf{P}$. The arising problem is called the *Linear Discriminant Analysis* (LDA) problem (also called Fisher's linear discriminant FLDA) [6, 14]:

$$\textbf{(FLDA)} \quad \max_{\mathbf{P} \in \mathbb{S}_{d,p}} \frac{\phi_B(\mathbf{P})}{\phi_W(\mathbf{P})} = \max_{\mathbf{P} \in \mathbb{S}_{d,p}} \frac{\operatorname{Tr}(\mathbf{P}^T \mathbf{S}_B \mathbf{P})}{\operatorname{Tr}(\mathbf{P}^T \mathbf{S}_W \mathbf{P})}, \tag{1.2}$$

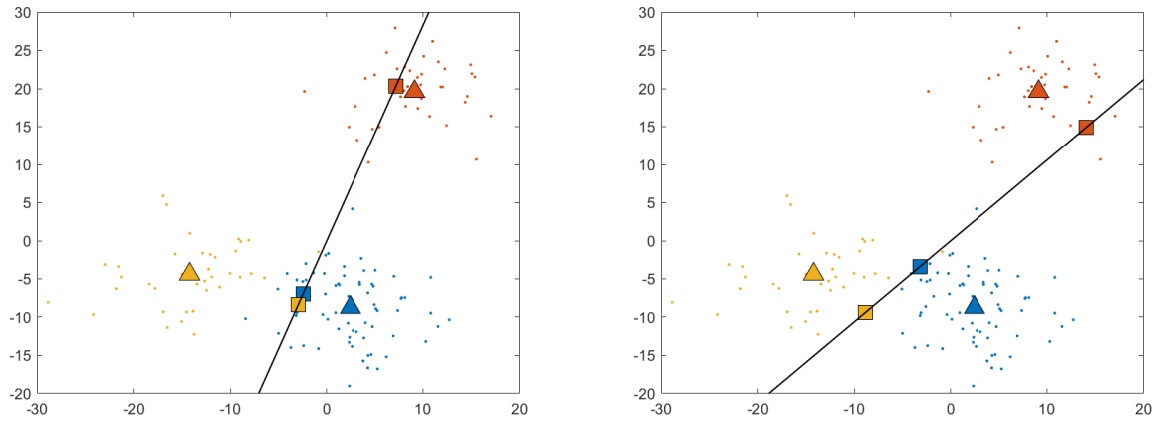
where $\mathbb{S}_{d,p}$ is the Stiefel manifold $\{\mathbf{P} \in \mathbb{R}^{d \times p} : \mathbf{P}^T \mathbf{P} = \mathbf{I}_p\}$. The specific formulation above is called *the trace ratio problem*. While in the one–dimensional case the problem formulation is well agreed, different formulations for the multidimensional case exist in the literature [3, 5, 7], apparently, because the trace ratio problem is considered too hard to handle [14]. In this paper we concentrate on the one-dimensional case ($p = 1$) in which the transformation matrix

becomes a vector, $\mathbf{P} = \mathbf{v} \in \mathbb{R}^{d\times 1}$ which is normalized ($\|\mathbf{v}\|_2 = 1$). The one-dimensional LDA problem therefore in this case reads as

$$\max_{\mathbf{v}:\|\mathbf{v}\|_2=1} \frac{\mathbf{v}^T\mathbf{S}_B\mathbf{v}}{\mathbf{v}^T\mathbf{S}_W\mathbf{v}}. \tag{1.3}$$

1.2 Motivation

One potential weakness of the FLDA approach is caused by the fact that when maximizing the sum of squared distances of all class centers to the mean point of all the dataset, some important information regarding the scattering of the classes in space is lost. The system favors solutions in which there is a large distance between classes to the overall mean, while the distances between different classes might be arbitrarily small. For an illustration of this flaw, see Figure 1a where we can see that in the projected subspace (black line) generated by solving the FLDA problem. The yellow and blue class centers almost overlap, which makes them difficult to separate, whereas their distance to the overall mean is relatively large, and therefore the objective function does not reflect the fact that they are almost overlapping.



(a) Fisher LDA Problem

(b) Max–Min LDA Problem

Figure 1: Fisher LDA and Max–Min LDA.
A two–dimensional dataset represented by small dots where each class is in a different color. The dataset is projected onto a one–dimensional subspace represented by a black line. Triangles represent mean points of each class in the original two–dimensional space, and squares represent the projected mean points of each class. The FLDA objective function values of the Max–Min LDA and the FLDA solutions are 8.431 *and* 9.56 *respectively.*

A simple calculation shows that we can express ϕ_B as the weighted sum of distances between all pairs of the class centers:

$$\phi_B(\mathbf{v}) = \sum_{k>i\ k,i\in[c]} \frac{n_i n_k}{n}(\mathbf{v}^T\left(\boldsymbol{\mu}^{(i)} - \boldsymbol{\mu}^{(k)}\right))^2.$$

To ensure that *any* pair of classes can be easily separated, we suggest an objective function that considers the *minimum* distance between all classes pairs instead of their weighted

squared sum:

$$\phi_m(\mathbf{v}) = \min_{k>i\ k,i\in[c]} \left\{ \left(\mathbf{v}^T(\boldsymbol{\mu}^{(i)} - \boldsymbol{\mu}^{(k)})\right)^2 \right\}. \tag{1.4}$$

Finally, the problem that we seek to solve in this paper is the max-min LDA problem, which consists of maximizing the ratio between the minimum squared distance between class centers ϕ_m and the variance within the classes ϕ_w:

$$\max_{\mathbf{v}} \left\{ \frac{\phi_m(\mathbf{v})}{\phi_W(\mathbf{v})} = \frac{\min_{k>i\ k,i\in[c]} \left\{ \mathbf{v}^T \left(\boldsymbol{\mu}^{(i)} - \boldsymbol{\mu}^{(k)}\right))^2 \right\}}{\mathbf{v}^T \mathbf{S}_W \mathbf{v}} : \|\mathbf{v}\|_2 = 1 \right\}. \tag{MMLDA}$$

The analysis in the paper is made under the following underlying assumption that will be made throughout the paper:

Assumption 1. *(A)* $S_{\mathbf{W}} \succ \mathbf{0}$

(B) $\boldsymbol{\mu}^{(1)}, \boldsymbol{\mu}^{(2)}, \ldots, \boldsymbol{\mu}^{(c)}$ *are linearly independent.*

We note that the positive *semi*definiteness of $\mathbf{S_W}$ is clear from the definition of $\mathbf{S_W}$. The positive definiteness of $\mathbf{S_W}$ is a rather common assumption in the LDA literature [10, 14]. The assumption that $\boldsymbol{\mu}^{(1)}, \boldsymbol{\mu}^{(2)}, \ldots, \boldsymbol{\mu}^{(c)}$ are linearly independent is very reasonable whenever $d > c$.

1.3 Paper Layout

In this paper we present an effective branch and bound scheme to solve the new max-min LDA problem (MMLDA) that takes advantage of the unique structure of the problem. Section 2 introduces a reformulation of the problem as a problem consisting of minimizing a convex quadratic function over a nonconvex polyhedral set, and then describes an additional transformation reducing the number of variables to the number of classes c. Building on these reformulations, Section 3 introduces a branch and bound method for finding the global optimal solution of the problem. Finally, numerical experiments on random data sets illustrating the effectiveness of the proposed method in reducing the size of the enumeration trees are described in Section 4. The preliminary experiments on random instances presented in Section 4 are encouraging and provide a proof of concept. More elaborate tests on real-life problems are needed to assess true potential of the method.

1.4 Notation

Vectors are denoted by boldface lowercase letters, e.g., $\mathbf{y}$, and matrices by boldface uppercase letters, e.g., $\mathbf{B}$. The vectors of all zeros and ones are denoted by $\mathbf{0}$ and $\mathbf{e}$ respectively. The canonical basis of $\mathbb{R}^n$ is denoted by $\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_n$. We use the standard notation $[n] \equiv \{1, 2, \ldots, n\}$ for a positive integer n.

2 Problem Reformulation

2.1 The Inverse Max–Min LDA Problem

Due to the fact that the objective function in (MMLDA) is invariant to scalar multiplications, we can neglect the norm constraint and solve the next problem:

$$\max_{\mathbf{0}\neq\mathbf{v}\in\mathbb{R}^d} \quad \frac{\phi_m(\mathbf{v})}{\mathbf{v}^T\mathbf{S}_W\mathbf{v}}. \tag{2.1}$$

The two problems are equivalent. Indeed, if $\mathbf{v}$ maximizes problem (MMLDA), it will be a maximizer of (2.1) as well. In the other direction, if $\mathbf{v}$ is an optimal solution of (2.1), then $\frac{\mathbf{v}}{\|\mathbf{v}\|_2}$ is an optimal solution of (MMLDA). We will show that the MMLDA problem can be equivalently reformulated as the following problem, which we refer to as the *inverse max–min LDA problem*:

$$\min_{\mathbf{v}\in\mathbb{R}^d}\{\mathbf{v}^T\mathbf{S}_W\mathbf{v} : \phi_m(\mathbf{v}) \geq 1\}. \tag{2.2}$$

Problem (2.1) is in fact a generalized fractional programming problem [16]. By using a scale invariant property of the problem, the next simple result shows that models (2.1) and (2.2) are equivalent, and describes the exact relations between their optimal solutions.

Lemma 2.1. Let $\phi : \mathbb{R}^d \to \mathbb{R}$ be a function satisfying the following properties:

(i) for all $\alpha \in \mathbb{R}$ and $\mathbf{v} \in \mathbb{R}^d$ $\phi(\alpha\mathbf{v}) = \alpha^2\phi(\mathbf{v})$;

(ii) there exists $\hat{\mathbf{v}} \in \mathbb{R}^d$ s.t $\phi(\hat{\mathbf{v}}) > 0$,

and let $\mathbf{S} \in \mathbb{R}^{d\times d}$ where $\mathbf{S} \succ \mathbf{0}$. Consider the following models: the "ratio model"

$$\text{(R)} \qquad \max_{\mathbf{v}\neq\mathbf{0}} \frac{\phi(\mathbf{v})}{\mathbf{v}^T\mathbf{S}\mathbf{v}},$$

and the "inequality model"

$$\text{(I)} \qquad \min_{\mathbf{v}:\phi(\mathbf{v})\geq 1} \mathbf{v}^T\mathbf{S}\mathbf{v}.$$

Then

(a) if $\mathbf{v}^*$ is a maximizer of (R) with maximal value r^*, then $\frac{\mathbf{v}^*}{\sqrt{\phi(\mathbf{v}^*)}}$ is a minimizer of (I) with minimal value $\frac{1}{r^*}$;

(b) any optimal solution of (I) is an optimal solution of (R) as well.

