An Extension of Linear Discriminant Analysis for Multi-Label Data

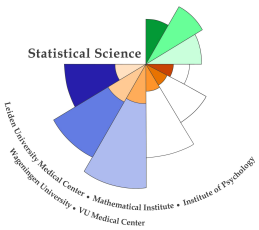
Author: Britt Broere (s2388367)

Thesis advisor: Prof. Dr. M. J. de Rooij

MASTER THESIS

Defended on Month Day, 2021

Specialization: Data Science



STATISTICAL SCIENCE
FOR THE LIFE AND BEHAVIOURAL SCIENCES

Abstract

Linear Discriminant Analysis (LDA) is a widely used tool for classification and dimension reduction. However, LDA can only be directly applied to single-labelled data and not to multi-labelled data. This work proposes an extension to LDA called Joint Linear Discriminant Analysis (JLDA) which can be applied to multi-labelled data. First, the data are transformed into single-labelled multi-class data through a method called label powerset. This method creates a new indicator matrix $\mathbf{Y}$, containing the observed classes for the objects. Because $\mathbf{Y}$ is a cross classification of different response variables, a model structure can be implied on the class coordinates matrix $\mathbf{G}$. This structure, a design matrix $\mathbf{Z}$, links the predictors to the profile of response variables and reduces the number of parameters. In addition to JLDA, this work also proposes a regularized version of JLDA using the L_1 penalty. The aim of this regularized version is to automatically obtain the optimal dimensionality M.

Both JLDA and regularized JLDA are presented and applied to an existing data set. Then, a simulation study was performed to assess the performance of regularized JLDA. Because the true dimensionality of data is often unknown, it was investigated whether regularized JLDA is able to select the correct dimensionality. This was done by means of a simulation study. Data with known dimensionality were generated and how often the regularized JLDA model was able to select the correct dimensionality was used as a measure of performance. Multiple design factors were considered to study the performance of regularized JLDA in different scenarios. Design factors include usage of the one-standard-error rule, method of calculating the prediction error, true dimensionality, sample size, dependence of the predictors, and order of the design matrix $\mathbf{Z}$.

Results showed that regularized JLDA is able to select the correct dimensionality, provided that the sample size is large enough, the one-standard-error rule is used when selecting the hyperparameter λ, and the marginal prediction error is used.

Contents

1

Introduction

Classification is an important part of every day life. Our minds identify objects on a daily basis. For example, when we see an object flying through the sky, we identify this object as either a bird, a plane, a drone, a helicopter, etc. Besides classification by our cognitive systems, there are also many machines and systems that work with classification (Duda et al., 2001). For example, the classification of email messages as spam or not spam. In its most simple form, the classification problem is a single-labelled classification problem. This implies that there is only a single response variable or a single label assigned to an object (Tsoumakas & Katakis, 2009). In a classification problem, we wish to assign an object to one of the C different classes. When $C = 2$, it is also called binary classification. In this case, an object belongs to one of two possible classes. Spam filtering is an example of binary classification, where, for instance, a spam email is coded as 1 and a non-spam email is coded as 0. When $C \geq 3$, it is also called multi-class classification. Assigning objects to one of the C different classes can be done by defining discriminant functions $\delta_c(x)$ for each class and setting decision boundaries between the classes, which is the line that separates, for example, class c from class l. This boundary is defined by the set of points {x: $\delta_c(x)$ - $\delta_l(x) = 0$}. A new object is assigned to the class for which its discriminant function has the largest value. An alternative way of assigning an object is by modelling the posterior probabilities $\Pr(C = c|X = x)$. A new object is assigned to the class with the highest posterior probability (Fukunaga, 2013; Hastie et al., 2009). A classification problem where the outcome variable is present in the learning process is called a *supervised learning* problem. A well-known *supervised* linear method for classification is linear discriminant analysis (LDA).

As mentioned, LDA is a widely used tool for classification (Hastie et al., 1995). The case where there are two classes, $C = 2$, was first introduced by R. Fisher and therefore it is also known as Fishers Linear Discriminant Analysis (Fisher, 1936). There are multiple ways to formulate the LDA classification rule. One of the formulations results from using the optimal scoring approach (Clemmensen et al., 2011). This approach is defined by minimizing the following loss function:

$$L(\mathbf{B}, \mathbf{G}) = \| \mathbf{XB} - \mathbf{YG} \|^2, \tag{1.1}$$

where $\mathbf{X}$ is a centered data matrix containing the responses of objects on certain predictor variables, $\mathbf{B}$ is a coefficient matrix, $\mathbf{Y}$ is an indicator matrix which indicates to which class an object belongs, and $\mathbf{G}$ is a matrix containing the class coordinates. The matrix $\mathbf{XB}$ contains the scores of the objects, which are also referred to as the discriminant scores. The matrix $\mathbf{G}$ is computed as:

$$\hat{\mathbf{G}} = \mathbf{D}_y^{-1}\mathbf{Y}^\top\mathbf{XB}, \tag{1.2}$$

where $\mathbf{D}_y = \mathbf{Y}^\top\mathbf{Y}$ is a diagonal matrix which contains the frequencies of how often each class occurs. More specifically, $\mathbf{G}$ can be represented as $\mathbf{G} = [\mathbf{g}_1, \ldots, \mathbf{g}_c]^\top$ and it is implied that the cth row $\mathbf{g}_c^\top$ is the averaged vector of the discriminant score vectors for the objects belonging to class c. Equation (1.1) could also be interpreted as a homogeneity approach to LDA. This implies that the discriminant score vector for an object ($\mathbf{x}_i\mathbf{B}$) should be homogeneous to the optimal class coordinates for the class ($\mathbf{g}_c$) to which this object belongs (Adachi, 2016). This homogeneity approach leads to the assumption that a new object is assigned to a class based on minimum distance classification. In the case of $C = 2$, this implies that for a new object, $\mathbf{x}_{new}$, two distances are calculated. The object $\mathbf{x}_{new}$ is then classified into the class for which the distance is smallest:

$$\min_{1\leq c\leq C} \parallel \mathbf{x}_{new}\mathbf{B} - \mathbf{g}_c \parallel . \tag{1.3}$$

Despite the fact that LDA has been originally developed for binary class classification, there are extensions of the method that allow multi-class classification. However, there are only a few extensions, as far as we know, that allow multi-labelled data.

Multi-label or multi-outcome data are data that have more than one response variable. In a multi-label classification problem, an object can be assigned to more than one class. Examples of multi-label classification concern classifying newspaper articles or identifying drug use in people. A newspaper article could be classified as both 'Entertainment' and 'Political' and a person can use both cannabis and cocaine. More specifically, classes are not mutually exclusive. The methods for multi-label classification are divided into two main groups: 1) problem transformation and 2) algorithm adaptation (Tsoumakas & Katakis, 2009). Methods that transform the multi-label classification problem into one or several single-label classification problems, belong to the problem transformation category. On the other hand, methods that alter the learning algorithm in such a way that it is able to handle multi-label data directly, belong to the algorithm adaptation category.

To our knowledge, there are relatively few techniques that allow applying LDA to a multi-label classification problem. The problem arises that when objects can have multiple classes, the decision regions can overlap. There is a method called Multi-label Linear Discriminant Analysis (MLDA) which adapts the formulation of LDA and incorporates class correlations (Wang et al., 2010). Another article proposes a computationally efficient LDA algorithm for multi-labelled data, which is called LDA-ALL (Park & Lee, 2008). In LDA-ALL, a data set is composed where an object with multiple class labels appears several times. Then LDA is applied to this new data set, in the same

way as it would have been applied in a single-labelled problem. However, there is a very obvious and simple way to extent LDA for a multi-label classification problem. Namely, transform the multi-label data set into a single multi-class data set. This can be done by considering each unique outcome combination as an unique class (Pushpa & Karpagavalli, 2017).

This thesis proposes an extension of LDA, namely Joint Linear Discriminant Analysis (JLDA), which is able to handle multi-labelled data. This method transforms multi-labelled data into single-labelled data and links the predictors to the profile of response variables. Therefore it can be considered to be a problem transformation method. From the R different response variables, JLDA creates a single new response variable through the cross classification of the original R response variables.

A disadvantage of creating a single new response variable from the cross classification of the original R response variables is that as the number of response variables increases, the number of distinct outcome combinations also increases. In fact, for R dichotomous response variables, there are 2^R number of combinations. Consequently, there might be some classes with only a few instances and thus the performance of the classifier may deteriorate. An advantage of JLDA is the possibility of dimension reduction. Determining the optimal dimensionality could be done through K-fold cross-validation. When using this method, multiple models with different dimensionalities are fitted and the optimal dimensionality is selected by choosing the model whose prediction error is lowest. However, this needs to be done manually, which is time consuming.

Therefore, we propose a regularized version of JLDA using the L_1 penalty (Tibshirani, 1996). By adding this penalty, the discrete dimension selection problem is transformed into a continuous selection problem. The penalty is applied to the class coordinates matrix $\mathbf{G}$. It creates sparsity by setting some values to zero and thereby achieves dimensionality reduction. A major advantage of regularized JLDA is that the dimensionality selection is now automatic instead of manual.

This thesis proposes an extension of LDA that is able to deal with multi-label data and is called Joint Linear Discriminant Analysis (JLDA). Since the dimensionality of the JLDA model can be high, it is desirable to perform dimension reduction. Therefore, a regularized version of JLDA is proposed. The regularization uses the L_1 penalty to select the optimal dimensionality automatically. This thesis has three aims. The first aim is to propose JLDA. The second aim is to propose an algorithm for regularized JLDA. The last aim is to evaluate the performance of regularized JLDA. The evaluation of the performance of regularized JLDA is done by the means of a simulation study. Data sets with known dimensionality are generated and regularized JLDA is applied to these new data sets. Then, it is recorded how many times the regularized JLDA model selects the correct dimensionality.

2

An extension of Linear Discriminant Analysis

In this section, Linear Discriminant Analysis (LDA) is described. Next, an extension of LDA for multi-label data, called Joint Linear Discriminant Analysis (JLDA) is presented together with a regularized version of JLDA.

2.1 Linear Discriminant Analysis

Let $\mathbf{X}$ be a $n \times P$ data matrix containing responses of objects $i = 1, \ldots, n$ on $p = 1, \ldots, P$ predictor variables. Assume that $\mathbf{X}$ is centered such that $\mathbf{1}^\top \mathbf{X} = \mathbf{0}$. Each of the n observations falls in one of C classes. Let vector $\mathbf{y^*}$ be of length n and containing the observed classes for each object $i = 1, \ldots, n$. From this vector an indicator matrix $\mathbf{Y}$ of size $n \times C$ is created.

The homogeneity approach to Linear Discriminant Analysis (LDA) is defined by minimizing the following loss function:

$$L(\mathbf{B}, \mathbf{G}) = \| \mathbf{XB} - \mathbf{YG} \|^2, \tag{2.1}$$

as described by Adachi (2016). The matrix $\mathbf{B}$ is of size $P \times M$, where M is the number of dimensions and contains the regression coefficients. The maximum dimensionality in LDA is $\min(P, C-1)$. The matrix $\mathbf{G}$ is of size $C \times M$ and contains the optimal class coordinates. However, finding a solution for $\mathbf{B}$ and $\mathbf{G}$ that minimizes the loss function would result in setting both to 0. In order to avoid such a solution, the loss is minimized subject to the constraint $\frac{1}{n}\mathbf{B}^\top \mathbf{X}^\top \mathbf{XB} = \mathbf{I}_M$. The matrix $\mathbf{G}$ is computed as

$$\hat{\mathbf{G}} = \mathbf{D}_y^{-1} \mathbf{Y}^\top \mathbf{XB}, \tag{2.2}$$

where $\mathbf{D}_y = \mathbf{Y}^\top \mathbf{Y}$ is a diagonal matrix containing the frequencies that indicate how often each class occurs.

Using this estimate of $\mathbf{G}$ in the loss function in Equation (2.1), the following loss function is obtained:

$$L(\mathbf{B}) = \| \mathbf{XB} - \mathbf{YD}_y^{-1} \mathbf{Y}^\top \mathbf{XB} \|^2 . \tag{2.3}$$

This function can be rewritten as:

$$L(\mathbf{B}) = \text{tr}\mathbf{B}^\top\mathbf{X}^\top\mathbf{X}\mathbf{B} + \text{tr}\mathbf{B}^\top\mathbf{X}^\top\mathbf{Y}\mathbf{D}_y^{-1}\mathbf{Y}^\top\mathbf{Y}\mathbf{D}_y^{-1}\mathbf{Y}^\top\mathbf{X}\mathbf{B} - 2\text{tr}\mathbf{B}^\top\mathbf{X}^\top\mathbf{Y}\mathbf{D}_y^{-1}\mathbf{Y}^\top\mathbf{X}\mathbf{B} \tag{2.4}$$

$$= \text{tr}\mathbf{B}^\top\mathbf{X}^\top\mathbf{X}\mathbf{B} + \text{tr}\mathbf{B}^\top\mathbf{X}^\top\mathbf{Y}\mathbf{D}_y^{-1}\mathbf{Y}^\top\mathbf{X}\mathbf{B} - 2\text{tr}\mathbf{B}^\top\mathbf{X}^\top\mathbf{Y}\mathbf{D}_y^{-1}\mathbf{Y}^\top\mathbf{X}\mathbf{B} \tag{2.5}$$

$$= nM - \text{tr}\mathbf{B}^\top\mathbf{X}^\top\mathbf{Y}\mathbf{D}_y^{-1}\mathbf{Y}^\top\mathbf{X}\mathbf{B}. \tag{2.6}$$

Minimizing this function equals maximizing $\text{tr}\mathbf{B}^\top\mathbf{X}^\top\mathbf{Y}\mathbf{D}_y^{-1}\mathbf{Y}^\top\mathbf{X}\mathbf{B}$ over $\mathbf{B}$.

In order to obtain the solution for $\mathbf{B}$ and to minimize the loss function, the generalized singular value decomposition (GSVD) is used. Let $\mathbf{V} = \frac{1}{n}\mathbf{X}^\top\mathbf{X}$ be a $P \times P$ symmetric positive definite matrix, containing the total variance. Let $\mathbf{M} = \mathbf{X}^\top\mathbf{Y}\mathbf{D}_y^{-1}\mathbf{Y}^\top\mathbf{X}$ be the $P \times P$ symmetric and non-negative definite matrix with rank r, containing the between-variance. Let $\mathbf{U} = \mathbf{X}^\top\mathbf{Y}\mathbf{D}_y^{-\frac{1}{2}}$ and thus $\mathbf{M} = \mathbf{X}^\top\mathbf{Y}\mathbf{D}_y^{-1}\mathbf{Y}^\top\mathbf{X} = \mathbf{U}\mathbf{U}^\top$. Then we have to

$$maximize\ \text{tr}\mathbf{B}^\top\mathbf{M}\mathbf{B} \tag{2.7}$$

over $\mathbf{B}$ subject to the constraint $\mathbf{B}^\top\mathbf{V}\mathbf{B} = \mathbf{I}_M$. This can be solved through the singular value decomposition

$$\mathbf{U}^\top\mathbf{V}^{-\frac{1}{2}} = \mathbf{P}\Phi\mathbf{Q}^\top. \tag{2.8}$$

The solution for $\mathbf{B}$ is obtained as

$$\mathbf{B} = \mathbf{V}^{-\frac{1}{2}}\mathbf{Q}_M, \tag{2.9}$$

where $\mathbf{Q}_M$ are the M singular vectors belonging to the M largest singular values. A proof can be found in chapter 15 of Adachi (Adachi, 2016). Both $\mathbf{D}_y^{-\frac{1}{2}}$ and $\mathbf{V}^{-\frac{1}{2}}$ can be obtained by the eigenvalue decomposition of $\mathbf{D}_y$ and $\mathbf{V}$, respectively.

2.2 Joint Linear Discriminant Analysis

The procedure described in section 2.1 applies to situations where there is only one outcome variable or in other words, single-labelled data. However in fields such as psychology, it may occur that there are multiple outcome variables and thus that the data is multi-labelled. For example, the drug consumption database, available from the UCI depository, contains records on the use of 18 different substances. For each of the respondents, 12 attributes are known such as age, gender, and certain personality characteristics. Participants were questioned concerning their use of 18 legal and illegal drugs (see section 3.1 for a more detailed description of the drug consumption data set).

Using this data set as an example, there are now 18 response variables instead of a single response variable. Therefore the classification problem is no longer single-labelled but it becomes a multi-label classification problem. An important fact is that these classes are not mutually exclusive, a participant can use both alcohol and cocaine. Since LDA does not work for multi-label data, the data have to be transformed. Through a method called label powerset, the multi-label data set is transformed into a single

multi-class data set. This is done by considering each unique outcome combination as an unique class (Pushpa & Karpagavalli, 2017). Therefore, the problem becomes a single-labelled multi-class classification problem and can be solved with LDA. We call this problem transformation method Joint Linear Discriminant Analysis (JLDA).

Again, let $\mathbf{X}$ be a $n \times P$ data matrix containing responses of objects $i = 1, \ldots, n$ on $p = 1, \ldots, P$ predictor variables and assume that $\mathbf{X}$ is centered such that $\mathbf{1}^\top \mathbf{X} = \mathbf{0}$. From the R different response variables, a single new response variable $\mathbf{y^*}$ is made as the cross classification of the original R response variables. This new response variable has $C_j = \prod_r C_r$ categories, where C_r is the number of categories in response variable R. To illustrate, suppose we have five dichotomous response variables, thus $R = 5$ and $C_r = 2$ for all R. We have indicator matrix $\mathbf{Y^*}$, of size $n \times R$, combining each of the five response variables,

$$\mathbf{Y^*} = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 1 & 1 & 0 \end{bmatrix}.$$

Then the new response variable $\mathbf{y^*}$ becomes

$$\mathbf{y^*} = \begin{bmatrix} 10110 \\ 01101 \\ 11110 \\ 01110 \\ \vdots \\ 00110 \end{bmatrix},$$

where '10110' is a new category indicating a positive response (for example, 'yes') on the first, third, and fourth response variables. The new categories, such as '10110', are also called response profiles. The new response variable is represented by an indicator matrix $\mathbf{Y}$ of size $n \times C_j$. It should be noted that as the number of response variables increases, the number of distinct outcome combinations also increases with only a few instances per class. Consequently, the performance of the classifier may deteriorate. Also, the analysis might be computationally challenging because the matrix $\mathbf{Y}$ can become quite large. However, because the indicator matrix $\mathbf{Y}$ is a cross classification of R original response variables, a model structure on the class coordinates $\mathbf{G}$ in terms of main effects and associations can be implied. Therefore, a design matrix $\mathbf{Z}$ is defined that imposes a structure on $\mathbf{G}$. To illustrate, suppose we have three dichotomous response variables,

then this matrix $\mathbf{Z}$ might, for example, only code the main effects as shown by $\mathbf{Z}^{(1)}$.

$$\mathbf{Z}^{(1)} = \begin{array}{c} \begin{matrix} \text{Intercept} & R_1 & R_2 & R_3 \end{matrix} \\ \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} \end{array}$$

In this case R_1, R_2, and R_3 represent the first, second, and third response variable respectively. Moreover, this matrix might also code the main effects and the two-way associations as shown by $\mathbf{Z}^{(2)}$.

$$\mathbf{Z}^{(2)} = \begin{array}{c} \begin{matrix} \text{Intercept} & R_1 & R_2 & R_3 & R_1:R_2 & R_1:R_3 & R_2:R_3 \end{matrix} \\ \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \end{array}$$

Here, the two-way associations are also included, indicated by $R_1 : R_2$, $R_1 : R_3$, and $R_2 : R_3$. In the case of a design matrix $\mathbf{Z}$ that codes both the main effects and associations, the solution of the model then indicates how a predictor variable affects the association between two response variables. Also, the matrix $\mathbf{Z}$ reduces the number of parameters. The class coordinates matrix is now defined as $\mathbf{G} = \mathbf{Z}\mathbf{B}_g$. The matrix $\mathbf{B}_g$ is of size $\#columns\,\mathbf{Z} \times M$ and contains the optimal response variable coordinates. The maximum dimensionality in JLDA is $\min(P, \#columns\,\mathbf{Z})$. The loss function then becomes

$$L(\mathbf{B}, \mathbf{B}_g) = \| \mathbf{X}\mathbf{B} - \mathbf{W}\mathbf{B}_g \|^2 \tag{2.10}$$

where $\mathbf{W} = \mathbf{Y}\mathbf{Z}$. The optimal solution for $\mathbf{B}_g$ is given by

$$\hat{\mathbf{B}_g} = \left(\mathbf{W}^\top \mathbf{W}\right)^{-1} \mathbf{W}^\top \mathbf{X}\mathbf{B}. \tag{2.11}$$

Then similar to the linear discriminant case of one response variable, the solution can be found by a generalized singular value decomposition of

$$\mathbf{U}^\top \mathbf{V}^{-\frac{1}{2}} = \mathbf{P}\Phi\mathbf{Q}^\top, \tag{2.12}$$

where $\mathbf{U} = \mathbf{X}^\top \mathbf{Y}\mathbf{Z}\mathbf{D}_{zy}^{-\frac{1}{2}}$ with $\mathbf{D}_{zy} = \mathbf{Z}^\top \mathbf{D}_y \mathbf{Z}$ and $\mathbf{V} = \frac{1}{n}\mathbf{X}^\top \mathbf{X}$. The solution is then obtained by

$$\mathbf{B} = \mathbf{V}^{-\frac{1}{2}} \mathbf{Q}_M. \tag{2.13}$$

Again, the eigenvalue decomposition of $\mathbf{D}_{zy}$ and $\mathbf{V}$ is required to obtain $\mathbf{D}_{zy}^{-\frac{1}{2}}$ and $\mathbf{V}^{-\frac{1}{2}}$.

2.3 Goodness-of-fit test for dimensionality reduction

An advantage of LDA is that the number of dimensions can be reduced (Shu et al., 2015). We would need a goodness-of-fit (GOF) test or a badness-of-fit (BOF) test to choose the optimal dimensionality. The minimum loss obtained in Equation (2.3) might be an appropriate BOF test. However, the loss function seems to have a strange property:

$$L(\mathbf{B})_M \geq L(\mathbf{B})_{M-1}. \tag{2.14}$$

That is, the loss increases as the dimensionality M increases (Adachi, 2004). This can be shown by substituting Equation (2.9) into Equation (2.3). The following loss function is obtained:

$$L(*)_M = \| \mathbf{X}\mathbf{V}^{-\frac{1}{2}}\mathbf{Q}_M - \mathbf{Y}\mathbf{D}_y^{-1}\mathbf{Y}^\top \mathbf{X}\mathbf{V}^{-\frac{1}{2}}\mathbf{Q}_M \|^2 \tag{2.15}$$

$$\begin{aligned} &= \mathrm{tr}\mathbf{Q}_M^\top (\mathbf{V}^{-\frac{1}{2}})^\top \mathbf{X}^\top \mathbf{X}\mathbf{V}^{-\frac{1}{2}}\mathbf{Q}_M + \mathrm{tr}\mathbf{Q}_M^\top (\mathbf{V}^{-\frac{1}{2}})^\top \mathbf{X}^\top \mathbf{Y}\mathbf{D}_y^{-1}\mathbf{Y}^\top \mathbf{Y}\mathbf{D}_y^{-1}\mathbf{Y}^\top \mathbf{X}^\top \mathbf{V}^{-\frac{1}{2}}\mathbf{Q}_M \\ &\quad - 2\mathrm{tr}\mathbf{Q}_M^\top (\mathbf{V}^{-\frac{1}{2}})^\top \mathbf{X}^\top \mathbf{Y}\mathbf{D}_y^{-1}\mathbf{Y}^\top \mathbf{X}\mathbf{V}^{-\frac{1}{2}}\mathbf{Q}_M \end{aligned} \tag{2.16}$$

$$\begin{aligned} &= \mathrm{tr}\mathbf{Q}_M^\top (\mathbf{V}^{-\frac{1}{2}})^\top \mathbf{X}^\top \mathbf{X}\mathbf{V}^{-\frac{1}{2}}\mathbf{Q}_M + \mathrm{tr}\mathbf{Q}_M^\top (\mathbf{V}^{-\frac{1}{2}})^\top \mathbf{X}^\top \mathbf{Y}\mathbf{D}_y^{-1}\mathbf{Y}^\top \mathbf{X}\mathbf{V}^{-\frac{1}{2}}\mathbf{Q}_M \\ &\quad - 2\mathrm{tr}\mathbf{Q}_M^\top (\mathbf{V}^{-\frac{1}{2}})^\top \mathbf{X}^\top \mathbf{Y}\mathbf{D}_y^{-1}\mathbf{Y}^\top \mathbf{X}\mathbf{V}^{-\frac{1}{2}}\mathbf{Q}_M \end{aligned} \tag{2.17}$$

$$= nM - \mathrm{tr}\mathbf{Q}_M^\top (\mathbf{V}^{-\frac{1}{2}})^\top \mathbf{X}^\top \mathbf{Y}\mathbf{D}_y^{-1}\mathbf{Y}^\top \mathbf{X}\mathbf{V}^{-\frac{1}{2}}\mathbf{Q}_M \tag{2.18}$$

$$= nM - \sum_{m=1}^{M} \psi_m^2, \tag{2.19}$$

where ψ_m are the singular values of Φ. Now, the same can be done for a lower dimensionality and the following loss function is obtained:

$$L(*)_{M-1} = \| \mathbf{X}\mathbf{V}^{-\frac{1}{2}}\mathbf{Q}_{M-1} - \mathbf{Y}\mathbf{D}_y^{-1}\mathbf{Y}^\top \mathbf{X}\mathbf{V}^{-\frac{1}{2}}\mathbf{Q}_{M-1} \|^2 \tag{2.20}$$

$$= nM - n - \sum_{m=1}^{M-1} \psi_m^2. \tag{2.21}$$

The inequality from Equation (2.14) follows from the fact that $L(*)_M - L(*)_{M-1} = n - \psi_M^2 \geq 0$ since the singular values of Φ are in decreasing order, thus $\psi_M^2 \leq \psi_1^2$, and $\psi_1^2 = \mathbf{B}_{M=1}^\top \mathbf{X}^\top \mathbf{Y}\mathbf{G}_{M=1}$.

The same strange property also applies to the loss function of JLDA:

$$L(\mathbf{B}, \mathbf{B}_g)_M \geq L(\mathbf{B}, \mathbf{B}_g)_{M-1}. \tag{2.22}$$

From this inequality it follows that $L(*,*)_M - L(*,*)_{M-1} = n - \psi_M^2 \geq 0$ since $\psi_M^2 \leq \psi_1^2$ and $\psi_1^2 = \mathbf{B}_{M=1}^\top \mathbf{X}^\top \mathbf{Y}\mathbf{Z}\mathbf{B}_{g_{M=1}}$.
The properties of GOF and BOF tests can be classified into two classes. The first class includes tests whose values increase with increasing values for M. The second class includes tests whose minima are considered to be the true dimensionality (Adachi, 2004). The inequalities in Equations (2.14) and (2.22) are not in line with the classes of these properties. Therefore, the loss function cannot be used in order to determine the dimensionality.

2.4 Automatic dimensionality selection by Regularized Joint Linear Discriminant Analysis

As explained in the previous section, the loss functions from Equations (2.1) and (2.10) cannot be used to determine the optimal dimensionality. Determining the optimal dimensionality could be done through a method called K-fold cross-validation, which will be further discussed later on in this section. When using K-fold cross-validation, multiple models with different values for M are fitted and the optimal dimensionality is selected by choosing the model whose prediction error is lowest. However, fitting these models needs to be done manually and thus, the dimension selection problem is still discrete. Therefore, a regularized version of Joint Linear Discriminant Analysis is proposed using the L_1 penalty (Tibshirani, 1996). Adding this penalty transforms the discrete dimension selection problem into a continuous problem and selects the dimensionality automatically. Also, the addition of a penalty might be better for interpretability and may reduce overfitting (Clemmensen et al., 2011). The idea is to apply the penalty to the class point matrix $\mathbf{B}_g$. The penalty creates sparsity by setting some values to zero and thus achieves dimensionality reduction. The loss function that has to be minimized then becomes

$$L(\mathbf{B}, \mathbf{B}_g) = \| \mathbf{XB} - \mathbf{WB}_g \|^2 + \lambda |\mathbf{B}_g|, \tag{2.23}$$

where $|\mathbf{B}_g|$ is the sum of the absolute values of $\mathbf{B}_g$ and λ is a tuning parameter.

In order to minimize Equation (2.23), instead of using the generalized singular value decomposition as described above, an alternating least squares algorithm is used to obtain the solution. In order to solve this problem, the loss function has to be partially optimized with respect to $\mathbf{B}$ and then with respect to $\mathbf{B}_g$. The derivative of the loss function with respect to $\mathbf{B}$ is taken and set to 0. Then the update for $\mathbf{B}$ is obtained as

$$\mathbf{B} = \left(\mathbf{X}^\top \mathbf{X}\right)^{-1} \mathbf{X}^\top \mathbf{W}\mathbf{B}_g \tag{2.24}$$

$$= \left(\mathbf{X}^\top \mathbf{X}\right)^{-1} \mathbf{X}^\top \mathbf{Y}\mathbf{Z}\mathbf{B}_g \tag{2.25}$$

$$= \left(\mathbf{X}^\top \mathbf{X}\right)^{-1} \mathbf{X}^\top \mathbf{Y}\mathbf{G}. \tag{2.26}$$

Again, matrix $\mathbf{B}$ needs to be rescaled such that $\frac{1}{n}\mathbf{B}^\top \mathbf{X}^\top \mathbf{X}\mathbf{B} = \mathbf{I}_M$. This is done through the singular value decomposition

$$\mathbf{X}\mathbf{B} = \mathbf{U}\Phi\mathbf{V}^\top. \tag{2.27}$$

The matrix $\mathbf{B}$ is then rescaled as

$$\mathbf{B} = \sqrt{n}\mathbf{B}\mathbf{V}(\Phi)^{-1}. \tag{2.28}$$

This procedure is repeated with respect to $\mathbf{B}_g$. The update is obtained as

$$\mathbf{B}_g = S\left(\left(\mathbf{W}^\top \mathbf{W}\right)^{-1} \mathbf{W}^\top \mathbf{X}\mathbf{B}, \lambda\right), \tag{2.29}$$

where $S(z, \gamma)$ is the soft-thresholding operator (Friedman et al., 2010) with value

$$sign(z)(|z| - \gamma)_+ = \begin{cases} z + \gamma, & \text{if } z > 0 \text{ and } \gamma < |z| \\ z - \gamma, & \text{if } z < 0 \text{ and } \gamma < |z| \\ 0, & \text{if } \gamma \geq |z| \end{cases} \tag{2.30}$$

Updating $\mathbf{B}$ and $\mathbf{B}_g$ defines one iteration of the algorithm. These matrices are updated until convergence or until a maximum of iterations is exceeded.