Proof. (a) Let $\mathbf{v}^*$ be an optimal solution of (R) corresponding to the maximal value r^*. Then

$$\phi(\mathbf{v}^*) > 0, \qquad \text{[by property } (ii)\text{]} \tag{2.3}$$

$$\phi\left(\frac{\mathbf{v}^*}{\sqrt{\phi(\mathbf{v}^*)}}\right) = \frac{\phi(\mathbf{v}^*)}{\left(\sqrt{\phi(\mathbf{v}^*)}\right)^2} = 1. \qquad \text{[by property } (i)\text{]} \tag{2.4}$$

Thus $\mathbf{u}^* \equiv \frac{\mathbf{v}^*}{\sqrt{\phi(\mathbf{v}^*)}}$ is a feasible solution of (I). We have

$$r^* = \frac{\phi(\mathbf{v}^*)}{\mathbf{v}^{*T}\mathbf{S}\mathbf{v}^*} = \frac{\frac{1}{\phi(\mathbf{v}^*)}\phi(\mathbf{v}^*)}{\frac{1}{\phi(\mathbf{v}^*)}\mathbf{v}^{*T}\mathbf{S}\mathbf{v}^*} = \frac{1}{\mathbf{u}^{*T}\mathbf{S}\mathbf{u}^*}. \tag{2.5}$$

Consequently, $\mathbf{u}^{*T}\mathbf{S}\mathbf{u}^* = \frac{1}{r^*}$. We are left with the task of showing that $\frac{1}{r^*}$ is the minimal value of (I). To show this, take $\mathbf{x}$, a feasible solution of (I), meaning $\phi(\mathbf{x}) \geq 1$. Then since r^* is the optimal value of (R), it follows that $r^* \geq \frac{\phi(\mathbf{x})}{\mathbf{x}^T\mathbf{S}\mathbf{x}} \geq \frac{1}{\mathbf{x}^T\mathbf{S}\mathbf{x}}$, and we obtain that $\mathbf{x}^T\mathbf{S}\mathbf{x} \geq \frac{1}{r^*}$. (b) Let $\mathbf{v}^*$ be an optimal solution of (I) with minimal value i^*. From property (i), $\phi(\mathbf{0}) = \mathbf{0}$ and hence $\mathbf{v}^* \neq \mathbf{0}$; consequently, $i^* = (\mathbf{v}^*)^T\mathbf{S}\mathbf{v}^* > 0$. Denote $\alpha \equiv \phi(\mathbf{v}^*)$ and let $\mathbf{u} \equiv \frac{\mathbf{v}^*}{\sqrt{\alpha}}$. Obviously $\mathbf{u}$ is a feasible point of (I) with $\mathbf{u}^T\mathbf{S}\mathbf{u} = \frac{1}{\alpha}\mathbf{v}^{*T}\mathbf{S}\mathbf{v}^* = \frac{1}{\alpha}i^*$ but since i^* is the minimal value of (I), $\mathbf{u}^T\mathbf{S}\mathbf{u} \geq i^*$ implying that $\alpha = \phi(\mathbf{v}^*) \leq 1$. On the other hand, $\alpha = \phi(\mathbf{v}^*) \geq 1$ by the fact that $\mathbf{v}^*$ is a feasible solution of (I). Therefore, $\alpha = \phi(\mathbf{v}^*) = 1$. Note that $\frac{\phi(\mathbf{v}^*)}{(\mathbf{v}^*)^T\mathbf{S}\mathbf{v}^*} = \frac{1}{(\mathbf{v}^*)^T\mathbf{S}\mathbf{v}^*} = \frac{1}{i^*}$. To show that $\mathbf{v}^*$ is an optimal solution of (R), take any $\mathbf{v} \neq \mathbf{0}$. If $\phi(\mathbf{v}) \leq 0$, then obviously $\frac{\phi(\mathbf{v})}{\mathbf{v}^T\mathbf{S}\mathbf{v}} \leq 0 \leq \frac{1}{i^*}$. If $\phi(\mathbf{v}) > 0$, define $\mathbf{w} = \frac{1}{\sqrt{\phi(\mathbf{v})}}\mathbf{v}$. Then $\phi(\mathbf{w}) = 1$, and thus, since i^* is the optimal value of (I), it holds that $\mathbf{w}^T\mathbf{S}\mathbf{w} \geq i^*$. Finally, $\frac{\phi(\mathbf{v})}{\mathbf{v}^T\mathbf{S}\mathbf{v}} = \frac{1}{\mathbf{w}^T\mathbf{S}\mathbf{w}} \leq \frac{1}{i^*}$, and hence $\mathbf{v}^*$ is an optimal solution of (R) with a corresponding optimal value of $\frac{1}{i^*}$.

□

Under Assumption 1, the conditions of Lemma 2.1 are met with $\mathbf{S} = \mathbf{S}_W$ and $\phi = \phi_m$, and the equivalence of models (2.1) and (2.2) is established.

2.2 Dimension Reduction of the Inverse Max–Min LDA Problem

We can rephrase (2.2) as a quadratic problem with nonconvex polyhedral constraints, by using the expression of ϕ_m given in (1.4). The inverse max–min LDA problem can thus be rewritten as:

$$\begin{array}{ll} \min\limits_{\mathbf{v}\in\mathbb{R}^d} & \mathbf{v}^T\mathbf{S}_W\mathbf{v} \\ \text{s.t.} & |\mathbf{v}^T\left(\boldsymbol{\mu}^{(i)} - \boldsymbol{\mu}^{(j)}\right)| \geq 1 \quad \text{for all } i,j \in [c],\ i > j. \end{array} \tag{ILDA}$$

The following lemma describes an equivalent reformulation of problem (ILDA) by passing to a decision variables vector $\boldsymbol{\alpha} \in \mathbb{R}^c$ constructed via the relation $\alpha_i = \mathbf{v}^T\boldsymbol{\mu}^{(i)}$, meaning that α_i is the projection of the mean vector $\boldsymbol{\mu}^{(i)}$ on the one-dimensional subspace spanned by the vector $\mathbf{v}$.

Lemma 2.2. Let $\boldsymbol{\alpha} \in \mathbb{R}^c$ be an optimal solution of the problem

$$\begin{array}{ll} \min\limits_{\boldsymbol{\alpha}\in\mathbb{R}^c} & \boldsymbol{\alpha}^T\left(\mathbf{M}^T\mathbf{S}_{\mathbf{W}}^{-1}\mathbf{M}\right)^{-1}\boldsymbol{\alpha} \\ \text{s.t.} & |\alpha_i - \alpha_j| \geq 1 \quad \text{for all } i > j,\ i,j \in [c], \end{array} \tag{2.6}$$

where $\mathbf{M}$ is the $d \times c$ matrix whose columns are $\boldsymbol{\mu}^{(1)}, \boldsymbol{\mu}^{(2)}, \ldots, \boldsymbol{\mu}^{(c)}$. Then[1]

$$\mathbf{v} = \mathbf{S}_W^{-1}\mathbf{M}\left(\mathbf{M}^T\mathbf{S}_W^{-1}\mathbf{M}\right)^{-1}\boldsymbol{\alpha} \tag{2.7}$$

is an optimal solution of (ILDA).

[1]Recall that by Assumption 1[B], $\mathbf{M}$ has full column rank.

Proof. Introducing the variables $\boldsymbol{\alpha} = \mathbf{M}^T\mathbf{v}$, problem (ILDA) becomes

$$\begin{array}{ll} \min\limits_{\mathbf{v}\in\mathbb{R}^d,\boldsymbol{\alpha}\in\mathbb{R}^c} & \mathbf{v}^T\mathbf{S_W}\mathbf{v} \\ \text{s.t.} & \mathbf{M}^T\mathbf{v} = \boldsymbol{\alpha}, \\ & |\alpha_i - \alpha_j| \geq 1 \quad \text{for all } i > j,\ i,j \in [c]. \end{array} \tag{2.8}$$

Fixing $\boldsymbol{\alpha} \in \mathbb{R}^c$, we can solve (2.8) w.r.t. $\mathbf{v}$, which boils down to solving the problem

$$\min_{\mathbf{v}\in\mathbb{R}^d}\{\mathbf{v}^T\mathbf{S_W}\mathbf{v} : \mathbf{M}^T\mathbf{v} = \boldsymbol{\alpha}\},$$

whose optimal solution is

$$\mathbf{v} = \mathbf{S_W^{-1}}\mathbf{M}\left(\mathbf{M}^T\mathbf{S_W^{-1}}\mathbf{M}\right)^{-1}\boldsymbol{\alpha}.$$

Pluging the above expression into (2.8), the objective function can be expressed in terms of $\boldsymbol{\alpha}$ as follows:

$$\mathbf{v}^T\mathbf{S_W}\mathbf{v} = \boldsymbol{\alpha}^T\left(\mathbf{M}^T\mathbf{S_W^{-1}}\mathbf{M}\right)^{-T}\mathbf{M}^T\mathbf{S_W^{-T}}\mathbf{S_W}\mathbf{S_W^{-1}}\mathbf{M}\left(\mathbf{M}^T\mathbf{S_W^{-1}}\mathbf{M}\right)^{-1}\boldsymbol{\alpha} = \boldsymbol{\alpha}\left(\mathbf{M}^T\mathbf{S_W^{-1}}\mathbf{M}\right)^{-1}\boldsymbol{\alpha},$$

and therefore the optimization problem reduces to (2.6) and the relation (2.7) is established. □

The above lemma shows that the d-dimensional problem (ILDA) can be reduced into the c-dimensional problem (2.6), which will be the main problem for which we will develop a solution technique. By denoting $\mathbf{S} = \left(\mathbf{M}^T\mathbf{S}_W^{-1}\mathbf{M}\right)^{-1}$, the main problem becomes

Main Problem

$$\text{(M)} \qquad \begin{array}{ll} \min\limits_{\boldsymbol{\alpha}\in\mathbb{R}^c} & \boldsymbol{\alpha}^T\mathbf{S}\boldsymbol{\alpha} \\ \text{s.t} & |\alpha_i - \alpha_j| \geq 1 \quad \text{for all } i > j,\ i,j \in [c]. \end{array}$$

3 The Branch and Bound Algorithm

In this section we will develop a branch and bound algorithm to solve our main problem (M). Note that solving problem (M) is a difficult task due to the nonconvexity of the feasible set, as illustrated in Figure 2. Thus, branch and bound methods with exponential complexity in the worst case, are a viable option.