The optimal value of the tuning parameter λ in Equations (2.23) and (2.29) is determined using K-fold cross-validation. This is one of the most useful tools for selecting hyperparameters such as, in this case, λ (Zhang, 1993). We wish to select the tuning parameter that minimizes the cross-validation prediction error. Cross-validation works by splitting the data into K approximately equal-sized folds. For the kth fold, the model is fitted to the other $K - 1$ folds. Then this fitted model is used to predict the responses of the retained kth part of the data. When doing so, an estimate for the prediction error is obtained as:

$$\mathrm{PE}_k = \frac{1}{n_k} \sum_{i}^{n_k} I(y_i \neq \hat{y}_i), \tag{2.31}$$

where n_k is the sample size of the Kth fold, y_i the true outcome and $\hat{y}_i$ the predicted outcome. We do this for all folds, $k = 1, \ldots K$, which leads to K estimates for the prediction error. These estimates are averaged and this leaves us with a final estimate of the prediction error. The choice of K and the bias-variance trade-off are closely connected. When the value for K is very big, the cross-validation estimator has a low bias for the true prediction error. However, it can have high variance because the training sets are very similar. On the other hand, when K is very small, the estimator has lower variance but might have a higher bias. Logically, a good compromise between the bias and the variance is needed, hence the name bias-variance trade-off. Typically five- or tenfold cross-validation is regarded as such a good compromise (Breiman & Spector,

1992; Kohavi, 1995). Here, $K = 5$ is used and thus 5-fold cross-validation is used to determine the optimal value of λ.

Functions that perform JLDA (`jlda`) and regularized JLDA (`rjlda`) are implemented in R and can be found in Appendix A. The function that performs cross-validated regularized JLDA (`cv.rjlda`) can also be found in Appendix A. Furthermore, some additional functions were written to be used in the `jlda`, `rjlda`, and `cv.rjlda` functions. These additional functions are needed in order for the three main functions to work and can be found in Appendix B.

3

Empirical Application

In this section, the use of JLDA and regularized JLDA is demonstrated through the application to the existing drug consumption data set. For both analyses, a distinction is made between a model that only takes the main effects into account and a model that takes both main effects and two-way interactions into account. The outcome will be evaluated in terms of five-fold cross-validated prediction error (PE). However, there are two ways for calculating this prediction error. The first is based on the entire response profile and will be referred to as the joint prediction error:

$$\text{PE}_k = \frac{1}{n_k} \sum_{i}^{n_k} I(y_i \neq \hat{y}_i), \tag{3.1}$$

where n_k is the sample size of the kth fold, y_i the true response profile and $\hat{y}_i$ the predicted response profile. Then the joint prediction error is obtained as:

$$\text{PE} = \frac{1}{K} \sum_{k}^{K} \text{PE}_k. \tag{3.2}$$

The second way is based on the five separate response variables and will be referred to as the marginal prediction error:

$$\text{PE}_r = \frac{1}{n_k} \sum_{i}^{n_k} I(y_{ir} \neq \hat{y}_{ir}), \tag{3.3}$$

where r indicates the r-th response variable, n_k is the sample size of the kth fold, y_{ir} the true outcome and $\hat{y}_{ir}$ the predicted outcome. Then the prediction error per fold is obtained as:

$$\text{PE}_k = \frac{1}{R} \sum_{r}^{R} \text{PE}_r. \tag{3.4}$$

Then the marginal prediction error is obtained as:

$$\text{PE} = \frac{1}{K} \sum_{k}^{K} \text{PE}_k. \tag{3.5}$$

3.1 The data

The data set contains information on the consumption of 18 central nervous system psychoactive drugs. Elaine Fehrman has collected the data using an online survey methodology (Fehrman et al., 2017). The study initially recruited 2015 participants over a one-year recruitment period. After data cleansing however, there were 1885 subjects (943 male, 942 female) included in the study.

The goal of the study was to predict the risk of drug consumption for each individual according to their personality traits. In order to measure the personality traits of the participants, the Revised NEO Five-Factor Inventory (NEO-FFI-R) questionnaire, the Barratt Impulsiveness Scale (BIS-11) and the Impulsiveness Sensation Seeking Scale (ImpSS) were used. The NEO-FFI-R is a 60-item questionnaire containing statements which apply to five personality domains. These five domains are Neuroticism (N), Extraversion (E), Openness to Experience (O), Agreeableness (A) and Conscientiousness (C). The scale's items are scored on a five-point Likert scale. The BIS-11 is a 30-item self-report questionnaire about impulsiveness. The items are scored on a four-point Likert scale. Lastly, the ImpSS is a measure regarding sensation seeking behaviour. The scale consists of 19 true-false statements (Fehrman et al., 2017).

Participants were asked about their use of 18 drugs, namely alcohol, amphetamines, amyl nitrite, benzodiazepines, cannabis, chocolate, cocaine, caffeine, crack, ecstasy, heroin, ketamine, legal highs, LSD, methadone, magic mushrooms, nicotine, VSA and one fictitious drug Semeron. This last drug was added to identify over-claimers. For this thesis however, the focus lies upon only five drugs: amphetamines, cannabis, cocaine, ecstasy and LSD. The participants were asked to indicate if they never used the drug and to specify, if they did use it before, whether that use was in the last day, week, month, year, decade or whether it was over a decade ago. This leads to seven possible answers for every drug.

The data set was downloaded from the UCI website. Although the original data set included the predictor variables 'education', 'country' and 'ethnicity', we did not include these three variables and only used 'age', 'gender' and the seven personality traits as our predictor variables. Consequently, the matrix $\mathbf{X}$ was of size 1885×9. Regarding the distinction between drug users and non-users, people who used a given drug in the last year, last month, last week or last day were considered to be users. As mentioned before, only five of the 18 drugs were taken into account and thus is the matrix $\mathbf{Y^*}$ of size 1885×5. The first column of $\mathbf{Y^*}$ represents amphetamines, the second column represents cannabis, the third cocaine, the fourth ecstasy and the fifth LSD. After transforming the data, it turns out that there are 28 distinct response profiles and therefore matrix $\mathbf{Y}$ is of size 1885×28. Consequently, matrix $\mathbf{D}_y$ is of size 28×28, matrix $\mathbf{Z}$ is of size 28×6 for a structure with only main effects and of size 28×16 for a structure with both main effects and two-way associations. This implies that matrix $\mathbf{D}_{zy}$ is of size 6×6 for a main effects structure and of size 16×16 for a main effects and two-way associations structure.

3.2 Application of Joint Linear Discriminant Analysis

3.2.1 Main effects

When only the main effects are taken into account, the maximum number of dimensions is $M = 6$, since $\max M = min(P, \#columns\ \mathbf{Z})$. The prediction errors for the various values for M are presented in Table 3.1. It can be seen that both the joint and the marginal prediction errors increase as M increases. The lowest joint PE = 0.7735 and the lowest marginal PE = 0.3288 are both obtained for a model where $M = 1$. The marginal PE is significantly lower than the joint PE. The highest joint PE = 0.8897 is obtained for a model where $M = 6$ and the highest marginal PE = 0.3462 is obtained for a model where $M = 5$.

Table 3.1

The cross-validated prediction errors for the JLDA model with only main effects and for different dimensions.

	Prediction error	
Dimension	Joint	Marginal
$M = 1$	0.7735	0.3288
$M = 2$	0.8366	0.3346
$M = 3$	0.8727	0.3374
$M = 4$	0.8743	0.3415
$M = 5$	0.8886	0.3462
$M = 6$	0.8897	0.3461

Note. Joint prediction error is based on the entire response profile. Marginal prediction error is based on the five separate response variables.

An advantage of LDA is the possibility of a graphical representation of the results. A biplot is used to graphically represent the JLDA model in two dimensions (Gabriel, 1971; Gower & Hand, 1996). In such a biplot, the response categories, the predictor variables and the subjects are presented. Figure 3.1 shows the biplot of the JLDA model fitted on the drug consumption data in two dimensions. The subject positions (the black dots) are obtained by a linear combination of their scores on the predictor variables. The variable lines are a representation of participants with varying values on the predictor variable of interest and zero on the other variables. The blue dots indicate typical values of the predictor variables. The numbers in the blue dots represent the standard deviations of the predictor variable, because $\mathbf{X}$ is centered. The green dots indicate the positions for the classes. When a certain subject position falls onto one of the blue lines, this could mean that the subject had average values for the other predictor variables except for the one that its position falls onto. Another possibility is that scores on predictor variables that go in the opposite direction, cancel each other out, for example a score of 3 on the domain Agreeableness (A) and a score of 3 on impulsiveness (I) would sum up to be 0. However, it could also be that it is mere coincidence that a subject falls onto one of the variable lines. It is not possible to say something about the scores

of a subject based on its position in the biplot. From Figure 3.1 it seems that age,

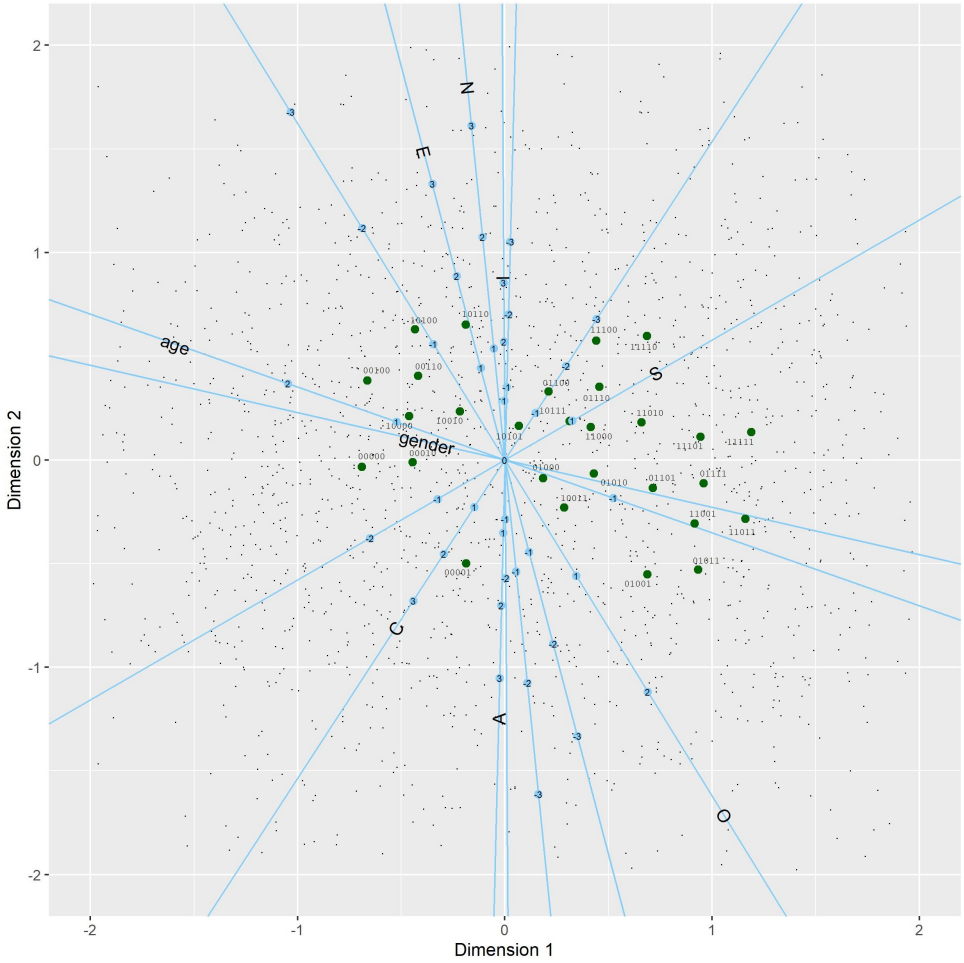
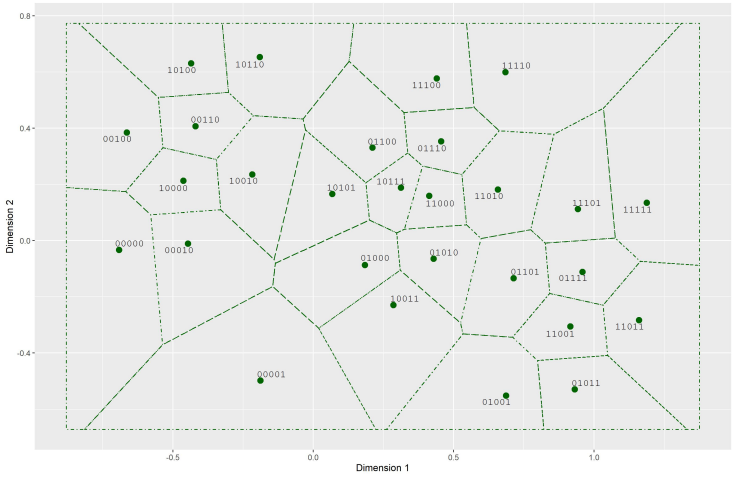


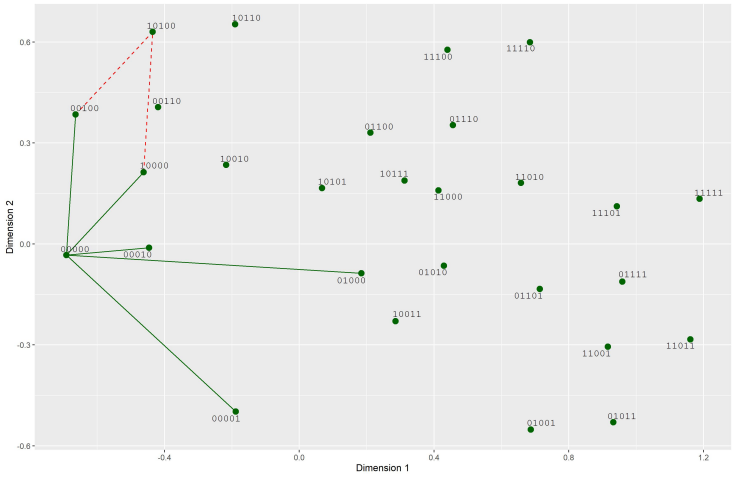
Figure 3.1. Biplot of the JLDA model with only the main effects fitted on the drug data in two dimensions. The black dots are the subject positions. The blue lines represent subjects with varying values on the predictor variable of interest and zero on the other variables. The blue dots indicate typical values of the predictor variables. The green dots represent the positions of the classes. N = Neuroticism. E = Extraversion. O = Openness to Experience. A = Agreeableness. C = Conscientiousness. I = Impulsiveness. S = Sensation Seeking.

gender, and sensation seeking (S) are most important for the first dimension. It can be seen that the positions of older people and females lie on the left side of the biplot while the position of younger people and males lie on the right side. The position of people who seek sensation lie on the right side of the biplot while the position of people who do not seek sensation lie on the left side. The remaining predictor variables are more important for the second dimension. From the figure, it can be seen that the

positions of people who score high on the domains Neuroticism (N), Extraversion (E), and impulsiveness lie at the top of the biplot while the positions of people who score high on Conscientiousness (C), Agreeableness, and Openness to Experience (O) lie at the bottom. Figure 3.2*a* shows that the two-dimensional space is partitioned into 28 regions



(a)



(b)

Figure 3.2. Plots for the JLDA model with only the main effects fitted on the drug data. In (*a*) the two-dimensional space is partitioned into 28 regions of response profiles. In (*b*) the class coordinates are fitted in two dimensions. The green lines show the original five responses. The red dashed lines show how the response profile '10100' is obtained by completing the parallelogram of response profiles '00000', '00100', and '10000'.

of profiles of response variables. Some classes have quite large regions while others have somewhat smaller regions. These regions indicate the class boundaries, so if a participant i has the coordinates $(-0.25, -0.40)$ in the two-dimensional space, then this participant will be assigned to class '00001' since this is the class most nearby. Also, Figure 3.1 and Figure 3.2*a* can be combined. When these figures are compared, it can be seen that high scores on N and on E are associated with the use of cocaine. Furthermore, low scores on S are associated with no drug use or use of a single drug. Figure 3.2*b* shows how the profile of the responses can be obtained by completion of parallelograms of the original responses. The red dashed lines show how the response profile '10100' is obtained by completing the parallelogram of response profiles '00000', '00100', and '10000'. Also, in this plot it can be seen that a positive score on the first dimension seems to indicate the use of multiple drugs while a negative score on this dimension seems to indicate no drug use or use of a single drug. More specifically, the green line going from profile '00000' to '00100' is quite vertical, indicating that the second dimension separates cocaine users from non-cocaine users. The same can be done for the green line going from profile '00000' to '01000', this line is quite horizontal which indicates that the first dimension separates cannabis users from non-cannabis users. The same is true for the green line going from '00000' to '00010' except in this case it concerns separation between ecstasy users and non-ecstasy users. The lines from '00000' to '10000' and '00001' are not vertical nor horizontal but diagonal. This implies that there is no dimension superior to the other with regard to separation between amphetamines users/non-users and LSD users/non-users.