To characterize the subproblem defining each node in the branch and bound tree, we begin with a simple observation. Suppose that we have in our disposal the information on the order of the variables $\alpha_1, \alpha_2, \ldots, \alpha_c$ in an optimal solution. Specifically, suppose that we know that i_k is the index of the kth largest value in $\boldsymbol{\alpha}$ for any $k \in [c]$. Then in this unrealistic scenario we can rewrite the constraints in (M) without the absolute values as follows:

$$\alpha_{i_k} - \alpha_{i_{k+1}} \geq 1, \quad k \in [c-1],$$

and problem (M) becomes a convex quadratic problem. The above discussion is relevant *only* if we know the order of an optimal $\boldsymbol{\alpha}$, which we do not in general. We can still solve problem

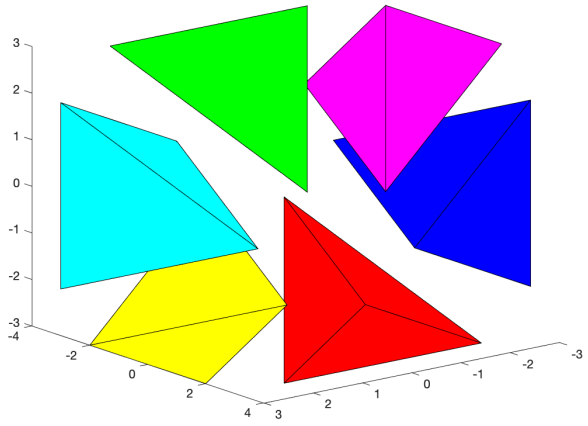


Figure 2: Illustration of the feasible set of (M) in the case $c = 3$.
The set has a decompsition into 6 convex subregions. The set is unbounded, but the figure describes its intersection with the region $[-3, 3]^3$.

(M) by full enumeration over all possible orderings of $\boldsymbol{\alpha}$. This gives us $c!$ possibilities in general. We can avoid orderings that are in reverse order of each other, since if $\boldsymbol{\alpha}^*$ is an optimal solution of (M) then so is $-\boldsymbol{\alpha}^*$. Therefore, full enumeration will require solving $\frac{c!}{2}$ convex quadratic problems, implying that the number of convex problems that need to be solved in this suggested exhaustive algorithm grows exponentially with the number of classes.

Branch and bound (BB) is a general method for solving optimization problems by a clever enumeration of possible solutions [11, 12, 15]. We suggest decomposing the problem according to the order of elements of $\boldsymbol{\alpha}$. We build a series of *subproblems*, each represented by a *node*, where we enforce a specific *partial order* on $\boldsymbol{\alpha}$. By partial order we refer to the information on the identity of the indices corresponding to the largest and smallest components in an optimal solution[2]. The exact definition of the subproblem is given in the next section.

3.1 Subproblem Definition

We define the subproblem as the main problem (M) with additional partial order constraints on $\boldsymbol{\alpha}$. Specifically, each subproblem (and hence node in the BB tree) is defined by two sequences:

(a) The sequence $d = (d_1, d_2, \dots)$ of indices defining the order of the largest values in $\boldsymbol{\alpha}$ in the sense that α_{d_i} is the ith largest element in $\boldsymbol{\alpha}$.

(b) The sequence $a = (a_1, a_2, \dots)$ is the sequence of indices defining the order of the smallest values in $\boldsymbol{\alpha}$ with α_{a_i} being the ith smallest value in $\boldsymbol{\alpha}$.

[2]not to be confused with the mathematical term of "partial order" used to describe a reflexive, antisymmetric and transitive relation on the elements of a given set.

The numbers of elements in the sequences d and a are denoted by $l(d)$ and $l(a)$ respectively. We will avoid situations where there is an overlap of indices between the sequences, which is the same as assuming that $l(d)+l(a) \leq c$. Actually, the complete order of $\boldsymbol{\alpha}$ is already known when $l(d)+l(a) = c-1$, and our underlying assumption will thus be that $l(d)+l(a) \leq c-1$. We denote the last element in each sequence with the subscript -1, meaning that $d_{l(d)} = d_{-1}$ and $a_{l(a)} = a_{-1}$. To summarize,

[A] α_{d_i} is the ith largest element in $\boldsymbol{\alpha}$, for all $i = 1, \ldots, l(d)$,

[B] α_{a_i} is the ith smallest element in $\boldsymbol{\alpha}$, for all $i = 1, \ldots, l(a)$.

We will now formulate explicitly the partial order constraints corresponding to the sequences (d, a), meaning the constraints that mathematically express properties [A] and [B] described above.
The first set of constraints states that $\alpha_{d_1}, \ldots, \alpha_{d_{-1}}$ is a decreasing sequence and that $\alpha_{a_1}, \ldots, \alpha_{a_{-1}}$ is an increasing sequence. By the absolute value constraints, it follows that the distance between any two consecutive members of the sequences is at least 1, leading to the following set of constraints:

$$\alpha_{d_i} - \alpha_{d_{i+1}} \geq 1,\ i = 1, \ldots, l(d) - 1, \tag{3.1}$$
$$\alpha_{a_{i+1}} - \alpha_{a_i} \geq 1,\ i = 1 \ldots, l(a) - 1. \tag{3.2}$$

The second set of constraints quantifies the relation between the elements whose order is unknown and those whose order is known. Denote by $U_{d,a}$ the set of indices of elements in $\boldsymbol{\alpha}$ whose order is unknown, meaning that they are not contained in $D \cup A$ where $D = \{d_1, d_2, \ldots, d_{-1}\}$ and $A = \{a_1, a_2, \ldots, a_{-1}\}$ are the sets comprising the elements in the sequences d and a respectively. Then

$$U_{d,a} := [c] \setminus (D \cup A). \tag{3.3}$$

Since we assume that $l(a)+l(d) \leq c-1$, it follows that in any case $U_{d,a}$ is nonempty and that $|U_{d,a}| = c - l(a) - l(d)$. The second set of constraints states that all the components in $\boldsymbol{\alpha}$ with an unknown order are "between" the components with the largest values $\alpha_{d_1}, \ldots, \alpha_{d_{-1}}$ and those with the smallest values $\alpha_{a_1}, \ldots, \alpha_{a_{-1}}$, leading to the following set of constraints:

$$\alpha_{d_{-1}} - \alpha_u \geq 1, u \in U_{d,a}, \tag{3.4}$$
$$\alpha_u - \alpha_{a_{-1}} \geq 1, u \in U_{d,a}. \tag{3.5}$$

The order constraints are thus formulated by the inequalities (3.1), (3.2), (3.4), (3.5) and we can eliminate the absolute value constraints corresponding to the elements in the sequences d and a. Problem (M) with these additional constraints constitutes the subproblem corresponding to node $[d, a]$ in the BB tree:

Subproblem Corresponding to Node $[d,a]$

$$
(P^{a}_{d}) \qquad \begin{array}{ll} \min\limits_{\boldsymbol{\alpha}\in\mathbb{R}^c} & \boldsymbol{\alpha}^T\mathbf{S}\boldsymbol{\alpha} \\ \text{s.t.} & |\alpha_i-\alpha_j|\geq 1, \quad i>j,\ i,j\in U_{d,a}, \\ & \alpha_{d_i}-\alpha_{d_{i+1}}\geq 1, \quad i=1,\dots,l(d)-1, \\ & \alpha_{a_{i+1}}-\alpha_{a_i}\geq 1, \quad i=1,\dots,l(a)-1, \\ & \alpha_{d_{-1}}-\alpha_u\geq 1, \quad u\in U_{d,a}, \\ & \alpha_u-\alpha_{a_{-1}}\geq 1, \quad u\in U_{d,a}. \end{array} \tag{3.6}
$$

Example 3.1. Suppose that $c=8$. The subproblem corresponding to the sequences $d=(5,1)$ and $a=(4,7,8)$ is given by (note that $U_{d,a}=\{2,3,6\}$):

$$
(P^{4,7,8}_{5,1}) \qquad \begin{array}{ll} \min\limits_{\boldsymbol{\alpha}\in\mathbb{R}^8} & \boldsymbol{\alpha}^T\mathbf{S}\boldsymbol{\alpha} \\ \text{s.t.} & |\alpha_i-\alpha_j|\geq 1, \quad \text{for all } i>j,\ i,j\in\{2,3,6\}, \\ & \alpha_5-\alpha_1\geq 1, \\ & \alpha_7-\alpha_4\geq 1, \quad \alpha_8-\alpha_7\geq 1, \\ & \alpha_1-\alpha_2\geq 1, \quad \alpha_1-\alpha_3\geq 1, \alpha_1-\alpha_6\geq 1, \\ & \alpha_2-\alpha_8\geq 1, \quad \alpha_3-\alpha_8\geq 1, \alpha_6-\alpha_8\geq 1. \end{array}
$$

3.2 The Branch and Bound Scheme

The branch and bound scheme generates an enumeration tree whose nodes correspond to subproblems of the form (P^{a}_{d}) were a and d are sequences coding a given partial order as explained in Section 3.1. The corresponding node will be denoted as $[d,a]$. The root node is $[\emptyset,\emptyset]$, meaning that it corresponds to $(P^{\emptyset}_{\emptyset})$, which is exactly problem (M). At each *branching step*, we split the subproblem denoted as *parent–node* into new *child–node* subproblems, each representing a different possible partial order where we enlarge the number of elements in the solution vector whose order is known. The subproblems (P^{a}_{d}) are still difficult nonconvex problems, but at the core of the the branch and bound scheme we assume that there is a process that produces two elements from these problems:

- **Lower bound.** Using a relaxation of (P^{a}_{d}), a lower bound on $\text{val}(P^{a}_{d})$ is obtained.
- **Feasible solution.** A feasible solution of the original problem (M) is extracted.

We will denote the transformation from the node $[d,a]$ to the corresponding lower bound and feasible solution by $\mathcal{R}$, that is, the relation

$$(\ell,\boldsymbol{\alpha})=\mathcal{R}([d,a])$$

means that ℓ is a lower bound of problem (P^{a}_{d}) obtained by a corresponding relaxation and $\boldsymbol{\alpha}$ is a feasible solution of (M). In this setting we will say that ℓ is the lower bound corresponding to node $[d,a]$ (or to subproblem (P^{a}_{d})). The construction of $\mathcal{R}$ is rather intricate and will be discussed in Sections 3.3 and 3.4.

Since the branching process comprises the enlargement of the set of indices whose order is

known, the feasible set of each parent–node subproblem contains the feasible set of its child–node subproblems, and therefore, the child–node's lower bound will be greater or equal to the parent–node's lower bound. In the case where the upper bound over (M) is lower than the lower bound over a subproblem, then we can close this node and denote it as *fathomed*, that is, closed for further branching. When all the nodes have been fathomed, the branch and bound scheme stops, and an optimal solution of problem (M) is the feasible solution corresponding to the lowest upper bound over (M).

In contrary to the common BB scheme, each branching step does not depend on the results of the solution of the parent–node subproblem. If the number of elements in d is larger than then number of elements in a, we add a new child-node subproblem for each possible choice of the next element in a. In case where the number of elements in both sequences is equal, we add a new child-node subproblem for each possible choice of the next element in d. To ensure that the search process does not revisit permutations that have already been considered, we require the rule that $d_1 < a_1$, meaning that the index of the largest element in the optimal solution (d_1) is smaller than the index of the smallest element (a_1).

The set of open nodes, meaning nodes that were not fathomed, paired with their respective lower bounds is denoted by $\mathcal{P}$. Thus, a typical member of $\mathcal{P}$ has the form $([d, a], v_{lb})$ where v_{lb} is a lower bound on the problem (P_d^a). We choose the next node to process as the open node with the lowest lower bound v_{lb} among all open nodes. The scheme is described below.

Branch and Bound for solving (M)

Step 0: Initialization. Compute $(v_{lb}, \boldsymbol{\alpha}) = \mathcal{R}([\emptyset, \emptyset])$. Set $\boldsymbol{\alpha}^* \leftarrow \boldsymbol{\alpha}$ and $f^* = f(\boldsymbol{\alpha}^*)$. Set $\mathcal{P} = \{([\emptyset, \emptyset], v_{lb})\}$.

Step 1: Choosing a node.

- If $\mathcal{P}$ is empty, then **finish** and the optimal solution is $\boldsymbol{\alpha}^*$.
- Choose $([d, a], v_{lb}) \in \mathcal{P}$ with the lowest lower bound v_{lb} among all the nodes in $\mathcal{P}$ and remove it from $\mathcal{P}$. If the lower bound v_{lb} is higher than the value of the best solution f^*, then **finish** and the optimal solution is $\boldsymbol{\alpha}^*$.

Step 2: Branching step.

(a) For each index $u \in U_{d,a}$ (as defined in (3.3))

(i) Create new subproblem by enlarging a or d as follows:

If $l(d) > l(a)$, enlarge the sequence a by adding u as the last element:

$$\tilde{a} = (a_1, a_2, \ldots, a_{-1}, u), \tilde{d} = d.$$

If $l(d) = l(a)$, enlarge the sequence d, by adding u as the last element:

$$\tilde{a} = a, \tilde{d} = (d_1, d_2, \ldots, d_{-1}, d).$$

(ii) Compute $(v_{lb}, \boldsymbol{\alpha}) = \mathcal{R}([\tilde{d}, \tilde{a}])$. If $v_{lb} < f^*$, then

$*$ set $\mathcal{P} \leftarrow \mathcal{P} \cup \{([\tilde{d}, \tilde{a}], v_{lb})\}$

$*$ if $f(\boldsymbol{\alpha}) < f^*$, then set $\boldsymbol{\alpha}^* \leftarrow \boldsymbol{\alpha}$ and $f^* \leftarrow f(\boldsymbol{\alpha})$.

(b) Goto step 1.

An example of the *full* enumeration tree, without any fathoming of nodes, is given in Figure 3 for the case $c = 4$.

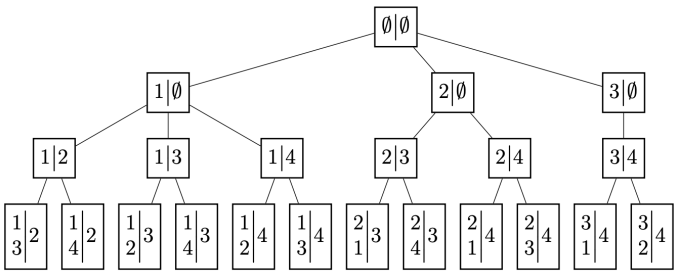


Figure 3: Full enumeration tree for the case $c = 4$.

In the case where $|U_{d,a}| = 1$, the full order is known. Consequently, the absolute value constraints can be removed, and the resulting subproblem (P_d^a) is the following linearly

constrained *convex* quadratic programming problem (here, instead of describing the order by the two sequences a, d, we just assume for the sake of presentation that the sequence $(d_1, d_2, \ldots, d_c)$ is known):

$$\begin{array}{ll} \min\limits_{\boldsymbol{\alpha}\in\mathbb{R}^c} & \boldsymbol{\alpha}^T\mathbf{S}\boldsymbol{\alpha} \\ \text{s.t.} & \alpha_{d_i} - \alpha_{d_{i+1}} \geq 1, \quad \text{for all } i = 1, \ldots, c-1. \end{array} \tag{3.7}$$

The implication of the above is twofold:

1. in the case $|U_{d,a}| = 1$ the operator $\mathcal{R}$ returns the exact optimal value and solution of problem (P_d^a) (no need for a relaxation) and the corresponding node is fathomed.

2. In the worst case, the BB scheme is simply an enumeration of all the possible permutations, see for example Figure 3.

From now on, we will assume that $|U_{d,a}| > 1$.

3.3 The Relaxation

As was explained in the previous section, the BB scheme is based on the construction of relaxations to the subproblems (P_d^a) that are solved at each node. One very natural relaxation of problem (M) can be defined by removing the nonconvex absolute value constraints, leading to a convex quadratic problem that can be solved efficiently. Unfortunately, this is an extremely weak relaxation that empirically might lead to very large enumeration trees, see Section 4.2.1. To generate a tighter relaxation, we first strengthen the formulation.

3.3.1 Strengthening the Formulation

The linear constraints in (P_d^a) (problem (3.6)) form an exact formulation of the partial order constraints. We will add additional linear constraints that constitute "cuts" to subproblem (P_d^a) in the sense that they do not change the feasible set of the subproblem, but they will lead to a possibly tighter relaxation.

Gap constraint. If both $l(d)$ and $l(a)$ are nonzero, then we can add a constraint that takes into account the gap between the $l(d)$ largest and the $l(a)$ smallest element of $\boldsymbol{\alpha}$:

$$\alpha_{d_{-1}} - \alpha_{a_{-1}} \geq c - l(a) - l(d) + 1,$$

where in the above we used the fact the $l(a)$ smallest element in $\boldsymbol{\alpha}$ is the $c - l(a) + 1$ largest element, and between every two consecutive elements in $\boldsymbol{\alpha}$, the gap is at least 1.

Unknown order constraints. The gap between $\alpha_{d_{-1}}$ and the largest element in $\{\alpha_u\}_{u\in U_{d,a}}$ is at least 1. Consequently, the gap between $\alpha_{d_{-1}}$ and the second largest element in $\{\alpha_u\}_{u\in U_{d,a}}$ is at least 2, as the distance between any two elements in $\{\alpha_u\}_{u\in U_{d,a}}$ is at least 1. In general, the gap between $\alpha_{d_{-1}}$ and the ith largest value in $\{\alpha_u\}_{u\in U_{d,a}}$ is at least i. Summing these inequalities over all the elements in U we get

$$\sum_{u\in U_{d,a}} (\alpha_{d_{-1}} - \alpha_u) \geq 1 + 2 + \cdots + |U_{d,a}| = \gamma(|U_{d,a}|),$$

where $\gamma(t) = \frac{1}{2}t(t+1)$. Similarly,

$$\sum_{u \in U_{d,a}} (\alpha_u - \alpha_{a_{-1}}) \geq \gamma(|U_{d,a}|)$$

Note that $|U_{d,a}| = c - l(a) - l(d)$.

Quadratic constraint. Lastly, we add a quadratic constraint that expresses the relation within the group $U_{d,a}$ of elements of $\boldsymbol{\alpha}$ with unknown order. We can square the constraints of the absolute values of the main problem (M) and represent them as the following quadratic constraints:

$$((\mathbf{e}_i - \mathbf{e}_j)^T \boldsymbol{\alpha})^2 = (\alpha_i - \alpha_j)^2 \geq 1 \quad \text{for all } i > j,\ i, j \in U_{d,a}. \tag{3.8}$$

Taking into consideration that the distance between each pair of coordinates is a least 1, then we have $|U_{d,a}| - 1$ pairs of components with a distance greater than 1, $|U_{d,a}| - 2$ pairs with distance greater than 2 and so on. We thus conclude that summing up the $\binom{|U_{d,a}|}{2}$ inequalities in (3.8), results in the inequality

$$\boldsymbol{\alpha}^T \mathbf{E}_{U_{d,a}} \boldsymbol{\alpha} \geq \sum_{i=1}^{|U_{d,a}|-1} i \cdot (|U_{d,a}| - i)^2 = \frac{|U_{d,a}|^2}{12}(|U_{d,a}|^2 - 1),$$

where

$$\mathbf{E}_{U_{d,a}} = \sum_{\substack{i>j,\\ i,j \in U_{d,a}}} (\mathbf{e}_i - \mathbf{e}_j)(\mathbf{e}_i - \mathbf{e}_j)^T. \tag{3.9}$$

We can finally formulate the subproblem corresponding to node $[d, a]$ in the suggested branch and bound algorithm. In case where $|U_{d,a}| > 1$ we get the following subproblem:

$$\begin{array}{lll}
\min\limits_{\boldsymbol{\alpha} \in \mathbb{R}^c} & \boldsymbol{\alpha}^T \mathbf{S} \boldsymbol{\alpha} & \\
\text{s.t.} & |\alpha_i - \alpha_j| \geq 1, & i > j,\ i, j \in U_{d,a}, \\
& \alpha_{d_i} - \alpha_{d_{i+1}} \geq 1, & i = 1, \ldots, l(d) - 1, \\
& \alpha_{a_{i+1}} - \alpha_{a_i} \geq 1, & i = 1, \ldots, l(a) - 1, \\
& \alpha_{d_{-1}} - \alpha_u \geq 1, & u \in U_{d,a}, \\
& \alpha_u - \alpha_{a_{-1}} \geq 1, & u \in U_{d,a}, \\
& \alpha_{d_{-1}} - \alpha_{a_{-1}} \geq (c - l(a)) - l(d) + 1, & \\
& \sum_{u \in U_{d,a}} (\alpha_{d_{-1}} - \alpha_u) \geq \gamma(|U_{d,a}|), & \\
& \sum_{u \in U_{d,a}} (\alpha_u - \alpha_{a_{-1}}) \geq \gamma(|U_{d,a}|), & \\
& \boldsymbol{\alpha}^T \mathbf{E}_{U_{d,a}} \boldsymbol{\alpha} \geq \frac{|U_{d,a}|^2}{12}(|U_{d,a}|^2 - 1). &
\end{array} \tag{3.10}$$

By alternating between enlarging sequences a and d, we increase the number of linear constraints in each subproblem (3.10) in comparison to the option of using only one sequence.