The predictions of the JLDA models can be represented in confusion matrices. Table 3.2 shows the joint confusion matrix for the JLDA model with only the main effects and when $M = 1$. Tables 3.3, 3.4, 3.5, 3.6, and 3.7 show the separate confusion matrices for amphetamines, cannabis, cocaine, ecstasy, and LSD, respectively. The rows represent the actual response profile and the columns represent the predicted response profile. The elements on the diagonal of the matrix indicate the number of times that class is classified correctly. The confusion matrix can be used as an alternative way to obtain the prediction error:

$$\text{PE} = 1 - \frac{\sum diag(\text{confusion matrix})}{n}, \tag{3.6}$$

where n is the sum of the elements in the confusion matrix. Based on Table 3.2, the model has the least trouble predicting non-users since this class is predicted correctly most of the times. However, often (131 times) true non-users are classified as cocaine users. Unfortunately, the model, in most cases, fails to predict the right category even once. For example, cocaine users ('00100') are not once classified as cocaine users. This is probably due to the fact that there were only 9 cocaine users (i.e., participants who solely used cocaine and no other drugs) among the 1885 participants. Looking at the five separate confusion matrices, it can be seen that the model does best in predicting cannabis use and LSD use.

Table 3.2

The 28×28 *confusion matrix of the JLDA model with only main effects for* $M = 1$.

		Predicted																												
		00000	00001	00010	00100	00110	01000	01001	01010	01011	01100	01101	01110	01111	10000	10010	10011	10100	10101	10110	10111	11000	11001	11010	11011	11100	11101	11110	11111
Actual	00000	348	46	27	131	20	21	6	7	1	16	8	3	2	28	33	11	21	25	19	10	2	1	8	4	8	1	6	3
	00001	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1
	00010	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0
	00100	2	1	2	0	0	1	0	1	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
	00110	1	2	0	0	0	1	1	0	0	0	0	1	0	2	0	0	1	0	0	2	0	0	1	0	0	1	1	0
	01000	37	23	10	17	9	13	3	3	5	10	13	8	4	6	11	3	6	12	8	5	4	6	7	13	3	7	4	31
	01001	1	0	1	0	0	2	2	1	1	1	1	1	2	0	1	2	0	1	0	1	0	2	3	8	0	2	0	17
	01010	1	4	0	1	1	3	2	1	1	3	2	0	0	1	3	1	1	3	1	1	0	1	2	5	2	3	2	11
	01011	0	0	0	0	0	1	3	1	1	1	6	2	3	2	2	0	0	1	0	0	1	2	2	6	0	1	4	27
	01100	7	7	1	1	1	1	0	1	2	3	3	0	1	0	0	2	0	1	1	2	1	0	1	2	1	0	1	6
	01101	0	0	0	0	0	0	0	0	0	0	2	1	0	0	0	0	0	0	1	0	0	0	0	1	0	0	1	3
	01110	4	4	0	2	2	2	1	1	1	1	1	3	2	1	3	1	0	3	0	2	0	0	4	4	1	1	3	8
	01111	1	0	0	1	0	0	3	0	1	0	3	1	2	0	0	0	1	2	2	3	1	0	1	1	1	1	1	12
	10000	2	2	0	0	0	0	0	1	0	1	0	0	0	3	0	0	0	1	0	0	0	0	0	0	0	0	0	2
	10010	2	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1
	10011	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
	10100	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1
	10101	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	10110	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	1	0	0	1	0	0	0
	10111	1	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	0
	11000	4	5	1	5	2	2	1	0	2	0	1	1	1	3	0	1	0	3	1	3	1	2	3	4	3	1	3	11
	11001	0	1	0	0	0	1	1	0	0	0	0	1	2	0	0	0	0	2	0	1	1	0	1	2	0	0	0	12
	11010	1	0	1	1	1	2	1	0	1	1	1	2	2	0	0	0	0	0	0	1	0	1	0	2	0	2	0	8
	11011	0	1	0	1	0	0	0	1	2	0	2	2	1	0	1	0	0	1	0	1	0	3	2	9	1	3	4	20
	11100	2	1	0	0	1	1	1	1	0	1	0	1	1	0	1	0	1	2	0	2	0	0	3	4	0	1	2	6
	11101	0	0	0	0	0	0	1	2	1	1	0	0	0	0	0	0	1	0	0	0	1	0	0	0	3	0	0	0	5
	11110	1	3	0	1	0	0	0	3	2	0	5	2	4	0	3	3	1	3	3	2	0	5	2	4	1	3	1	20
	11111	3	0	0	1	1	1	2	1	4	1	4	3	4	0	2	2	0	1	0	3	2	2	4	23	2	1	2	38

Note. The rows represent the actual response profile and the columns represent the predicted response profile.

Table 3.3

Confusion matrix for amphetamines of the JLDA model with only main effects and for $M = 1$.

		Predicted	
		1	0
Actual	1	302	134
	0	525	924

Note. The rows represent the true response profile and the columns represent the predicted response profile. 1 = use. 0 = no use.

Table 3.4

Confusion matrix for cannabis of the JLDA model with only main effects and for $M = 1$.

		Predicted	
		1	0
Actual	1	679	320
	0	127	759

Note. The rows represent the true response profile and the columns represent the predicted response profile. 1 = use. 0 = no use.

Table 3.5

Confusion matrix for cocaine of the JLDA model with only main effects and for $M = 1$.

		Predicted	
		1	0
Actual	1	233	184
	0	633	835

Note. The rows represent the true response profile and the columns represent the predicted response profile. 1 = use. 0 = no use.

Table 3.6

Confusion matrix for ecstasy of the JLDA model with only main effects and for $M = 1$.

		Predicted	
		1	0
Actual	1	348	169
	0	440	928

Note. The rows represent the true response profile and the columns represent the predicted response profile. 1 = use. 0 = no use.

Table 3.7

Confusion matrix for LSD of the JLDA model with only main effects and for $M = 1$.

		Predicted	
		1	0
Actual	1	290	90
	0	477	1028

Note. The rows represent the true response profile and the columns represent the predicted response profile. 1 = use. 0 = no use.

3.2.2 Main effects and two-way interactions

The JLDA model is again fitted to the drug consumption data, but now both the main effects and the two-way interactions are taken into account. The solution of this model indicates how a predictor variable affects the association between two response variables. The prediction errors for the various values for M are presented in Table 3.8. The max-

Table 3.8

The cross-validated prediction errors for the JLDA model with both main effects and two-way interaction effects and for different dimensions.

	Prediction error	
Dimension	Joint	Marginal
$M = 1$	0.7236	0.3024
$M = 2$	0.7857	0.3127
$M = 3$	0.8302	0.3337
$M = 4$	0.8467	0.3407
$M = 5$	0.8525	0.3444
$M = 6$	0.8589	0.3493

Note. Joint prediction error is based on the entire response profile. Marginal prediction error is based on the five separate response variables.

imum number of dimensions is $M = 9$, since $\max M = min(P, \#columns\ \mathbf{Z})$. However, only the first six dimensions are fitted. As was the case for the model that only took main effects into account, both the joint and the marginal prediction errors increase as the values for M increase. The lowest joint PE = 0.7236 and the lowest marginal PE = 0.3024 are both obtained for a model where $M = 1$. This is slightly better than it was for the JLDA model with only the main effects. Again, the marginal PE is significantly lower than the joint PE. The highest joint PE = 0.8589 and the highest marginal PE = 0.3493 are both obtained for a model where $M = 6$.

The model is fitted in two dimensions again. Figure 3.3 shows the biplot of the model. There are no major differences compared to Figure 3.1 for the variable lines, it is just that the class coordinates are different. Nevertheless, these differences are not that big. In Figure 3.4*a* it can be seen that there is some sort of cluster in the bottom right corner with smaller regions. This might be because the class positions are somewhat closer together. For the model with only the main effects (Figure 3.2*a*), this cluster seemed to be more in the middle. It might be the case that the structure of the two-way interactions pulls the class positions towards the bottom right corner with respect to the positions of the single drugs. Furthermore, when we combine Figure 3.3 and Figure 3.4*a*, it can be concluded that the use of multiple drugs is more common among younger participants. Also, sensation-seekers (high score on S) tend to use multiple drugs. High scores on N, E, and I are associated with the use of cocaine. Figure 3.4*b* shows the class coordinates and the lines indicate the distance between no drug use ('00000') and the other five drugs. The smaller lines around the class point '00000' show the interactions. With all these lines, all class coordinates of the combinations of drugs can be represented through the method of completion of parallelograms. Again, the red dashed lines show how the response profile '10010' is obtained by completing the parallelogram of profiles '00000', '10000', and '00010' and adding the line for the interaction 'am:ex' to this parallelogram. Furthermore, the second dimension separates amphetamines users from non-amphetamines users. This is different from the model with

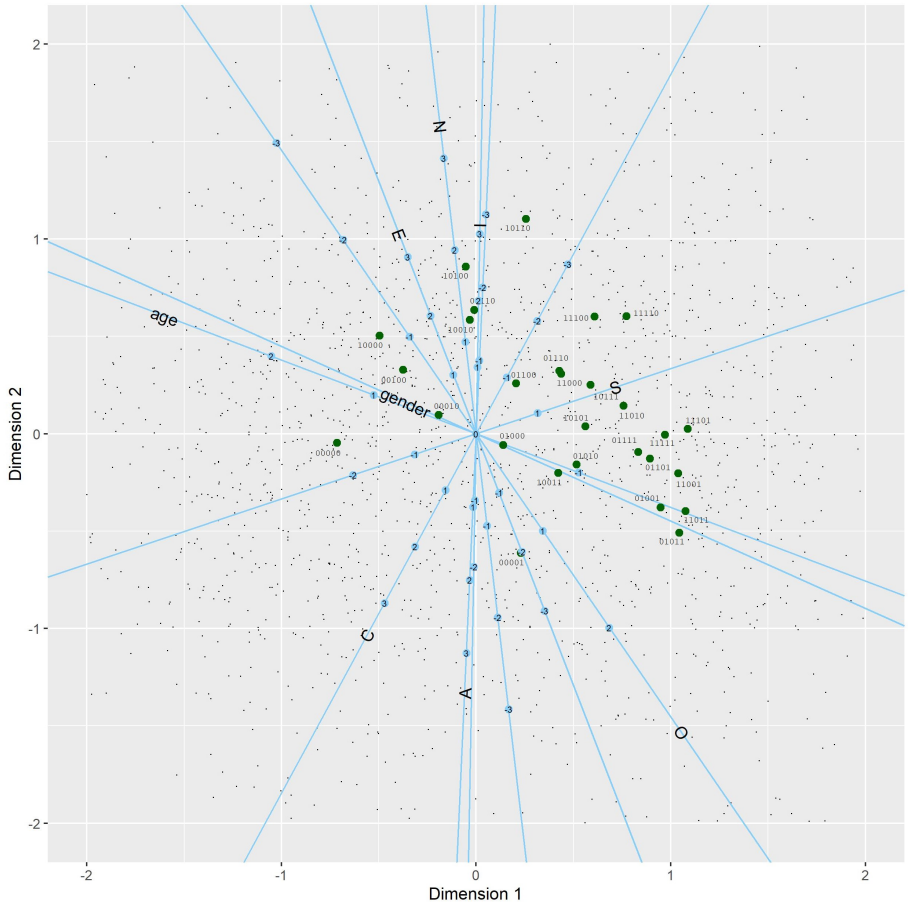
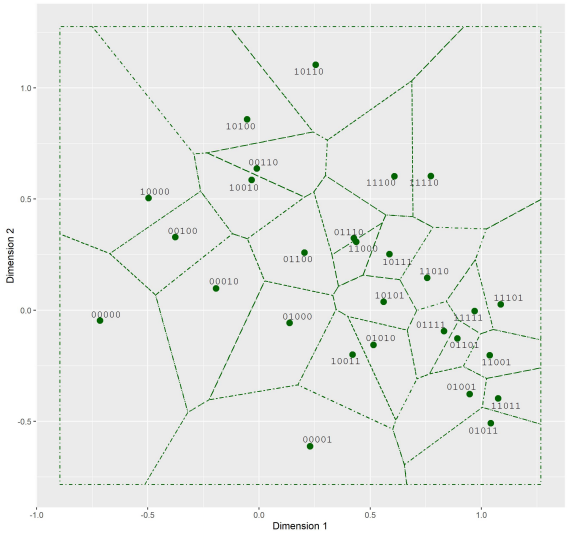


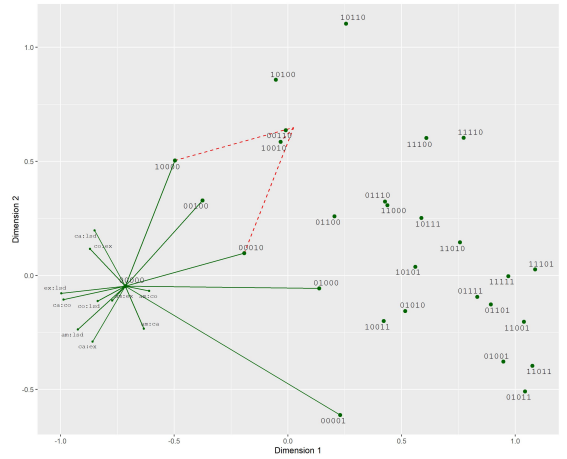
Figure 3.3. Biplot of the JLDA model with main effects and associations fitted on the drug data in two dimensions. The black dots are the subject positions. The blue lines represent subjects with varying values on the predictor variable of interest and zero on the other variables. The blue dots indicate typical values of the predictor variables. The green dots represent the positions of the classes. N = Neuroticism. E = Extraversion. O = Openness to Experience. A = Agreeableness. C = Conscientiousness. I = Impulsiveness. S = Sensation Seeking.

only main effects, where the second dimension separated cocaine users from non-cocaine users. The first dimension separates drug use in quite the same way as the previous model, thus cannabis use and ecstasy use. The remaining lines (from '00000' to '00100' and '00001') are diagonal, implying no superior dimension with regard to separation between cocaine users/non-users and LSD users/non-users.