Example 3.2. Continuing Example 3.1, the subproblem corresponding to the sequences

$a = (4, 7, 8), d = (5, 1)$ is (note that that $U_{d,a} = \{2, 3, 6\}$)

$$
\begin{array}{ll}
\min\limits_{\boldsymbol{\alpha}\in\mathbb{R}^8} & \boldsymbol{\alpha}^T\mathbf{S}\boldsymbol{\alpha} \\
\text{s.t.} & |\alpha_i - \alpha_j| \geq 1, \qquad\qquad \forall i > j,\ i, j \in \{2, 3, 6\}, \\
& \alpha_5 - \alpha_1 \geq 1, \\
& \alpha_7 - \alpha_4 \geq 1, \alpha_8 - \alpha_7 \geq 1, \\
& \alpha_1 - \alpha_2 \geq 1, \alpha_1 - \alpha_3 \geq 1, \alpha_1 - \alpha_6 \geq 1, \\
& \alpha_2 - \alpha_8 \geq 1, \alpha_3 - \alpha_8 \geq 1, \alpha_6 - \alpha_8 \geq 1 \\
& \alpha_1 - \alpha_8 \geq 4, \\
& 3\alpha_1 - \alpha_2 - \alpha_3 - \alpha_6 \geq 6, \\
& \alpha_2 + \alpha_3 + \alpha_6 - 3\alpha_8 \geq 6, \\
& (\alpha_2 - \alpha_3)^2 + (\alpha_2 - \alpha_6)^2 + (\alpha_3 - \alpha_6)^2 \geq 6.
\end{array}
$$

3.3.2 Matrix Lifting

Problem (3.10) is inherently nonconvex, and consequently, we will solve an appropriate relaxation. To construct the relaxation, we first remove the absolute value constraints:

$$
\begin{array}{lll}
\min\limits_{\boldsymbol{\alpha}\in\mathbb{R}^c} & \boldsymbol{\alpha}^T\mathbf{S}\boldsymbol{\alpha} & \\
\text{s.t.} & \alpha_{d_i} - \alpha_{d_{i+1}} \geq 1, & i = 1, \ldots, l(d) - 1, \\
& \alpha_{a_{i+1}} - \alpha_{a_i} \geq 1, & i = 1, \ldots, l(a) - 1, \\
& \alpha_{d_{-1}} - \alpha_u \geq 1, & u \in U_{d,a}, \\
& \alpha_u - \alpha_{a_{-1}} \geq 1, & u \in U_{d,a}, \\
& \alpha_{d_{-1}} - \alpha_{a_{-1}} \geq (c - l(a)) - l(d) + 1, & \\
& \sum_{u\in U_{d,a}} (\alpha_{d_{-1}} - \alpha_u) \geq \gamma(|U_{d,a}|), & \\
& \sum_{u\in U_{d,a}} (\alpha_u - \alpha_{a_{-1}}) \geq \gamma(|U_{d,a}|), & \\
& \boldsymbol{\alpha}^T\mathbf{E}_{U_{d,a}}\boldsymbol{\alpha} \geq \frac{|U_{d,a}|^2}{12}(|U_{d,a}|^2 - 1). &
\end{array}
\tag{3.11}
$$

Problem (3.11) is nonconvex due to the last quadratic constraint. One possibility to generate a convex approximation of the problem is to remove the quadratic constraint. However, we will show in Section 4.2.1 that this suggested relaxation performs poorly. We will therefore choose a different path and use the standard matrix lifting technique [17, 18]. We begin by defining the new variables matrix $\mathbf{X} \in \mathbb{R}^{(c+1)\times(c+1)}$ as $\mathbf{X} = \begin{pmatrix} \boldsymbol{\alpha}\boldsymbol{\alpha}^T & \boldsymbol{\alpha} \\ \boldsymbol{\alpha}^T & 1 \end{pmatrix}$. Denote by $\mathbf{X}^1$ the upper left $c \times c$-dimensional submatrix, $\mathbf{X}^1 = \mathbf{X}_{1:c,1:c}$, by $\mathbf{x}^2$ the c-dimensional vector $\mathbf{x}^2 = \mathbf{X}_{1:c,c+1}$ and by x^4 the scalar $X_{c+1,c+1}$. All together, we have

$$
\mathbf{X} = \begin{pmatrix} \mathbf{X}^1 & \mathbf{x}^2 \\ (\mathbf{x}^2)^T & x^4 \end{pmatrix}.
$$

Using the above relation, we conclude that an equivalent problem to (3.11) is the next minimization problem:

$$
\begin{array}{lll}
\min\limits_{\mathbf{X}\in\mathbb{S}^{c+1}} & \mathrm{Tr}(\mathbf{S}\mathbf{X}^1) & \\
\text{s.t.} & x_{d_i}^2 - x_{d_{i+1}}^2 \geq 1, & i = 1,\ldots,l(d)-1, \\
& x_{a_{i+1}}^2 - x_{a_i}^2 \geq 1, & i = 1,\ldots,l(a)-1, \\
& x_{d_{-1}}^2 - x_u^2 \geq 1, & u \in U_{d,a}, \\
& x_u^2 - x_{a_{-1}}^2 \geq 1, & u \in U_{d,a}, \\
& x_{d_{-1}}^2 - x_{a_{-1}}^2 \geq c - l(a) - l(d) + 1, & \\
& \sum_{u\in U_{d,a}}(x_{d_{-1}}^2 - x_u^2) \geq \gamma(|U_{d,a}|), & \\
& \sum_{u\in U_{d,a}}(x_u^2 - x_{a_{-1}}^2) \geq \gamma(|U_{d,a}|), & \\
& \mathrm{Tr}(\mathbf{E}_{U_{d,a}}\mathbf{X}^1) \geq \dfrac{|U_{d,a}|^2}{12}(|U_{d,a}|^2 - 1), & \\
& x^4 = 1, & \\
& \mathbf{X} \succeq \mathbf{0}, & \\
& \mathrm{rank}(\mathbf{X}) = 1. &
\end{array}
\tag{3.12}
$$

Next we discard the nonconvex rank constraint and obtain the following semidefinite relaxation (SDR) of the problem:

$$
(P_{SDR}^{a_d^d})
\qquad
\begin{array}{lll}
\min\limits_{\mathbf{X}\in\mathbb{S}^{c+1}} & \mathrm{Tr}(\mathbf{S}\mathbf{X}_1) & \\
\text{s.t.} & x_{d_i}^2 - x_{d_{i+1}}^2 \geq 1, & i = 1,\ldots,l(d)-1, \\
& x_{a_{i+1}}^2 - x_{a_i}^2 \geq 1, & i = 1,\ldots,l(a)-1, \\
& x_{d_{-1}}^2 - x_u^2 \geq 1, & u \in U_{d,a}, \\
& x_u^2 - x_{a_{-1}}^2 \geq 1, & u \in U_{d,a}, \\
& x_{d_{-1}}^2 - x_{a_{-1}}^2 \geq c - l(a) - l(d) + 1, & \\
& \sum_{u\in U_{d,a}}(x_{d_{-1}}^2 - x_u^2) \geq \gamma(|U_{d,a}|), & \\
& \sum_{u\in U_{d,a}}(x_u^2 - x_{a_{-1}}^2) \geq \gamma(|U_{d,a}|), & \\
& \mathrm{Tr}(\mathbf{E}_{U_{d,a}}\mathbf{X}_1) \geq \dfrac{|U_{d,a}|^2}{12}(|U_{d,a}|^2 - 1), & \\
& x^4 = 1, & \\
& \mathbf{X} \succeq \mathbf{0}. &
\end{array}
$$

The optimal value of $(P_{SDR}^{a_d^d})$ is obviously a lower bound on $(P^{a_d^d})$, and this will be the lower bound that will be used during the BB scheme. In our implementation, the subproblems were solved using an interior point method constructed similarly to the one proposed in [9] for the max-cut problem.

Remark 3.3 (number of constraints)**.** The complexity of the solution of each semidefinite relaxation $(P_{SDR}^{a_d^d})$ is at least $O(k^3)$ [9], where k is the number of constraints. We could have added to problem (3.10) a set of quadratic constraints for each pair of elements in $\boldsymbol{\alpha}$ in the form of (3.8). While this might lead to a tighter relaxation when constructing problem $(P_{SDR}^{a_d^d})$, it will also result with a considerable larger problem since the number of constraints in the suggested modified problem is $O(c^2)$ while the number of constraints in the suggested formulation is $(P_{SDR}^{a_d^d})$ is $O(c)$. This means that the dependence of the complexity in c in the

modified problem will be $O(c^6)$ instead of $O(c^3)$. This dependence of the complexity in the number of classes lead us to ignore the suggested set of quadratic inequalities.

3.4 Constructing a Feasible Solution of (M)

The optimal value of problem $(P^{a}_{d}{}_{SDR})$ is a lower bound over the corresponding problem (3.10). The second objective is to extract a feasible solution $\boldsymbol{\alpha} \in \mathbb{R}^c$ of (M) that will be driven from an optimal solution $\tilde{\mathbf{X}}$ to problem $(P^{a}_{d}{}_{SDR})$. The extraction process is done in three stages, and is described below.

Extraction Process:

Input: $\tilde{\mathbf{X}} \in \mathbb{R}^{(c+1)\times(c+1)}$–optimal solution of $(P^{a}_{d}{}_{SDR})$.
Output: $\boldsymbol{\alpha} \in \mathbb{R}^c$–feasible solution of (M).

- **Stage 1.** Find a vector $\mathbf{x} \in \mathbb{R}^c$ for which the rank–one matrix $\begin{pmatrix}\mathbf{x}\\1\end{pmatrix}\begin{pmatrix}\mathbf{x}\\1\end{pmatrix}^T$ is closest to $\tilde{\mathbf{X}}$ in the Frobenius norm, meaning that $\mathbf{x}$ is an optimal solution of

$$\text{(RNK)} \qquad \min_{\mathbf{x}\in\mathbb{R}^{c+1}} \quad \left\| \begin{pmatrix}\mathbf{x}\\1\end{pmatrix}\begin{pmatrix}\mathbf{x}\\1\end{pmatrix}^T - \tilde{\mathbf{X}} \right\|_F^2$$

- **Stage 2.** Find a vector $\mathbf{y}$, which is feasible w.r.t. the original problem (M), and is closest as possible to $\mathbf{x}$:

$$\mathbf{y} \in \operatorname*{argmin}_{\mathbf{y}\in\mathbb{R}^c} \{\|\mathbf{y}-\mathbf{x}\|_2 : |y_i - y_j| \geq 1 \quad \text{for all } i,j \in [c]\}.$$

- **Stage 3.** Employ the gradient projection method on problem (M) with $\mathbf{y}$ being the initial point and obtain a vector $\boldsymbol{\alpha} \in \mathbb{R}^c$.

To summarize, if we denote by $\mathbf{X}_{d,a}$ an optimal solution of $(P^{a}_{d}{}_{SDR})$ and by $\mathcal{E}(\mathbf{X}_{d,a})$ the output of the extraction process, we have

$$\mathcal{R}([d,a]) = (\mathrm{Tr}(\mathbf{S}\mathbf{X}_{d,a}), \mathcal{E}(\mathbf{X}_{d,a})).$$

Note that the optimization problems that are being solved in stages 1 and 2 are nonconvex, and despite this, we will now show that they can be solved efficiently. The third stage aims at finding a feasible solution with a better objective function.