Table 3.9 shows the joint confusion matrix for the JLDA model when $M = 1$. Tables 3.10, 3.11, 3.12, 3.13, and 3.14 show the separate confusion matrices for amphetamines, cannabis, cocaine, ecstasy, and LSD, respectively. Looking at Table 3.9, this model also has least trouble predicting non-users since this class is predicted correctly most



(a)



(b)

Figure 3.4. Plots for the JLDA model taking both main effects and interactions into account, fitted on the drug data. In (*a*) the two-dimensional space is partitioned into 28 regions of response profiles. In (*b*) the class coordinates are fitted in two dimensions and the two-way interactions are shown. The green lines show the original five responses and the interactions. The red dashed lines show how the response profile '10010' is obtained by completing the parallelogram of response profiles '00000', '00010', '10000', and 'am:ex'.

of the times. However, often true non-users are classified as ecstasy users ('00010'), cocaine users ('00100') or amphetamines users ('10000'). Based on the separate confusion matrices, the model does best in predicting cannabis and LSD use, as did the model with only the main effects.

Table 3.9

The 28×28 *confusion matrix of the JLDA model with* $M = 1$ *and with both the main effects and two-way interactions.*

		Predicted																												
		00000	00001	00010	00100	00110	01000	01001	01010	01011	01100	01101	01110	01111	10000	10010	10011	10100	10101	10110	10111	11000	11001	11010	11011	11100	11101	11110	11111
Actual	00000	458	7	50	60	15	21	0	3	0	9	0	12	3	75	16	9	25	3	10	3	4	1	6	7	9	3	6	1
	00001	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	1	0
	00010	0	0	0	1	1	1	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
	00100	2	0	2	3	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0
	00110	1	0	1	0	0	0	0	0	0	0	0	2	0	3	0	0	1	1	1	0	1	0	0	1	0	0	2	0
	01000	50	6	13	28	7	17	5	3	3	5	7	10	5	14	12	1	10	2	6	3	5	1	5	26	10	20	3	4
	01001	1	4	1	1	0	2	3	0	4	0	1	0	0	0	0	0	0	0	1	2	2	0	0	15	3	9	1	0
	01010	2	1	2	3	3	2	0	2	1	1	2	3	0	2	4	1	2	2	0	0	1	2	1	10	1	5	2	1
	01011	0	0	1	3	0	0	1	1	1	1	0	1	2	0	0	0	0	1	2	4	1	0	2	21	1	14	9	0
	01100	8	1	3	0	3	2	1	2	0	1	3	0	0	2	1	4	2	1	1	0	0	0	1	3	0	5	1	1
	01101	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0	1	3	0	1	1	1
	01110	5	0	2	3	3	2	0	0	1	0	1	4	0	2	1	3	3	1	1	1	1	4	1	5	4	5	2	0
	01111	1	0	1	2	1	0	1	0	0	1	2	2	1	1	2	1	1	1	0	0	0	1	3	8	1	5	1	1
	10000	2	0	1	2	0	0	0	1	0	0	0	0	0	1	1	0	1	0	1	0	0	0	0	1	0	1	0	0
	10010	2	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
	10011	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
	10100	2	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
	10101	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	10110	1	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	1	0	1	0	0	0	0	0	0	0
	10111	1	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	1	0	0	0	1	2	1	0	1	0	0	0
	11000	8	0	6	3	2	1	2	2	0	1	0	2	2	4	1	2	1	1	1	1	1	1	1	9	3	6	2	1
	11001	0	0	0	0	2	0	0	1	0	1	0	0	0	0	1	0	0	1	2	1	0	0	0	9	0	6	0	1
	11010	2	1	0	1	0	2	0	2	0	0	1	0	2	1	0	1	0	0	0	0	0	0	1	7	0	4	1	2
	11011	1	1	0	1	0	0	2	1	1	0	3	1	2	0	1	2	0	0	0	0	1	1	3	17	2	12	3	0
	11100	3	0	2	0	1	0	0	0	1	3	1	0	0	1	0	1	0	1	0	1	1	0	1	7	5	2	0	1
	11101	0	1	0	0	0	0	0	0	0	1	1	1	0	0	0	1	0	0	1	0	0	1	0	6	0	1	1	0
	11110	1	1	0	3	1	1	5	4	1	2	5	1	1	2	4	0	4	1	0	3	0	2	1	16	1	8	2	2
	11111	3	1	1	2	0	2	2	4	6	1	2	3	4	1	0	3	0	2	0	1	1	1	3	33	2	25	2	2

Note. The rows represent the actual response profile and the columns represent the predicted response profile.

Table 3.10

Confusion matrix for amphetamines of the JLDA model with both main effects and interactions and for $M = 1$.

		Predicted	
		1	0
Actual	1	287	149
	0	528	921

Note. The rows represent the true response profile and the columns represent the predicted response profile. 1 = use. 0 = no use.

Table 3.11

Confusion matrix for cannabis of the JLDA model with both main effects and interactions and for $M = 1$.

		Predicted	
		1	0
Actual	1	645	354
	0	113	773

Note. The rows represent the true response profile and the columns represent the predicted response profile. 1 = use. 0 = no use.

Table 3.12

Confusion matrix for cocaine of the JLDA model with both main effects and interactions and for $M = 1$.

		Predicted	
		1	0
Actual	1	188	229
	0	449	1019

Note. The rows represent the true response profile and the columns represent the predicted response profile. 1 = use. 0 = no use.

Table 3.13

Confusion matrix for ecstasy of the JLDA model with both main effects and interactions and for $M = 1$.

		Predicted	
		1	0
Actual	1	279	238
	0	380	988

Note. The rows represent the true response profile and the columns represent the predicted response profile. 1 = use. 0 = no use.

Table 3.14

Confusion matrix for LSD of the JLDA model with both main effects and interactions and for $M = 1$.

		Predicted	
		1	0
Actual	1	265	115
	0	295	1210

Note. The rows represent the true response profile and the columns represent the predicted response profile. 1 = use. 0 = no use.

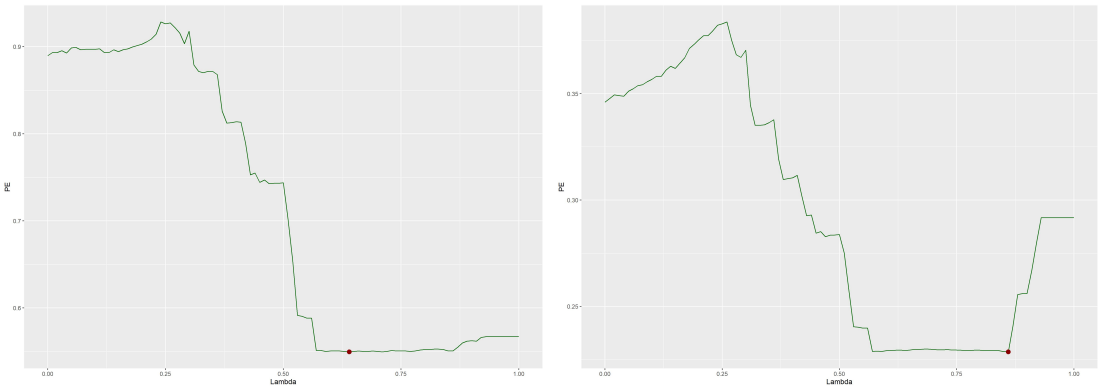
3.3 Application of regularized Joint Linear Discriminant Analysis

3.3.1 Main effects

For regularized JLDA, the tuning parameter λ needs to be selected. In order to select the tuning parameter λ in Equation (2.23), five-fold cross-validated regularized JLDA

is performed on the data set. More specifically, a sequence of models are fitted given a grid of values for λ (a sequence from $\lambda = 0$ to $\lambda = 1$). The value for λ which gives the smallest prediction error would seem to be the best. It turns out that $\lambda = 0.64$ gives the smallest joint prediction error, PE $= 0.5496$. We can also take a look at the marginal PE. Then $\lambda = 0.86$ gives the smallest marginal prediction error, PE $= 0.2287$. Both the joint and the marginal prediction errors are lower in this case than they were for JLDA. It seems as if regularization helps in achieving better prediction errors.

Now for regularized JLDA, it is somewhat difficult to give a graphical representation of the data because both of the selected hyperparameters λ have caused the dimensionality to reduce to $M = 1$. This means that all data points for the subjects fall onto a straight line, as do the coordinates of the different classes. It is, however, possible to inspect how the prediction errors for the grid values of λ behave. In Figure 3.5*a*, it can be seen that the joint prediction error is quite high for low values of λ, it even increases a bit around $\lambda = 0.15$. Then around $\lambda = 0.25$, the joint PE starts to decrease until it hits its minimum value of PE $= 0.5496$ for $\lambda = 0.64$, which is indicated by the red dot. The joint prediction error remains approximately constant up until $\lambda = 0.85$, after which it is starting to slightly increase again. This pattern of going down and then up again is desirable. When the line goes down, it shows that the model is learning and actually keeps doing better until it hits the best achievable prediction error. Then after remaining somewhat constant, the line starts to go up which indicates that the model is overfitting the data. When we look at the marginal prediction error, this pattern is even



(a) *(b)*

Figure 3.5. Plots for the regularized JLDA model taking only main effects into account, fitted on the drug data. In (a) the joint prediction errors are fitted against a sequence of values for λ. The red dot shows for which value of λ the PE is lowest. This is for $\lambda = 0.64$ and this gives PE $= 0.5496$. In (b) the same is done, except that the marginal prediction errors are now fitted against λ. The red dot shows that for $\lambda = 0.86$ the PE is lowest, PE $= 0.2287$.

more clear. Figure 3.5*b* shows this. Again, the red dot represents the minimum value for the marginal prediction error, PE $= 0.2287$ for $\lambda = 0.86$. Despite the big difference

Table 3.15

The 28×28 *confusion matrix of the regularized JLDA model with only main effects when* $M = 1$ *and* $\lambda = 0.64$.

		Predicted																											
		00000	00001	00010	00100	00110	01000	01001	01010	01011	01100	01101	01110	01111	10000	10010	10011	10100	10101	10110	10111	11000	11001	11010	11011	11100	11101	11110	11111
Actual	00000	702	0	0	0	0	114	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	00001	1	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	00010	3	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	00100	7	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	00110	6	0	0	0	0	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	01000	134	0	0	0	0	147	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	01001	3	0	0	0	0	47	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	01010	15	0	0	0	0	41	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	01011	5	0	0	0	0	61	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	01100	19	0	0	0	0	27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	01101	1	0	0	0	0	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	01110	20	0	0	0	0	35	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	01111	6	0	0	0	0	32	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	10000	8	0	0	0	0	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	10010	4	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	10011	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	10100	2	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	10101	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	10110	2	0	0	0	0	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	10111	2	0	0	0	0	7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	11000	23	0	0	0	0	41	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	11001	2	0	0	0	0	23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	11010	4	0	0	0	0	24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	11011	3	0	0	0	0	52	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	11100	7	0	0	0	0	25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	11101	1	0	0	0	0	14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	11110	12	0	0	0	0	60	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	11111	9	0	0	0	0	98	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Note. The rows represent the actual response profile and the columns represent the predicted response profile.

between the joint PE and the marginal PE, the figures in Figure 3.5 are quite similar.

Table 3.15 shows the joint confusion matrix when $M = 1$ and $\lambda = 0.64$. It can be seen that the model has least trouble identifying non-users and users of cannabis ('01000'). However, non-users are also often identified as cannabis users and cannabis users are often identified as non-users. Furthermore, users of all drugs are quite often predicted to be users of only cannabis. Strangely, from Table 3.15 it can be seen that the only predicted response profiles are '00000' and '01000', which seems to mean that the model only distinguishes between no drug use and cannabis use. This is probably due to the dimension reduction to $M = 1$ and because the coefficients of $\mathbf{B}_g$ are shrunken towards zero. When we inspect the matrix $\mathbf{B}_g$, we can see that this is indeed the case:

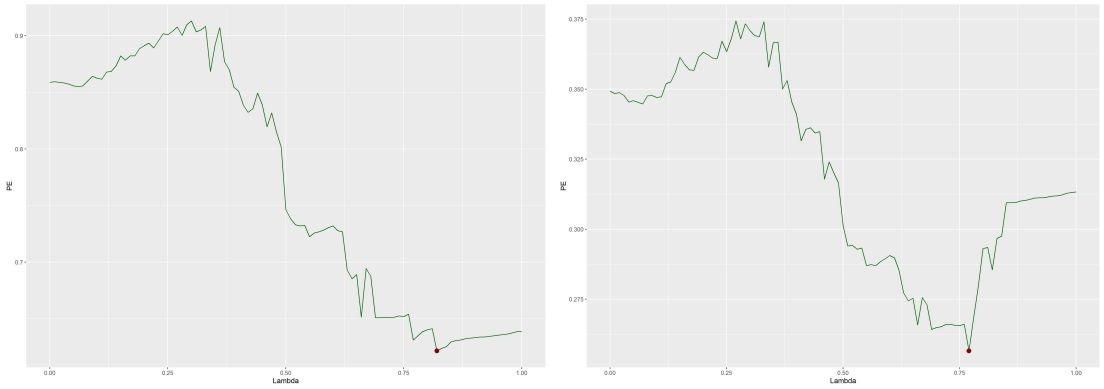
$$\mathbf{B}_g = \begin{bmatrix} -0.053 \\ 0.000 \\ 0.254 \\ 0.000 \\ 0.000 \\ 0.000 \end{bmatrix}.$$

The first row of $\mathbf{B}_g$ represents the intercept, the second row represents amphetamines, the third row represents cannabis, the fourth row represents cocaine, the fifth row represents ecstasy, and the last row represents LSD. The fact that the first row and the third row of $\mathbf{B}_g$ are non-zero, explains why only '00000' or '01000' are predicted in Table 3.15.

3.3.2 Main effects and two-way interactions

The tuning parameter λ needs to be selected again. A value for $\lambda = 0.82$ gives the smallest joint prediction error, PE $= 0.6218$. When the PE is based on the five separate response variables, the smallest marginal prediction error is PE $= 0.2566$ for $\lambda = 0.77$. Both prediction errors are higher for this model, than they were for the model with only the main effects.

For this model, it is also hard to graphically represent the data because the selected dimensionality is $M = 1$. Figure 3.6*a* shows the joint prediction error for different values of λ. The pattern discussed in the previous paragraph is also visible in this figure. However, the line in Figure 3.5*a* seems to decrease faster than the line in Figure 3.6*a*. The red dot represents the point where the joint prediction error was smallest, PE $= 0.6218$ for $\lambda = 0.82$. What also stands out, is that the line has more peaks, while the line in Figure 3.5*a* is more step-wise decreasing. Figure 3.6*b* shows the marginal prediction error for the different values of λ. As before, the pattern is much clearer in this case, but still the line is not very smooth. Again, despite the big difference between the joint PE and the marginal PE, the figures in Figure 3.6 are quite similar.



(a) *(b)*

Figure 3.6. Plots for the regularized JLDA model taking both main effects and interactions into account, fitted on the drug data. In (a) the joint prediction errors are fitted against a sequence of values for λ. The red dot shows for which value of λ the PE is lowest. This is for $\lambda = 0.82$ and this gives PE $= 0.6218$. In (b) the same is done, except that the marginal prediction errors are now fitted against λ. The red dot shows that for $\lambda = 0.77$ the PE is lowest, PE $= 0.2566$.

Table 3.16 shows the joint confusion matrix when $M = 1$ and $\lambda = 0.82$. It can be seen that the model has least trouble identifying non-users. Nevertheless, cannabis users are often classified as non-users or LSD users. Moreover, this model seems only to predict no use ('00000'), LSD use ('00001'), cannabis use ('01000'), and both LSD and cannabis use ('01001'). As mentioned before, this is probably due to dimension reduction to $M = 1$ and the shrinkage of the coefficients of $\mathbf{B}_g$. Again, the matrix $\mathbf{B}_g$ is inspected:

$$\mathbf{B}_g = \begin{bmatrix} 0.000 \\ 0.000 \\ 0.000 \\ 0.000 \\ 0.000 \\ 0.263 \\ 0.000 \\ 0.000 \\ 0.000 \\ 0.000 \\ 0.000 \\ 0.000 \\ 0.000 \\ 0.000 \\ 0.000 \\ 0.000 \end{bmatrix}$$

Table 3.16

The 28×28 *confusion matrix of the regularized JLDA model with both main effects and two-way interactions when* $M = 1$ *and* $\lambda = 0.82$*.*

		Predicted																												
		00000	00001	00010	00100	00110	01000	01001	01010	01011	01100	01101	01110	01111	10000	10010	10011	10100	10101	10110	10111	11000	11001	11010	11011	11100	11101	11110	11111
Actual	00000	688	79	0	0	0	11	38	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	00001	1	1	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	00010	2	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	00100	7	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	00110	9	2	0	0	0	1	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	01000	140	100	0	0	0	5	36	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	01001	4	26	0	0	0	1	19	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	01010	20	21	0	0	0	1	14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	01011	6	29	0	0	0	1	30	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	01100	20	12	0	0	0	3	11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	01101	1	4	0	0	0	0	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	01110	21	21	0	0	0	2	11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	01111	6	23	0	0	0	0	9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	10000	9	1	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	10010	4	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	10011	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	10100	3	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	10101	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	10110	3	1	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	10111	3	2	0	0	0	0	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	11000	24	23	0	0	0	1	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	11001	3	10	0	0	0	0	12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	11010	4	16	0	0	0	0	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	11011	3	36	0	0	0	0	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	11100	8	15	0	0	0	2	7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	11101	1	8	0	0	0	0	6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	11110	17	32	0	0	0	1	22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	11111	9	56	0	0	0	0	42	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Note. The rows represent the actual response profile and the columns represent the predicted response profile.

The first six rows represent the intercept, amphetamines (am), cannabis (ca), cocaine (co), ecstasy (ex), and LSD, respectively. The remaining rows represent the interactions am:ca, am:co, am:ex, am:lsd, ca:co, ca:ex, ca:lsd, co:ex, co:lsd, ex:lsd, respectively. However, this matrix $\mathbf{B}_g$ is somewhat unexpected. Given the fact that only '00000', '00001', '01000', and '01001' are predicted in Table 3.16, one would expect the third and sixth rows of $\mathbf{B}_g$ to be non-zero. However, the predictions are the result of 5-fold cross-validation and thus for each fold, a different model is selected. It could be that $\mathbf{B}_g$ of a certain fold has a non-zero third row, which would explain why '01000' and '01001' are predicted in Table 3.16. The optimal value for λ is determined by means of this cross-validation process and then the optimal λ is again used for all data and that would explain why only the sixth row of $\mathbf{B}_g$ is non-zero.

3.4 Conclusion

Table 3.17

Results (5-fold cross-validated prediction errors) from applying JLDA and regularized JLDA to the drug consumption data.

	Prediction Error		
Model	Joint	Marginal	Dimensionality
JLDA main effects	0.7735	0.3288	$M = 1$
JLDA main effects and interactions	0.7236	0.3024	$M = 1$
rJLDA main effects	0.5496	0.2287	$M = 1$
rJLDA main effects and interactions	0.6218	0.2566	$M = 1$

Note. Joint prediction error is based on the entire response profile. Marginal prediction error is based on the five separate response variables. For JLDA, the dimensionality M is chosen based on for which dimensionality (out of $M = 1, \dots 6$) the prediction error was smallest.

Both JLDA and regularized JLDA were applied to the drug consumption data. A distinction was made between coding for solely main effects or main effects and two-way interactions. The results obtained by the models described above are summarized in Table 3.17. For the JLDA models, the model with the lowest prediction error is chosen. For all models, as expected, the prediction error based on the five separate response variables is lower than the prediction error based on the response profiles. This is because the probability of predicting an entire response profile correctly, for example '10001' is probably lower than predicting the outcome on each of the single response variables. When there are only two equally sized categories, the probability of success, purely based on randomness, is 0.5. When there are 10 equally sized categories, this probability reduces to 0.10, with 20 categories to 0.05 and with 28 categories to 0.036. This is especially the case when there are only a few instances per response profile.

From Table 3.17 it can be seen that regularized JLDA performs better than JLDA. Applying the L_1 penalty to the matrix $\mathbf{B}_g$ results in lower prediction errors. Based on the prediction errors, the model that performs regularized JLDA with only taking the

main effects into account seems to perform best based on both types of prediction errors. Also, this model reduces the dimensionality from $M = 6$ to $M = 1$, which ultimately was the goal of regularized JLDA.

Now that the dimension selection in regularized JLDA is an automatic process, we would like to test whether the model is able to select the correct dimensionality. For this, a simulation study is performed and this will be described in the next section.

4

Method

To gain an understanding of how the regularized JLDA model works and whether it 'does what it is supposed to do', a simulation study is performed. First, the simulation set-up is described in section 4.1 and the various design factors are presented in section 4.1.1. Then, the measure of performance is described in section 4.1.2. Lastly, the statistical analysis and software are described in section 4.2.

4.1 Simulation

In simulation studies, data is created by pseudo-random sampling from known probability distributions (Morris et al., 2019). The reason for using a simulation study could be to check a new method (that is, regularized JLDA) and whether this method does what it is supposed to do. In order to simulate new data, an existing data set is used as a starting point. Using an existing data set ensures ecological validity of the simulated data (Passer, 2014). This data set is the drug consumption data set, which was used and described before (see section 3.1). The idea behind the simulation study is that we create new data, say two-dimensional data, and then regularized JLDA is performed on this new data. Hopefully, the model selects $M = 2$ as the optimal dimensionality. This is repeated many times, for example when 100 data sets consisting of three-dimensional data are generated, we would like the `rjlda` function (see Appendix A) to output $M = 3$ for all 100 data sets. However, this is probably not going to happen 100 out of 100 times. Nevertheless, we are interested in how many times the method selects the correct dimensionality and if not, whether it often selects a dimensionality that is too high or too low. A more detailed description of the simulation procedure is described below (for the actual R-code, see: https://github.com/BrittBroere/master_thesis):

1. Extract the mean of each predictor variable, $\bar{\mathbf{x}}_p$, from the drug data set.

2. Extract the covariance matrix of $\mathbf{X}$, $\mathbf{\Sigma}$, from the drug data set.

3. Perform JLDA on the drug data set with the dimension of interest and extract **B** and **G**. As in section 3, only five, thus $R = 5$, of the 18 drugs are used: amphetamines, cannabis, cocaine, ecstasy, and LSD.

4. Create the matrix $\mathbf{X}_{new}$ by generating n (sample size) new $\mathbf{x}_p$ from the multivariate normal distribution with means from step 1 and the covariance matrix from step 2, $\mathbf{X}_{new} \sim N(\bar{\mathbf{x}}_p, \mathbf{\Sigma})$. It is also possible to create independence among the predictor variables by setting the off-diagonal elements of the covariance matrix to zero.

5. Compute $\mathbf{H} = \mathbf{X}_{new}\mathbf{B}$, which is a $n \times M$ matrix containing the coordinates for the n subjects.

6. Compute the distances, **D**, between **H** and **G**. Thus, **D**(**H, G**) consists of elements d_{ic}, the distance from person i to class c, and is a $n \times C_j$ matrix.

7. Compute probabilities, p_{ic}, for every person for every class. This is done by $p_{ic} = \frac{e^{-d_{ic}^2}}{\sum_c e^{-d_{ic}^2}}$. The matrix **P** consists of the elements p_{ic} and is of size $n \times C_j$. It should be noted that sometimes a person i is an outlier, resulting in large distances between person i and the classes c. This produces an error in the computation of the probabilities. If this is the case, new values for the predictor variables are generated from the multivariate normal distribution (step 4) for this observation.

8. For every person, randomly draw a response profile C_j from the multinomial distribution (this is the profile, not the observation on the five different response variables). Create the response vector $\mathbf{y}^*$ of length n.

9. Transform the profiles in $\mathbf{y}^*$ to responses on each of the five response variables. This results in the matrix $\mathbf{Y}^*_{new}$, which is a $n \times 5$ matrix.

In the end, $\mathbf{X}_{new}$ and $\mathbf{Y}^*_{new}$ are used as input for the `cv.rjlda` function (see Appendix A) in order to select the tuning parameter λ. When λ is selected, $\mathbf{X}_{new}$ and $\mathbf{Y}^*_{new}$ are used as input for the `rjlda` function. Next, the index for λ obtained from running the `cv.rjlda` function is used to extract the estimated **G** from the output of `rjlda`. We can obtain the estimated optimal dimensionality by checking the number of columns of class coordinates matrix **G**.

The process described up till this point reflects a single repetition. However, the process is repeated $N = 1000$ times. So, every condition (see section 4.1.1) is repeated $N = 1000$ times. The end result of the simulation study is a matrix of size $\#design combinations \times 7$. The columns of this matrix indicate the selected dimensionality, $M = 0, \ldots 6$. The maximum number of dimensions is: $\max M = min(P, \#columns\mathbf{Z})$. In the case of the first order, this would be 6 and in the case of the second order this would be 9. However, a maximum of $M = 6$ is used because we generate data with $M = (2, 3, 4)$, so it is unlikely that high dimensions would be selected.

4.1.1 Design factors

The performance of the model is likely to vary over situations (Skrondal, 2000). Therefore, the simulation study is performed under different conditions, which are described in this section. We have selected a total of six factors that are used to alter the condition of the simulation study. These factors can be divided into within-design factors and between-design factors. Between-design factors are used to generate data and within-design factors are all used to analyse the data.

- Within-design factors:
  - Standard-error-rule: This indicates whether the one-standard-error rule is used or not. The one-standard-error rule entails selecting the smallest model that has a prediction error that is within one standard error of the optimal (smallest prediction error) model (James et al., 2013). This is a more parsimonious model. In the current simulation study, usage of the one-standard-error rule is indicated by *1se* and no application of the rule is indicated by *min*.
  - Prediction error: This indicates how the prediction error is calculated. There are two different ways of calculating the prediction error. One possibility is to take the mean of the five prediction errors obtained by the five 2×2 confusion matrices, referred to as the marginal prediction error. Another possibility is basing the prediction error on the $C_j \times C_j$ confusion matrix, referred to as the joint prediction error.
- Between-design factors:
  - True dimensionality: The existing drug data set is used as a base for generating new data. For example, the $\mathbf{B}$ and $\mathbf{G}$ obtained from performing JLDA where $M = 2$, are used to simulate two-dimensional data. Here we use $M = (2, 3, 4)$.
  - Sample size n: In order to study the effect of sample size, different values for n are used, namely $n = (100, 1000, 2000)$.
  - Dependence of the predictors: We also specify whether the predictor variables are independent or dependent. In the case of independence, this is done by setting the off-diagonal elements of the covariance matrix of $\mathbf{X}$ to zero. In the case of dependence, the covariance matrix of the original drug data set is used.
  - Order: This indicates whether the matrix $\mathbf{Z}$ only codes the main effects (first order) or the main effects and the two-way interactions (second order). The order of the generated data corresponds to the order of analysis. For example, if the generated data is of the first order, the model in this condition will also be analysed with the first order.

To emphasize, the within-design factors are used to analyse the data and the between-design factors are used to generate the data. For example, the condition with $M = 2$, $n = 100$, independence of predictor variables, and the first order (so one of each of the possibilities of the four between-design factors) is analysed $2 \times 2 = 4$ times (possible combinations of the two within-design factors).

The one-standard-error rule is often applied when selecting models through cross-validation (Hastie et al., 2009). The reason for including this rule as a design factor is because we were curious whether applying the rule would more often lead to correct dimensionality selection than when the rule was not applied. The reason for specifying whether the prediction error is based on the mean of the five prediction errors obtained by the five 2×2 confusion matrices or on the $C_j \times C_j$ matrix, is because the first might be more sensitive to small changes. This is because the chance of getting an entire response profile, such as '10101', correctly, is probably lower than assessing the predictions on the 5 response variables separately (as we saw in section 3). Secondly, various values for the true dimensionality M are used because we wanted to know whether the model has difficulty selecting the right dimensionality for more complex and thus higher dimensional data. Thirdly, the sample size can have an influence on the results. Data sets, sometimes, do not have a very big sample size. We would like to investigate whether regularized JLDA still works for small sample sizes or if it only works for large sample sizes. That is why different values for n are used. Fourthly, the covariance between the predictor variables was manipulated in such a way that there was either dependence as there is in the original drug data set or there was no covariance between the variables and thus independence. Lastly, the order of the model was also a design factor because we wanted to know whether the model had trouble selecting the correct dimensionality when it took main effects and two-way interactions into account instead of solely main effects.

So, taking all unique combinations of the within-design factors into account, there are in total $2 \times 2 = 4$ different ways to analyse the data. Furthermore, taking all unique combinations of the between-design factors into account, there are in total $3 \times 3 \times 2 \times 2 = 36$ different conditions of the simulation study. Each of these 36 conditions are analysed in the four different ways, so there will be $4 \times 36 = 144$ results. As mentioned, each of these are repeated $N = 1000$ times.

4.1.2 Measure of performance

For every simulation and thus every condition, we recorded how many times each dimension $M = 0, \ldots 6$ is selected by the regularized JLDA model. These numbers are collected in a matrix of size 144×7. However, this large matrix is split up in smaller matrices (for presentation purposes), as will be described in the next section. From this measure, we can derive a measure which indicates the percentage of correctly selected dimensions, the percentage of selected dimensions that are too high and the percentage of selected dimensions that are too low.

4.2 Statistical Analysis and Software

The statistical software R (version 3.6.1) was used to execute the simulation study and to implement the JLDA model and the regularized JLDA model (R Core Team, 2019). Since this extension of LDA is new, there is no top-of-the-shelf package or function in R yet. Therefore, we wrote functions that perform JLDA and regularized JLDA, called `jlda` and `rjlda`, respectively. The R-code for these functions can be found in Appendix A and the theory behind the functions is already discussed in sections 2.2 and 2.4. It is worth noting that the `rjlda` function fits a sequence of models given a grid of values (sequence of $\lambda = 0$ to $\lambda = 1$) for the tuning parameter λ. Also, cross-validation is used to select this hyperparameter λ. For this thesis, a function that performs five-fold cross-validated regularized JLDA, called `cv.rjlda`, was written. This function is used to select the hyperparameter λ. Again, the R-code for this function can also be found in Appendix A. In order for the functions `jlda`, `rjlda`, and `cv.rjlda` to work, some additional functions are needed. These functions can be found in Appendix B.

For the code of the simulation study described in section 4.1, please see the following github account: https://github.com/BrittBroere/master_thesis.

5

Results

This section presents the results of the simulation study. As mentioned in the previous section (4.1.1), there are 144 different conditions for the simulation study. It is hard to present a table with 144 rows in a structured way because of the size of this table. Therefore, this table is split into three smaller tables, each representing the true dimensionality of the generated data, thus $M = 2$, $M = 3$, and $M = 4$. Furthermore, as described in section 4.1.1, the usage of the one-standard-error rule and the way of calculating the prediction error are considered to be within-design factors. These two factors yield four methods of analysing the data. These methods are represented as columns of the three tables. The columns are each divided in three subcolumns representing the percentage of selected dimensionalities that are too low (%L), correct (%C), or too high (%H), respectively. With regard to the columns, *1se* stands for usage of the one-standard-error rule, *min* indicates that this rule is not used and that the optimal value for λ is based on the smallest prediction error. Furthermore, *marginal* refers to the marginal prediction error and *joint* refers to the joint prediction error. Thus, *min-joint* means that the one-standard-error rule is not used and that the joint prediction error is used, which is based on the entire response profile. To clarify, when we mention response profile, the cross classification of the original five response variables is meant. For example, '10101' is the response profile of a subject. The rows of the three tables indicate the condition of the simulation study. The rows are subdivided according to the three different sample sizes and each sample size has four conditions. More specifically, 'dependence' and 'independence' refer to dependence or independence among the predictor variables, respectively. Furthermore, 'or 1' refers to a first order structure (main effects only) and 'or 2' refers to a second order structure (main effects and two-way interaction effects). It is important to note that generated data with a first order structure, was also analysed with a first order structure.