3.4.1 Stage 1: Best Rank-One Approximation

Let $\tilde{\mathbf{X}} \in \mathbb{R}^{(c+1)\times(c+1)}$ be an optimal solution of $(P^{a}_{d}{}_{SDR})$. It is not guaranteed that the optimal solution is a rank–one matrix, meaning that there is no guarantee that a c-length vector $\mathbf{x}$

exists such that $\tilde{\mathbf{X}} = \begin{pmatrix}\mathbf{x}\\1\end{pmatrix}\begin{pmatrix}\mathbf{x}\\1\end{pmatrix}^T$ (recall that $\tilde{X}_{c+1,c+1} = 1$). We seek to find the best rank-one approximation of $\tilde{\mathbf{X}}$ by solving problem (RNK). A natural approach is to compute a leading eigenvector $\mathbf{v}$ of $\tilde{\mathbf{X}}$ and define the approximate vector as the vector $\frac{1}{v_{c+1}}\mathbf{v}_{1:c}$. However, this scheme does not produce the global minimizer of problem (RNK), and we suggest to find an exact solution. To do so, we first write $\tilde{\mathbf{X}}$ (as before) as

$$\tilde{\mathbf{X}} = \begin{pmatrix}\tilde{\mathbf{X}}^1 & \tilde{\mathbf{x}}^2\\(\tilde{\mathbf{x}}^2)^T & 1\end{pmatrix}. \tag{3.13}$$

The objective function of (RNK) is

$$\left\|\tilde{\mathbf{x}}\tilde{\mathbf{x}}^T - \tilde{\mathbf{X}}\right\|_F^2 = \|\tilde{\mathbf{X}}\|_F^2 + \|\tilde{\mathbf{x}}\tilde{\mathbf{x}}^T\|_F^2 - 2\,\mathrm{Tr}(\tilde{\mathbf{X}}\tilde{\mathbf{x}}\tilde{\mathbf{x}}^T). \tag{3.14}$$

Denote $\tilde{\mathbf{x}} = \begin{pmatrix}\mathbf{x}\\1\end{pmatrix}$. By (3.13) and some simple algebra, it follows that

$$\|\tilde{\mathbf{x}}\tilde{\mathbf{x}}^T\|_F^2 = (\|\mathbf{x}\|_2^2 + 1)^2,\ \mathrm{Tr}(\tilde{\mathbf{X}}\tilde{\mathbf{x}}\tilde{\mathbf{x}}^T) = \mathbf{x}^T\tilde{\mathbf{X}}_1\mathbf{x} + 2\mathbf{x}^T\tilde{\mathbf{x}}^2 + 1.$$

Plugging the above into (3.14), we obtain that

$$\left\|\tilde{\mathbf{x}}\tilde{\mathbf{x}}^T - \mathbf{X}^*\right\|_F^2 = \|\tilde{\mathbf{X}}\|_F^2 + (\|\mathbf{x}\|_2^2 + 1)^2 - 2\mathbf{x}^T\tilde{\mathbf{X}}_1\mathbf{x} - 4\mathbf{x}^T\tilde{\mathbf{x}}^2 - 2. \tag{3.15}$$

Thus, ignoring constant terms, problem (RNK) can be rewritten as

$$\min_{\mathbf{x}\in\mathbb{R}^c}\left\{(\|\mathbf{x}\|_2^2 + 1)^2 - 2\mathbf{x}^T\tilde{\mathbf{X}}_1\mathbf{x} - 4\mathbf{x}^T\tilde{\mathbf{x}}^2\right\}.$$

The above problem can be recast as the following double minimization problem (ignoring constant terms):

$$\begin{array}{ll}\min\limits_{\mathbf{x}\in\mathbb{R}^c,\gamma\in\mathbb{R}} & (\gamma+1)^2 - 2\mathbf{x}^T\tilde{\mathbf{X}}_1\mathbf{x} - 4\mathbf{x}^T\tilde{\mathbf{x}}^2\\ \text{s.t.} & \|\mathbf{x}\|_2^2 = \gamma,\end{array} \tag{3.16}$$

which is a generalized trust region subproblem (GTRS) that can be solved efficiently, see e.g., [13]. We summarize this discussion in the following lemma.

Lemma 3.4. Suppose that $(\mathbf{x}, \gamma)$ is an optimal solution of (3.16). Then $\mathbf{x}$ is an optimal solution of (RNK).

3.4.2 Stage 2: Feasible Solution Extraction

At the end of stage 1, we have at our disposal a vector $\mathbf{x} \in \mathbb{R}^c$ which is an optimal solution of (RNK). Unfortunately, $\mathbf{x}$ is not guaranteed to be in the feasible set of (M). It is thus natural to search for a feasible solution of (M) which is closest to $\mathbf{x}$. Denote by G the feasible set of (M):

$$G := \{\mathbf{x} : |x_i - x_j| \geq 1 \quad \text{for all } i, j \in [c],\ i > j\}. \tag{3.17}$$

Thus, we seek to find a point in the orthogonal projection of $\mathbf{x}$ onto G:

$$P_G(\mathbf{x}) = \operatorname*{argmin}_{\mathbf{y}\in G} \|\mathbf{y} - \mathbf{x}\|_2^2. \tag{3.18}$$

Note that since G is not convex, the orthogonal projection $P_G(\mathbf{x})$ might be a set and is not necessarily single-valued. The objective is to find a single member of the orthogonal projection set. In principle, computing the orthogonal projection operator on nonconvex sets is a difficult task, but in this case, by exploiting the symmetry of G, we will show how computing an orthogonal projection can be done efficiently.

The set G is symmetric with respect to permutations. Using this symmetry, by [1, Theorem 3.1], the following property known as *the order preservation property* holds: if the vector $\mathbf{x}$ can be ordered via

$$x_{i_1} \geq x_{i_2} \geq \cdots \geq x_{i_c}, \tag{3.19}$$

with $i_1, i_2, \ldots, i_c$ being a reordering of the elements in $[c]$, then there exists $\mathbf{y} \in P_G(\mathbf{x})$ satisfying $y_{i_1} \geq y_{i_2} \geq \cdots \geq y_{i_c}$.
The conclusion from the above discussion, and the specific structure of G, is that in order to project $\mathbf{x}$ onto G, the following two steps should be invoked:

- **Step 1:** Find indices $i_1, i_2, \ldots, i_c \in [c]$ that form a permutation of $[c]$ such that (3.19) holds.
- **Step 2:** Find $\mathbf{y} = P_{\tilde{G}}(\mathbf{x})$, where

$$\tilde{G} = \{\mathbf{y} \in \mathbb{R}^c : y_{i_k} \geq y_{i_{k+1}} + 1, k \in [c-1]\}.$$

We have thus reduced the nonconvex problem (3.18) to the convex problem of finding the orthogonal projection onto the closed and convex set $\tilde{G}$. The projection onto $\tilde{G}$ can be done very efficiently by further reducing the projection problem. Indeed, the problem that we seek to solve is

$$\min\{\|\mathbf{y} - \mathbf{x}\|_2^2 : y_{i_k} \geq y_{i_{k+1}} + 1, k \in [c-1]\}.$$

Making the change of variables $z_{i_k} = y_{i_k} + (k-1)$, the problem is transformed into

$$\min\{\|\mathbf{z} + \mathbf{d} - \mathbf{x}\|_2^2 : z_{i_k} \geq z_{i_{k+1}}, k \in [c-1]\},$$

where $\mathbf{d} = (0, 1, \ldots, c-1)^T$. The above problem of projecting $\mathbf{x} - \mathbf{d}$ onto $\hat{G} = \{\mathbf{z} \in \mathbb{R}^c : z_{i_k} \geq z_{i_{k+1}}, k \in [c-1]\}$ is identical to the *Isotonic Regression with respect to a Complete order* (IRC) problem [2, 8] (up to a permutation of the variables) and it can be solved using the Pool Adjacent Violators (PAV) algorithm with an efficient computational complexity of $O(c)$ [2, 8].

3.4.3 Stage 3: Refinement via Gradient Projection

The vector $\mathbf{y} \in G$ (G being the feasible set of (M), given in (3.17)), which is a product of stage 2, induces the upper bound $f(\mathbf{y})$ on the optimal value of problem (M). We can potentially improve the upper bound by employing an optimization algorithm that is guaranteed to produce function values that are nonincreasing. In the suggested BB method, we use the gradient projection method in which we take a step in the direction of the gradient, and then project it onto the feasible set:

$$\mathbf{x}^{k+1} = P_G\left(\mathbf{x}^k - \frac{2}{L}\mathbf{S}\mathbf{x}^k\right), \quad k = 0, 1, \ldots \tag{3.20}$$

The constant L is taken to be larger than the Lipschitz constant of the gradient of the objective function, meaning $L > 2\lambda_{\max}(\mathbf{S})$. The starting point is $\mathbf{x}^0 = \mathbf{y}$ and the stopping criterion will be $f(\mathbf{x}^{k+1}) > f(\mathbf{x}^k) - \varepsilon$. The convergence of the scheme (3.20) to a critical point of (M) was established in [4]. In our numerical results we used $\varepsilon = 10^{-10}$ and maximum of 300 iterations.

4 Numerical Experiments

In this section we demonstrate the proposed BB scheme as described in Section 3 for solving problem (M). We begin by comparing a full enumeration (FE) solution with the BB scheme. Next, we empirically demonstrate how our specific construction of the subproblem leads to a stronger relaxation than other simpler constructions. We show the effect of using the gradient projection method (Section 3.4.3) and best rank–one approximation stage (Section 3.4.1) on the performance of the scheme. Lastly, a comparison of CPU times of each scheme is given. All the datasets used in this section were randomly generated, as described in Section 4.1.

4.1 Datasets Generation

Given the number of classes c, we generated a dataset and built problem (MMLDA), and then transformed the problem into problem (M) using Lemma 2.2. We randomly generated c classes in $\mathbb{R}^{100}$. Each class contains n_i samples generated from a normal distribution with mean vector $\boldsymbol{\mu}_i$ and covariance matrix $\sigma_i^2\mathbf{I}$. For each class, the parameters $n_i, \boldsymbol{\mu}_i, \sigma_i$ were set as follows:

- n_i – the number of samples in each class was sampled from a uniform discrete distribution within the range $[201, 500]$
- σ_i – the standard deviation of each class was sampled from a normal distribution with zero mean and standard deviation 5.
- $\boldsymbol{\mu}_i$ – each element in the mean vector was independently sampled from the standard continuous uniform distribution on the open interval $(-20, 20)$.

4.2 Results

As a first demonstration of the effectiveness of the BB scheme, consider the tree in Figure 4 representing the BB scheme applied on problem (M) in a ten classes case. Each node represents a subproblem corresponding to a given information on the partial order. In this specific case, the total number of subproblems solved is 222, whereas in a full enumeration solution there are 1,814,400 subproblems.