Sections 5.1, 5.2, and 5.3 present the results of the simulation study split by the different dimensionalities of the generated data. Section 5.4 discusses each of the design factors and compares them with each other.

5.1 Data with $M = 2$

Table 5.1

Results of the simulation study for the different conditions when the true dimensionality of the generated data was $M = 2$. Regularized JLDA was applied to the generated data and it was recorded how many times the models selected the true dimensionality, a dimensionality too low, or a dimensionality too high.

		min-joint			min-marginal			1se-joint			1se-marginal		
Condition		%L	%C	%H	%L	%C	%H	%L	%C	%H	%L	%C	%H
$n = 100$													
	dependence, or 1	0.60	8.70	90.70	0.00	10.40	89.60	5.80	**33.10**	61.10	1.80	30.60	67.60
	independence, or 1	1.10	6.70	92.20	0.70	6.00	93.30	9.40	**29.90**	60.70	5.10	25.90	69.00
	dependence, or 2	12.20	19.10	68.70	9.90	14.30	75.80	42.40	**31.30**	26.30	30.90	27.70	41.40
	independence, or 2	9.00	15.60	75.40	12.40	13.80	73.80	39.10	**32.80**	28.10	34.50	27.60	37.90
$n = 1000$													
	dependence, or 1	0.00	17.50	82.50	0.00	64.20	35.80	0.00	75.50	24.50	0.00	**94.70**	5.30
	independence, or 1	0.00	20.60	79.40	0.00	66.10	33.90	0.00	69.90	30.10	0.00	**95.50**	4.50
	dependence, or 2	0.00	56.00	44.00	1.80	**94.70**	3.50	1.40	92.40	6.20	14.80	85.00	0.20
	independence, or 2	0.00	37.50	62.50	0.60	87.30	12.10	0.30	86.90	12.80	8.80	**90.30**	0.90
$n = 2000$													
	dependence, or 1	0.00	28.20	71.80	0.00	82.60	17.40	0.00	84.20	15.80	0.00	**98.30**	1.70
	independence, or 1	0.00	11.60	88.40	0.00	75.60	24.40	0.00	70.90	29.10	0.00	**98.70**	1.30
	dependence, or 2	0.00	57.60	42.40	0.20	**97.90**	1.90	0.00	95.50	4.50	6.50	93.50	0.00
	independence, or 2	0.00	35.70	64.30	0.00	89.70	10.30	0.00	86.80	13.20	1.00	**98.50**	0.50

Note. M = true dimensionality. n = sample size. min-joint = no usage one-standard-error rule and usage of joint prediction error. min-marginal = no usage one-standard-error rule and usage of separate prediction error. 1se-joint = usage one-standard-error rule and usage of joint prediction error. 1se-marginal = usage one-standard-error rule and usage of separate prediction error. %L = percentage of too low selected dimensionality. %C = percentage of correct selected dimensionality. %H = percentage of too high selected dimensionality. or 1 = first order structure (used for both data generation as well as analysis). or 2 = second order structure (used for both data generation as well as analysis). dependence refers to dependence among the predictor variables. independence refers to independence among the predictor variables. The bold faced percentages represent the highest percentage of correct selected dimensionalities for every condition.

Table 5.1 shows the results of the simulation study when the data generated was two-dimensional. The boldfaced percentages show the highest percentage of correct selected dimensionalities for every condition. When the sample size is $n = 100$, it can be seen that the percentages of correct selected dimensionalities are not that high, they range between 29.90% and 33.10%. Next to the true dimensionality being equal to $M = 2$ and having a small sample size, dependence among the predictor variables and the model being of the first order seems to lead to the highest percentage of correct selected dimensionalities (33.10%). Furthermore, in all four conditions with $n = 100$, the highest correct percentages are obtained when the one-standard-error rule is applied and when the prediction error is based on the entire response profile. Also, the models often select dimensionalities that are too high.

When the sample size is $n = 1000$, the percentages of correct selected dimensionalities are quite a lot higher than they were for $n = 100$. The highest correct percentages range between 90.30% and 95.50%. The condition where there is independence among the predictor variables and where the models only take the main effects into account has the highest percentage of correct selected dimensionalities (95.50%). When models with a sample size of $n = 1000$ are not able to select the correct dimensionality, they tend to select a higher dimensionality.

When the sample size is $n = 2000$, the percentages of correct selected dimensionalities are high. Nevertheless, the gap between the highest percentages for $n = 1000$ and the

highest percentages for $n = 2000$ is not as big as it was between $n = 100$ and $n = 1000$. The highest correct percentages range between 97.90% and 98.70%. Just like when $n = 1000$, the condition where there is no covariance between the predictor variables and where the models have a first order structure, has the highest percentage of correct selected dimensionalities (98.70%). When models with a sample size of $n = 2000$ fail to select the correct dimensionality, they often select a higher dimensionality than a lower dimensionality.

When looking at the four separate ways of analysing the data (the use of the one-standard-error rule and how the prediction error is obtained), it can be seen that for all conditions, the percentages correct are quite low for *min-joint*. It however does slightly better when the models are of the second order. Then for $n = 100$, using *min-marginal* for analysing the data does not make it any better. However, for $n = 1000$ and $n = 2000$, using *min-marginal* makes quite a difference. Using *1se-joint* to analyse the data seems the make a difference for $n = 100$ but not so much for $n = 1000$ or $n = 2000$. Lastly, using *1se-marginal* when $n = 100$ seems to result in a slightly lower percentages correct than when *1se-joint* is used. When $n = 1000$ or $n = 2000$, however, the percentages correct improve quite a bit.

5.2 Data with $M = 3$

Table 5.2

Results of the simulation study for the different conditions when the true dimensionality of the generated data was $M = 3$. Regularized JLDA was applied to the generated data and it was recorded how many times the models selected the true dimensionality, a dimensionality too low, or a dimensionality too high.

		min-joint			min-marginal			1se-joint			1se-marginal		
Condition		%L	%C	%H	%L	%C	%H	%L	%C	%H	%L	%C	%H
$n = 100$													
	dependence, or 1	0.10	15.70	84.20	0.10	25.50	74.40	1.10	51.60	47.30	0.60	**59.70**	39.70
	independence, or 1	0.10	13.80	86.10	0.10	19.90	80.00	2.40	49.90	47.70	1.50	**52.30**	46.20
	dependence, or 2	4.70	18.20	77.10	5.00	17.10	77.90	21.40	**47.10**	31.50	19.10	41.40	39.50
	independence, or 2	2.60	13.90	83.50	3.50	14.40	82.10	23.40	**36.90**	39.70	20.20	33.20	46.60
$n = 1000$													
	dependence, or 1	0.00	18.20	81.80	0.00	57.10	42.90	0.00	78.10	21.90	0.00	**94.80**	5.20
	independence, or 1	0.00	30.70	69.30	0.00	67.70	32.30	0.00	86.50	13.50	0.00	**97.50**	2.50
	dependence, or 2	0.00	41.80	58.20	0.00	74.80	25.20	0.00	88.70	11.30	0.10	**97.00**	2.90
	independence, or 2	0.00	51.00	49.00	0.00	90.70	9.30	0.00	93.20	6.80	0.00	**99.20**	0.80
$n = 2000$													
	dependence, or 1	0.00	21.30	78.70	0.00	72.60	27.40	0.00	86.00	14.00	0.00	**98.70**	1.30
	independence, or 1	0.00	37.00	63.00	0.00	83.30	16.70	0.00	92.80	7.20	0.00	**99.80**	0.20
	dependence, or 2	0.00	43.20	56.80	0.00	79.80	20.20	0.00	89.30	10.70	0.00	**98.70**	1.30
	independence, or 2	0.00	59.10	40.90	0.00	95.60	4.40	0.00	95.30	4.70	0.00	**99.80**	0.20

Note. M = true dimensionality. n = sample size. min-joint = no usage one-standard-error rule and usage of joint prediction error. min-marginal = no usage one-standard-error rule and usage of separate prediction error. 1se-joint = usage one-standard-error rule and usage of joint prediction error. 1se-marginal = usage one-standard-error rule and usage of separate prediction error. %L = percentage of too low selected dimensionality. %C = percentage of correct selected dimensionality. %H = percentage of too high selected dimensionality. or 1 = first order structure (used for both data generation as well as analysis). or 2 = second order structure (used for both data generation as well as analysis). dependence refers to dependence among the predictor variables. independence refers to independence among the predictor variables. The bold faced percentages represent the highest percentage of correct selected dimensionalities for every condition.

The results of the simulation study when $M = 3$ are presented in Table 5.2. This table is read in the same way as Table 5.1 was read, so we will only state the important results.

When the sample size is $n = 100$, the percentages of correct selected dimensionalities are somewhat moderate, they range between 36.90% and 59.70%. Next to the true dimensionality being equal to $M = 3$ and having a small sample size, the condition where there is dependence among the predictors and where the models only take the main effects into account seems to lead to the highest percentage of correct selected dimensionalities (59.70%). Furthermore, for two out of the four conditions, using the one-standard-error rule and using the joint prediction error seems to help by achieving the highest percentage of correct selected dimensionalities. More specifically, this applies to the conditions where the models takes both main effects and two-way interactions into account. On the contrary, when models only take the main effects into account, basing the prediction error on the entire response profile is no longer helpful while applying the one-standard-error rule still is.

When $n = 1000$, the percentages of correct selected dimensionalities increase substantially compared to when $n = 100$. The condition where there is independence among the predictors and where the order of the models is two, has the highest percentage of correct selected dimensionalities (99.20%). Also, using the one-standard-error rule and using the marginal prediction error seems to obtain the highest percentages of correct selected dimensionalities for all four conditions.

When the sample size is $n = 2000$, the highest percentages of correct selected dimensionalities are again high. Nevertheless, the difference in highest percentages for $n = 1000$ and $n = 2000$ is not very big. The highest percentage (99.80%) is achieved for the conditions where there is no covariance between the predictor variables, regardless of what the order is. Again for all four conditions, applying the one-standard-error rule and using the marginal prediction error seems to lead to the highest percentages of correct selected dimensionalities.

When we look at the four separate ways of analysing the data in Table 5.2, we see that the same pattern roughly occurs as was the case for $M = 2$.

5.3 Data with $M = 4$

The results of the simulation study, when the data generated was four-dimensional, are presented in Table 5.3. When the sample size is $n = 100$, the percentages of correct selected dimensionalities are moderate to high, they range between 53.10% and 75.30%. The condition where there is dependence among the predictor variables and where the the models have a first order structure, has the highest percentage of correct selected dimensionalities (75.30%). Next to having a sample size of $n = 100$, when models only take the main effects into account, then using the one-standard-error rule and using the marginal prediction error helps by achieving the best percentage of correct selected dimensionalities. However, when the interactions are also taken into account, this is no longer true. In such a situation it seems that it would be better for the prediction error to be based on the entire response profile while still applying the one-standard-error rule.

Table 5.3

Results of the simulation study for the different conditions when the true dimensionality of the generated data was $M = 4$. Regularized JLDA was applied to the generated data and it was recorded how many times the models selected the true dimensionality, a dimensionality too low, or a dimensionality too high.

		min-joint			min-marginal			1se-joint			1se-marginal		
Condition		%L	%C	%H	%L	%C	%H	%L	%C	%H	%L	%C	%H
$n = 100$													
	dependence, or 1	0.40	28.50	71.10	0.40	37.10	62.50	4.20	70.20	25.60	2.80	**75.30**	21.90
	independence, or 1	0.10	23.20	76.70	0.00	36.10	63.90	1.70	70.60	27.70	1.50	**74.10**	24.40
	dependence, or 2	0.10	25.80	74.10	0.70	23.40	75.90	11.40	**59.80**	28.80	8.10	53.00	38.90
	independence, or 2	1.10	18.40	80.50	1.20	17.50	81.30	11.50	**53.10**	35.40	9.70	49.40	40.90
$n = 1000$													
	dependence, or 1	0.00	40.60	59.40	0.00	73.30	26.70	0.00	90.10	9.90	0.00	**98.10**	1.90
	independence, or 1	0.00	18.50	81.50	0.00	51.60	48.40	0.00	77.70	22.30	0.00	**95.10**	4.90
	dependence, or 2	0.00	64.20	35.80	0.00	72.70	27.30	0.00	94.90	5.10	0.00	**96.90**	3.10
	independence, or 2	0.00	72.30	27.70	0.00	88.00	12.00	0.00	96.60	3.40	0.00	**99.10**	0.90
$n = 2000$													
	dependence, or 1	0.00	42.30	57.70	0.00	83.20	16.80	0.00	93.70	6.30	0.00	**99.30**	0.70
	independence, or 1	0.00	17.20	82.80	0.00	55.90	44.10	0.00	79.80	20.20	0.00	**96.70**	3.30
	dependence, or 2	0.00	70.10	29.90	0.00	83.40	16.60	0.00	97.40	2.60	0.00	**98.70**	1.30
	independence, or 2	0.00	82.20	17.80	0.00	95.60	4.40	0.00	99.10	0.90	0.00	**99.70**	0.30

Note. M = true dimensionality. n = sample size. min-joint = no usage one-standard-error rule and usage of joint prediction error. min-marginal = no usage one-standard-error rule and usage of separate prediction error. 1se-joint = usage one-standard-error rule and usage of joint prediction error. 1se-marginal = usage one-standard-error rule and usage of separate prediction error. %L = percentage of too low selected dimensionality. %C = percentage of correct selected dimensionality. %H = percentage of too high selected dimensionality. or 1 = first order structure (used for both data generation as well as analysis). or 2 = second order structure (used for both data generation as well as analysis). dependence refers to dependence among the predictor variables. independence refers to independence among the predictor variables. The bold faced percentages represent the highest percentage of correct selected dimensionalities for every condition.

When a larger sample size is used, so $n = 1000$, the percentages of correct selected dimensionalities increase compared to when the sample size was $n = 100$. Next to the true dimensionality being equal to $M = 4$ and having a large sample size, the condition where there is independence among the predictors and where the models both take main effects and interactions into account leads to the highest percentage (99.10%). Applying the one-standard-error rule and using the marginal prediction error leads to the highest percentages of correct selected dimensionalities for all four conditions.

For the largest sample size used in our simulation study, that is $n = 2000$, the percentages of correct selected dimensionalities are high. The condition where there is independence among the predictor variables and where the models are of the second order, leads to the highest percentage of correct selected dimensionalities (99.70%). Furthermore, as was the case when $n = 1000$, using the one-standard-error rule and using the marginal prediction error helps best by achieving the highest percentages of correct selected dimensionalities for all four conditions.

Again, looking at the four separate ways of analysing the data in Table 5.3, we see that roughly the same pattern occurs as was the case for $M = 2$ and $M = 3$.