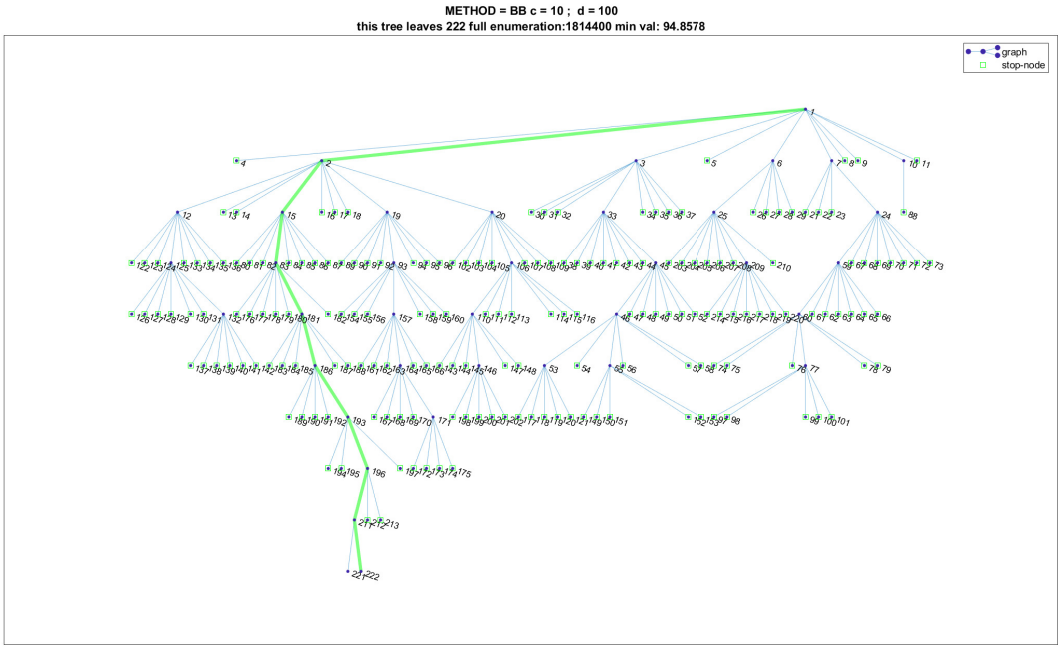


Figure 4: BB tree for a ten classes problem in $\mathbb{R}^{100}$.

Each level (horizontal row of nodes) in Figure 4 contains all the nodes with the same number of known partial order elements. The bold green path starts from the root node, representing the subproblem with no assumption on a partial order, and ends in a node representing a complete order of $\boldsymbol{\alpha}$ corresponding to the optimal solution. Each node in the path represents a subproblem with an appropriate partial order constraints. Next to each node is the chronical order in which the subproblems were processed.
While Figure 4 refers to a specific dataset for demonstration purposes, Figure 5 examines the average number of subproblems solved in the BB scheme (y-axis) for a given number of classes (x-axis) over ten randomly generated datasets. As can be clearly seen in Figure 5, the BB scheme reduces the number of explored subproblems by several orders of magnitude.

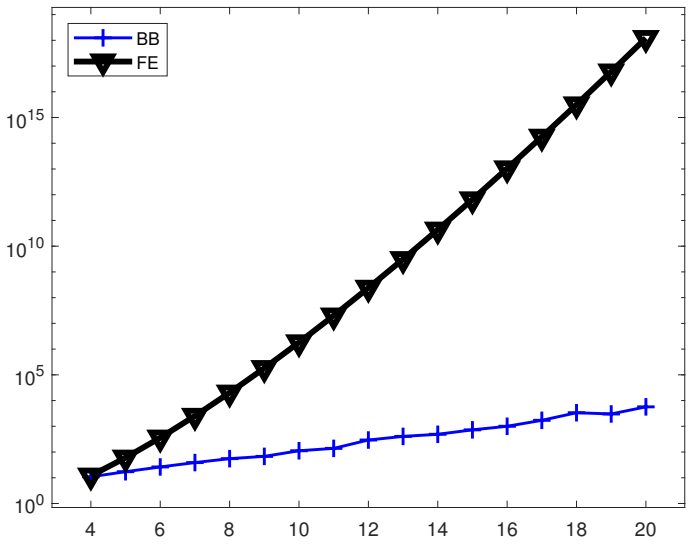


Figure 5: Number of explored subproblems in the BB and FE schemes for different number of classes.

4.2.1 Variations on the Branch and Bound Scheme

Recall that subproblem (3.6) represents an exact formulation of the partial order constraints. However, during the relaxation process we include additional constraints–three linear constraints and one quadratic constraint–leading to the formulation (3.10).

Denote by BBO the BB scheme where we use only the original order constraints, meaning that we consider the variant in which the relaxation considers the original form (3.6) without any additional constraints.

Another variant is to consider problem (3.10) without the additional quadratic constraint:

$$
\begin{array}{lll}
\min\limits_{\boldsymbol{\alpha}\in\mathbb{R}^c} & \boldsymbol{\alpha}^T\mathbf{S}\boldsymbol{\alpha} & \\
\text{s.t.} & |\alpha_i-\alpha_j|\geq 1, & i>j,\ i,j\in U_{d,a},\\
& \alpha_{d_i}-\alpha_{d_{i+1}}\geq 1, & i=1,\dots,l(d)-1,\\
& \alpha_{a_{i+1}}-\alpha_{a_i}\geq 1, & i=1,\dots,l(a)-1,\\
& \alpha_{d_{-1}}-\alpha_u\geq 1, & u\in U_{d,a},\\
& \alpha_u-\alpha_{a_{-1}}\geq 1, & u\in U_{d,a},\\
& \alpha_{d_{-1}}-\alpha_{a_{-1}}\geq (c-l(a))-l(d)+1, & \\
& \sum_{u\in U_{d,a}}(\alpha_{d_{-1}}-\alpha_u)\geq \gamma(|U_{d,a}|), & \\
& \sum_{u\in U_{d,a}}(\alpha_u-\alpha_{a_{-1}})\geq \gamma(|U_{d,a}|). &
\end{array}
\tag{4.1}
$$

Denote by BBwoQ the BB scheme where the relaxation is based on problems of the form (4.1). The corresponding relaxations in the BBO and BBwoQ schemes are problems (3.6) and (4.1) respectively without the absolute value constraints. Thus, each relaxation is a convex quadratic problem and can be solved efficiently. The rest of the scheme remains unchanged, except for stage 1 in the extraction process described in Section 3.4, which becomes irrelevant.

We denote by BBwoGP the BB scheme without using the gradient projection method (Section 3.4.3), that is, applying only stages 1 and 2 in the extraction process. We denote by BBsimpleVEC the BB scheme where in stage 1 of the process of constructing a feasible

solution (Section 3.4) instead of using best rank–one approximation stage (Section 3.4.1) we use a different simpler and popular heuristic approach to extract a vector from the optimal solution matrix $\tilde{\mathbf{X}}$ of (P^d_{SDR}). Let $\tilde{\mathbf{x}}$ be the eigenvector corresponding to the largest eigenvalue of $\tilde{\mathbf{X}}$ and let $\mathbf{x}$ be the first c elements in $\tilde{\mathbf{x}}$ and y be the last element. Then stage 1 returns $\frac{1}{y}\mathbf{x}$.
We denote by BBwoGPsimpleVEC the BBsimpleVEC scheme without applying stage 3 - the gradient projection method (Section 3.4.3) in the extraction process (Section 3.4).

The simulation results suggest that the BBO scheme is ineffective. Indeed, Figure 6 compares the average number of subproblems used in the solution of the BBO scheme, averaged on ten realizations, for each number of classes c. Clearly, the BBO scheme performs poorly—the number of subproblems it explores is even larger than the number of subproblems explored in the full enumeration solution. It can be also seen that BBwoQ is dominated by BB. As a further illustration, Figure 7 shows the enumeration tree of the BBwoQ scheme applied to the same dataset as the one leading to Figure 4. The amount of subproblems solved in the BBwoQ scheme is more than ten times the number of subproblems solved in the BB scheme (2343 instead of 222).

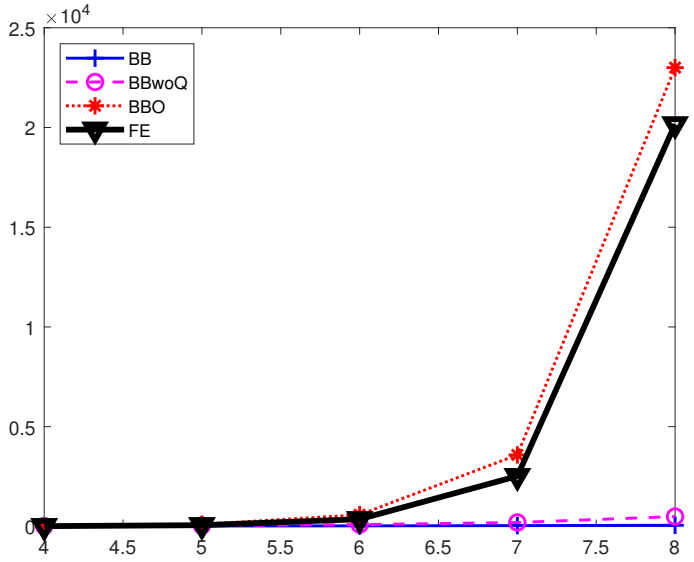


Figure 6: Comparison of the BB, BBwoQ and BBO schemes for different number of classes c.

In Figure 8 a comparison between the BB, BBwoQ and BBwoGP schemes is given. The average number of subproblems used in the solution of the each scheme, averaged on ten realizations, for each number of classes c is given (note that the y-axis is in log-scale). The suggested BB scheme is superior to all the other approaches. We see that the difference in the number of nodes in BBwoQ compared with BB grows drastically as c grows.
Using the BBwoGP scheme, we solved about twice the number of subproblems compared to the BB scheme. Table 1 summarizes all the results.
In Figure 9 a comparison between the BB, BBwoGP, and BBsimpleVEC, BBwoGPsimpleVEC schemes is given. The average number of subproblems used in the solution of the each scheme, averaged on ten realizations, for each number of classes c is given. When ex-

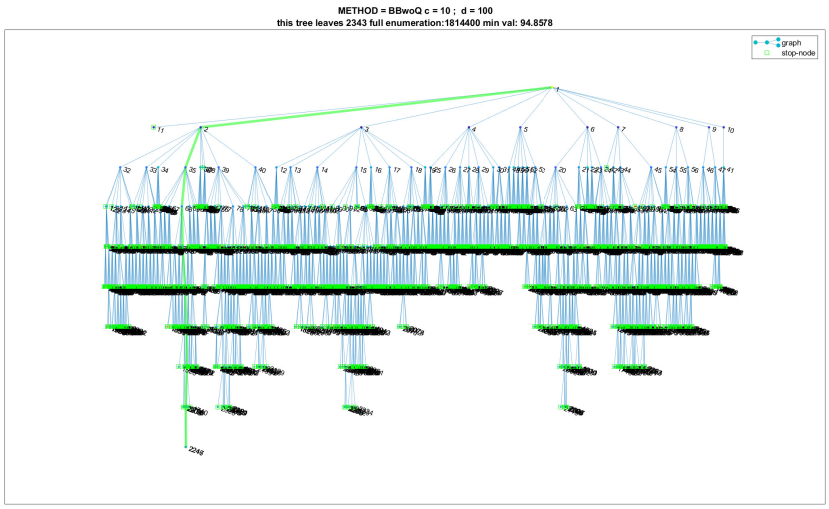


Figure 7: BBwoQ tree for a ten classes problem in $\mathbb{R}^{100}$.

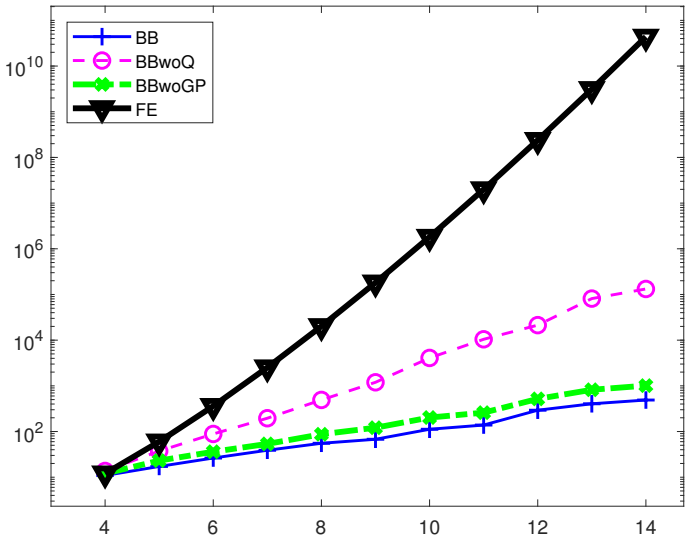


Figure 8: Comparison of the BB, BBwoQ and BBwoGP schemes for different number of classes c.

amining the effect of best rank–one approximation stage (Section 3.4.1) without using the gradient projection stage, that is, comparing BBwoGP and BBwoGPsimpleVEC, we see a minor advantage of using BBwoGP with respect to the number of subproblems solved. The use of gradient projection masks the effect of the rank–one approximation method and the advantage disappears.

In Figure 10 the average CPU time on ten realizations, for each number of classes c using different schemes is given. The tests were performed on Intel(R) i7-8665U CPU @ 1.90GHz, 2112 Mhz, 4 core(s), 8 logical processor(s) machine, using MATLAB R2020a. The suggested BB scheme is the fastest scheme. It is clear from Figure 8 and Figure 10 that the number of nodes has strong effect on the CPU time, although we use different subproblems definition which implies different CPU time process for each subproblem. Table 1 summarizes all the

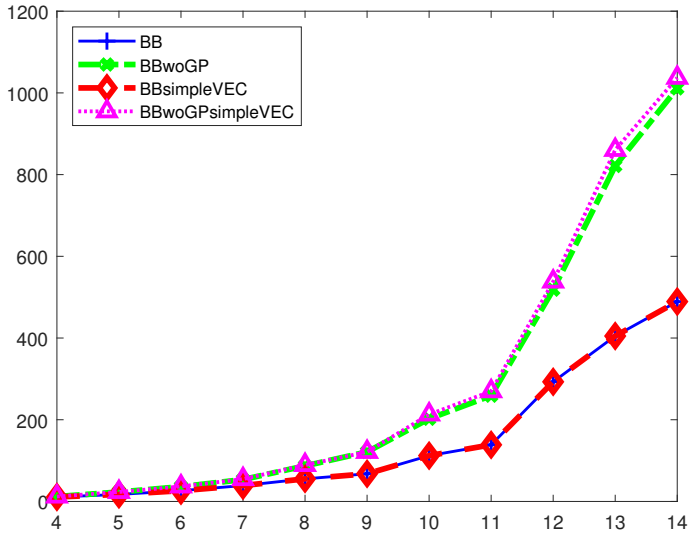


Figure 9: Comparison of the BB, BBwoGP, and BBsimpleVEC, BBwoGPsimpleVEC schemes for different number of classes c.

results.

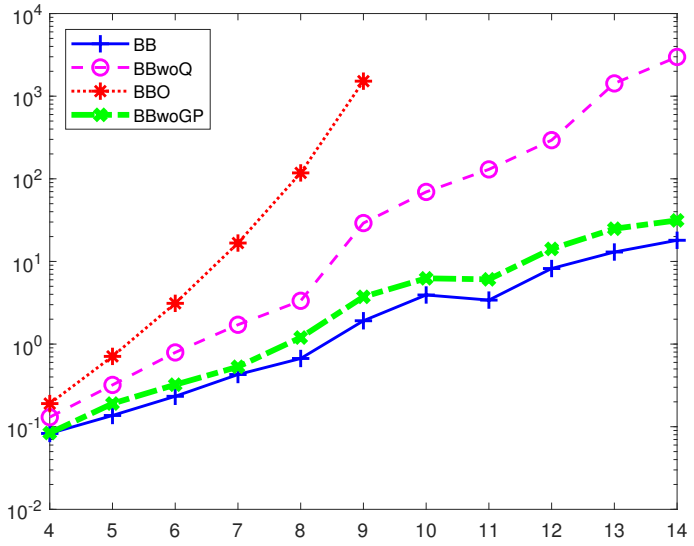


Figure 10: Comparison of the CPU time (log scale) in seconds of BB, BBO, BBwoQ and BBwoGP schemes for different number of classes c.

c	FE	BB		BBwoQ		BBO		BBwoGP	
	NS	NS	sec	NS	sec	NS	sec	NS	sec
3	3	6	0.049	6	0.049	7	0.049	7	0.052
4	12	11	0.083	14	0.13	23	0.19	12	0.084
5	60	17	0.137	37	0.32	103	0.71	23	0.191
6	360	26	0.233	89	0.792	583	3.112	36	0.322
7	2.52e+03	39	0.428	196	1.714	3583	16.704	54	0.531
8	2.02e+04	55	0.669	492	3.341	23002	117.88	87	1.199
9	1.81e+05	68	1.915	1196	29.136	165272	1519.896	121	3.73
10	1.81e+06	112	3.929	4081	69.407	NaN	NaN	202	6.248
11	2e+07	139	3.406	10450	129.994	NaN	NaN	259	6.074
12	2.4e+08	293	8.21	21318	293.604	NaN	NaN	518	14.179
13	3.11e+09	405	13.003	81287	1427.336	NaN	NaN	821	24.885
14	4.36e+10	489	18.033	132003	2983.059	NaN	NaN	1011	31.399
15	6.54e+11	727	22.446	NaN	NaN	NaN	NaN	1489	42.509
16	1.05e+13	1002	34.878	NaN	NaN	NaN	NaN	1953	54.003
17	1.78e+14	1707	60.604	NaN	NaN	NaN	NaN	3917	115.389
18	3.2e+15	3371	127.968	NaN	NaN	NaN	NaN	6147	178.905
19	6.08e+16	3013	111.777	NaN	NaN	NaN	NaN	6058	184.125
20	1.22e+18	5695	229.597	NaN	NaN	NaN	NaN	11122	350.103

Table 1: For each number of classes c, the average number of subproblems (NS) and the average number of CPU time in seconds (sec) over ten random generated datasets are given. Configurations with more than 150,000 nodes were not solved and are marked by '-'.

5 Conclusion

We introduced a variant of the linear discriminant analysis problem on c classes in which the objective is to maximize the minimum distance between classes instead of maximizing the sum of distances between classes, as is typically done. However, this model has the drawback of being highly nonconvex and nonsmooth. We showed a reduction of the problem to a c-dimensional problem consisting of minimizing a quadratic function over a disjoint union of simple convex polyhedra. Each set in the union corresponds to a given order of the decision variables.

We were able to define a branch and bound algorithm that uses subproblems that correspond to specific partial orders on the decision variables. Using the proposed branch and bound scheme, we observed a significant reduction in the size of the enumeration trees as well as in computation times on randomly generated datasets. However, there are limitations to the experiments, such as the use of randomly generated instances rather than real data. The aim of this paper was not to demonstrate the practical superiority of the bounding scheme, but rather to illustrate the potential strength of the bounds and to provide a potential path towards practical efficiency.

Possible future research directions include investigation of the model where the reduced subspace is in higher dimension, as well as models that maximize the l_1–norm of the distances

among all pairs instead of the minimum distance.

References

[1] Amir Beck and Nadav Hallak. On the minimization over sparse symmetric sets: projections, optimality conditions, and algorithms. *Mathematics of Operations Research*, 41(1):196–223, 2016.

[2] Michael J Best and Nilotpal Chakravarti. Active set algorithms for isotonic regression; a unifying framework. *Mathematical Programming*, 47(1):425–439, 1990.

[3] Christopher M. Bishop. *Pattern recognition and machine learning*. springer, 2006.

[4] Jérôme Bolte, Shoham Sabach, and Marc Teboulle. Proximal alternating linearized minimization for nonconvex and nonsmooth problems. *Math. Program.*, 146(1-2, Ser. A):459–494, 2014.

[5] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern classification*. John Wiley & Sons, 2012.

[6] Ronald A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of eugenics*, 7(2):179–188, 1936.

[7] Keinosuke Fukunaga. *Introduction to statistical pattern recognition*. Elsevier, 2013.

[8] Stephen J. Grotzinger and Christoph Witzgall. Projections onto order simplexes. *Applied mathematics and Optimization*, 12(1):247–270, 1984.

[9] Christoph Helmberg, Franz Rendl, Robert J Vanderbei, and Henry Wolkowicz. An interior-point method for semidefinite programming. *SIAM Journal on Optimization*, 6(2):342–361, 1996.

[10] Seung-Jean Kim, Alessandro Magnani, and Stephen Boyd. Robust fisher discriminant analysis. *Advances in neural information processing systems*, 18, 2005.

[11] Eugene L. Lawler and David E. Wood. Branch-and-bound methods: A survey. *Operations research*, 14(4):699–719, 1966.

[12] Jiri Matousek and Bernd Gärtner. *Understanding and using linear programming*. Springer Science & Business Media, 2007.

[13] Jorge J. Moré. Generalizations of the trust region problem. *Optimization methods and Software*, 2(3-4):189–209, 1993.

[14] Thanh T. Ngo, Mohammed Bellalij, and Yousef Saad. The trace ratio optimization problem. *SIAM review*, 54(3):545–569, 2012.

[15] Christos H. Papadimitriou and Kenneth Steiglitz. *Combinatorial optimization: algorithms and complexity*. Courier Corporation, 1998.

[16] Siegfried Schaible. Fractional programming. In *Handbook of global optimization*, pages 495–608. Springer, 1995.

[17] Naum Z Shor. Quadratic optimization problems. *Soviet Journal of Computer and Systems Sciences*, 25:1–11, 1987.

[18] Henry Wolkowicz, Romesh Saigal, and Lieven Vandenberghe, editors. *Handbook of semidefinite programming*, volume 27 of *International Series in Operations Research & Management Science*. Kluwer Academic Publishers, Boston, MA, 2000. Theory, algorithms, and applications.