5.4 Comparison

In order to draw conclusions from the simulation study and to make inferences about the various design factors, it is useful to compare the different conditions. Table 5.4 presents the results of the simulation study per design factor, when the other conditions

are not considered. The columns are means of percentage lower selected dimensionalities, correct selected dimensionalities, and higher selected dimensionalities respectively. Thus, Table 5.4 can be constructed by looking at a specific design factor in all three tables (Tables 5.1, 5.2, and 5.3). For example, when we look at the true dimensionality, there are 48 conditions per dimension, since there are three different dimensions used in the simulation study ($144/3 = 48$). This can also be checked by the looking at the tables discussed in the previous subsection. For example, Table 5.1 has 48 elements, 12 rows and 4 columns, per type of percentage. When the these elements are averaged, for every type of percentage, the fifth row ($M = 2$) of Table 5.4 is obtained. Each of the factors are discussed in the paragraphs below.

Table 5.4

Results of the simulation study for the different design factors considered alone. Regularized JLDA was applied to the generated data and it was recorded how many times the models selected the true dimensionality, a dimensionality too low, or a dimensionality too high.

		Percentage		
Condition		%L	%C	%H
One-standard-error and PE				
	min-joint	0.89	32.83	66.28
	min-marginal	1.01	58.86	40.13
	1se-joint	4.88	74.10	21.02
	1se-marginal	4.64	79.83	15.53
True dimensionality				
	$M = 2$	5.21	55.69	39.10
	$M = 3$	2.21	61.04	36.75
	$M = 4$	1.14	67.49	31.37
Sample size				
	$n = 100$	7.83	32.49	59.68
	$n = 1000$	0.58	73.56	25.86
	$n = 2000$	0.16	78.16	21.68
Dependence of predictors				
	Yes	2.90	62.24	34.86
	No	2.81	60.57	36.62
Order				
	First	0.58	57.56	41.86
	Second	5.13	65.25	29.61

Note. min-joint = no usage one-standard-error rule and usage of joint prediction error. min-marginal = no usage one-standard-error rule and usage of separate prediction error. 1se-joint = usage one-standard-error rule and usage of joint prediction error. 1se-marginal = usage one-standard-error rule and usage of separate prediction error. %L = percentage of too low selected dimensionality. %C = percentage of correct selected dimensionality. %H = percentage of too high selected dimensionality.

5.4.1 One-standard-error rule and prediction error

Starting with the four ways of analysing the data, these results can be found in the first four rows of Table 5.4. It can be seen that not applying the one-standard-error rule and using the joint prediction error (*min-joint*) leads to the lowest average percentage

of correct selected dimensionalities (32.83%). However, applying the one-standard-error rule and using the marginal prediction error (*1se - marginal*) leads to the highest average percentage of correct selected dimensionalities (79.83%). It seems as if, when the one-standard-error rule is not applied, it is important how the prediction error is calculated. The average percentage of correct selected dimensionalities increases from 32.83% to 58.86% when the marginal prediction error is used instead of the joint prediction error. This is quite an increase. Nevertheless, when the one-standard-error rule is applied, the calculation of the prediction error seems to be less important. The average percentage of correct selected dimensionalities slightly increases from 74.10% to 79.83% when the marginal prediction error is used instead of the joint prediction error.

5.4.2 True dimensionality

Table 5.4 shows that, on average, when the true dimensionality of the generated data was higher, the correct dimensionalities were selected more often. More specifically, a higher average percentage of correct selected dimensionalities is obtained when $M = 3$ compared to when $M = 2$, or when $M = 4$ compared to when $M = 3$. Using two-dimensional generated data leads to the lowest average percentage of correct selected dimensionalities (55.69%). When four-dimensional generated data is used, the average percentage of correct selected dimensionalities is highest (67.49%). However, the differences between the three dimensions are not very big.

5.4.3 Sample size

Looking at Table 5.4, it can be seen that using a small sample size, that is $n = 100$, leads to the lowest average percentage of correct selected dimensionalities (32.49%). Using a larger sample size, $n = 1000$, improves the average percentage of correct selected dimensionalities, since this percentage is now 73.56%. Nevertheless, using an even bigger sample size, $n = 2000$, does not really lead to big changes compared to when the sample size was $n = 1000$. The average percentage of correct selected dimensionalities still increases, but only slightly to 78.16%.

5.4.4 Dependence of the predictor variables

As can be seen in Table 5.4, there is only a minor difference between independence or dependence of the predictor variables in terms of average percentage of correct selected dimensionalities. Dependence among the predictor variables has a slight advantage over independence among the predictors, leading to an average percentage of correct selected dimensionalities of 62.24%.

5.4.5 Order

Lastly, the order can be investigated separately. Table 5.4 shows that the second order is preferred over the first order in terms of average percentage of correct selected di-

mensionalities (65.25%). The difference between these percentages, however, is not very large.

5.4.6 Conclusion

Now that all separate design factors are investigated, one would argue that a combination of the best option per design factor should lead to the highest percentage of correct selected dimensionalities. This means that the models using four-dimensional generated data with sample sizes of $n = 2000$, dependence among the predictor variables, using the second order, and using the method *1se-marginal* to analyse the outcome, should have the highest percentage of correct selected dimensionalities. Looking at Table 5.3, it can be seen that such a combination leads to 98.70% correct selected dimensionalities, which is very high. This, however, is not the highest percentage. In fact, models using three-dimensional generated data with sample sizes of $n = 2000$, independence among the predictor variables, using the first or the second order, and using the method *1se-marginal* to analyse the outcome, have the highest percentage of correct selected dimensionalities, namely 99.80%. However, this is probably due to randomness. First of all, the difference between 98.70% and 99.80% is not substantial. Secondly, the design factors that do not match between the two conditions (true dimensionality and dependence) are factors that do not really differ much in percentage of correct selected dimensionalities, as can be seen in Table 5.4.

In general, it would be best to use the one-standard-error rule and the marginal prediction error. Regularized JLDA has the tendency to select the correct dimensionality or a too high dimensionality, but hardly ever too low. Furthermore, bigger sample sizes are better while true dimensionality, dependence among the predictors, and order structure are less important.

6

Conclusion and discussion

The aims of this thesis were to propose an extension of LDA, propose an algorithm for regularized JLDA, and to evaluate the performance of regularized JLDA. The extension JLDA is able to handle multi-label data and regularized JLDA selects the optimal dimensionality automatically. This section draws conclusions regarding the proposed extensions and the results from the simulation study. In the discussion, limitations and recommendations for future research are discussed.

6.1 Conclusion

6.1.1 JLDA and regularized JLDA

This thesis proposed an extension of LDA, called JLDA. This extension is able to deal with multi-outcome or multi-label data. In addition, an algorithm for regularized JLDA is presented and this algorithm leads to automatic optimal dimensionality selection.

JLDA is a problem transformation method as opposed to an algorithm adaptation method. This extension transforms the multi-labelled classification problem into a single-labelled classification problem through a method called label powerset. This method generates new classes by considering each unique class combination. This implies that the predictors are linked to the profile of response variables. These new classes are stored in an indicator matrix $\mathbf{Y}$ of size $n \times C_j$, where n is the sample size and C_j is the number of classes. Also, a matrix $\mathbf{G}$ is defined, which contains the class coordinates to be estimated. A model structure on $\mathbf{G}$ in terms of main effects and associations can be implied by the design matrix $\mathbf{Z}$. In discriminant analysis, the maximum dimensionality is $\min(P, C_j - 1)$ and adding $\mathbf{Z}$ performs dimensionality reduction, which results in the fact that the the maximum dimensionality of a JLDA model becomes $\min(P, \#columns\ \mathbf{Z})$.

In addition to JLDA, we also proposed a regularized version of JDLA using the L_1 penalty. The penalty is applied to the matrix $\mathbf{B}_g$ (see Equation (2.23)), which contains the coordinates of the separate response variables. The optimal tuning parameter λ is determined using K-fold cross-validation. This tuning parameter minimizes the cross-validation prediction error. The loss function of regularized JDLA is minimized by

an alternating least squares algorithm, which updates the matrices $\mathbf{B}$ and $\mathbf{B}_g$ until convergence, or until a maximum number of iterations is exceeded.

The use of both JLDA and regularized JLDA was demonstrated through the application of these models to the existing drug consumption data set. The outcome was evaluated in terms of five-fold cross-validated prediction error. Two versions of this five-fold cross-validated prediction error were used, namely one based on the entire response profile (referred to as the joint prediction error) and one based on the separate response variables (referred to as the marginal prediction error). In both the application of JLDA and regularized JLDA, the estimated marginal prediction error was lower than the joint prediction error, as expected. The joint prediction errors for JLDA and regularized JLDA are quite high. Nonetheless, it can be concluded that regularization leads to better predictive performance compared to not using regularization.

6.1.2 The simulation study

After proposing JLDA and regularized JLDA, and applying the models to an existing data set, we focused our attention on the algorithm that performs regularized JLDA. The proposed algorithm selects the optimal dimensionality in regularized JLDA automatically. We tested whether this algorithm is able to select the correct dimensionality by means of a simulation study. In order to evaluate whether the correct dimensionality is selected, one would have to know what the actual dimensionality of the data is, which in empirical data analysis is often unknown. Therefore, we generated data of a certain dimensionality so that we knew the actual dimensionality of the generated data.

We used the existing drug consumption data set as a starting point for simulating new data. In the simulation study, we used different factors in order to assess the performance in different situations. We made a distinction between within-design factors and between-design factors in order to create the different conditions. As within-design factors, we used the one-standard-error-rule (yes or no) when selecting the tuning parameter λ and calculation of the prediction error (either the joint prediction error or the marginal prediction error). As between-design factors, we used varying values for the true dimensionality (either two, three, or four) of the generated data and the sample size (either 100, 1000, or 2000). Furthermore, we manipulated the covariance structure of the predictor variables by either using the covariance matrix of the original data set (dependence) or by setting the off-diagonal elements of the covariance matrix to zero (independence). Lastly, the order of the design matrix $\mathbf{Z}$ was manipulated. This matrix had either a main effects structure or a main effects and two-way interactions structure. Combining these within-design factors and between-design factors, resulted in a total of 144 different conditions in which the performance of regularized JLDA was tested, each repeated a 1000 times. How many times each dimension was selected by regularized JLDA, was used as a measure of performance.

Based on the results of the simulation study, it can be concluded that sample size, without consideration of the other factors, is of great importance for regularized JLDA to select the correct dimensionality. The percentage of correct selected dimensionalities increased when larger sample sizes were used ($n = 1000$ or $n = 2000$) compared to using

a small sample size ($n = 100$). Next to the sample size, usage of the one-standard-error rule, without consideration of the other factors, is also of importance. The percentage of correct selected dimensionalities increased substantially when this rule was used compared to when it was not used. The true dimensionality of the data, the covariance structure of the predictor variables, and the order structure of $\mathbf{Z}$, all on their own, did not seem to produce major differences in the percentages of correct selected dimensionalities. When all the factors are considered together, the results of the simulation study showed that regularized JLDA worked best when the true dimensionality of the generated data was three, the sample size was $n = 2000$, there was independence among the predictor variables, either the first or the second order structure was used, the one-standard-error rule was used, and the marginal predictor error was used. In this condition, the correct dimensionality was selected in 99.80% of the time. Furthermore, regularized JLDA has the tendency to select either the correct dimensionality or a too high dimensionality, but hardly ever a too low dimensionality.

It can be concluded that regularized JLDA is capable of selecting the correct dimensionality provided that a large enough sample size is used, the one-standard-error rule is used when selecting the tuning parameter λ, and the prediction error is based on the five separate response variables.

6.2 Discussion

6.2.1 JLDA and regularized JLDA

Using the label powerset method in JLDA to transform the problem into a single-labelled problem, despite being relatively easy and simple, does have a disadvantage. The fact is, as the number of response variables increases, the number of distinct class combinations, thus C_j, also increases with only a few instances per class. Logically, the indicator matrix $\mathbf{Y}$ can be become quite large. A disadvantage of such a large matrix $\mathbf{Y}$ is that the problem of dimensionality grows as the number of response variables increases. For example, the drug data set originally contains the data of 1885 participants about their use of 18 drugs. This implies that when the 18 dichotomous response variables are used, there are $2^{18} = 262144$ distinct class combinations C_j and the indicator matrix $\mathbf{Y}$ will be of size 1885×262144. As the number of classes increases, the size of class coordinates matrix $\mathbf{G}$ also increases. As described before, the design matrix $\mathbf{Z}$ alleviates this problem by performing immediate dimension reduction, which is an advantage of JLDA. However, even after this kind of dimensionality reduction, the maximum number of dimensions might still be quite big. When $\mathbf{Z}$ codes only the main effects, the number of columns, with 18 drugs, is 19. In such a situation, reducing the dimensionality even further is desired. Regularized JLDA overcomes this problem by using the L_1 penalty to automatically obtain the optimal number of dimensions. According to the results of the simulation study, regularized JLDA 'works' and is able to select the optimal dimensionality automatically using the L_1 penalty, most of the times, when the sample size is sufficiently large, the one-standard-error rule is applied, and the marginal pre-

diction error is used. It could be that other penalties, such as the elastic net penalty, work differently or maybe are preferred over the L_1 penalty. The elastic net penalty is combination of the L_1 penalty and the L_2 penalty (Friedman et al., 2010; Hastie et al., 2009). This penalty can deal with correlated variables while the L_1 penalty is indifferent to correlated variables. Future research could investigate the use of other penalization methods. Another important note regarding the dimensionality is that our focus was on dichotomous response variables. JLDA could be extended to non-dichotomous response variables, but the issue of growing dimensionality would become even bigger. Future research could explore if this would impact (regularized) JLDA and what kind of implications this would have on the performance of the models.

As mentioned, JLDA and regularized JLDA were applied to an existing data set, which resulted in high joint prediction errors. Because of the high joint prediction errors (the marginal prediction errors are not that bad), JLDA and regularized JLDA should not yet be used in practice to classify, in this case, drug use in people based on their age, gender, and personality characteristics. However, JLDA and regularized could be used to formulate theories about, in this case, drug use. By graphically representing the model in two dimensions (such as shown in a biplot), researchers can investigate the relationship between predictors and classes. For example, regarding the drug consumption data set, people who score high on the trait sensation seeking, tend to use more than one drug.

Finetuning JLDA and regularized JLDA

It is not yet possible to use JDLA and regularized JLDA in practice for classification. Future research could focus on finetuning the models in order to achieve lower prediction errors. The data matrix $\mathbf{X}$ can be altered in such a way, that it results in higher prediction accuracy. This can be done by means of basis expansions of the predictor variables (Hastie et al., 2009).

Both JLDA and regularized JLDA are linear models. However, it is unlikely that the true function $f(X)$ is linear in X. In order to better capture the true function, we might need more flexible representations for $f(X)$. The idea of basis expansion is to add new variables to the data matrix $\mathbf{X}$, which are transformations of predictor variables. For example, quadratic terms of each predictor variable can be included, X_p^2. Another possibility would be to include splines, or to use kernels. Regardless of what basis expansions are used, the models are linear in these new variables and fitting the models works the same way as before. Including non-linear relationships in the data might reduce bias, and thus improve the accuracy of the models. Once the models are sufficiently optimized, they may be used in practice to classify drug use.

6.2.2 The simulation study

The use of an existing data set in this simulation study has an advantage. It ensures ecological validity of the simulated data. However, at the same time, it also means that the simulated data is dependent on the existing drug consumption data set, which could be a disadvantage. For example, the original data set might contain characteristics that

are specific for this data set and are not transferable to other data sets. The simulated data could have inherited these data set specific characteristics. This is not necessarily a bad thing. It does, however, imply that the conclusion drawn about the performance of regularized JLDA should be interpreted with caution. Future research should investigate the performance of regularized JLDA using other ways of data generation in order to draw more general conclusions.

The regularized JLDA model may perform differently under different circumstances. We have tried to anticipate on this by including diverse within-design factors and between-design factors. It goes without saying that different factors and different values for these factors could have been chosen to create the different conditions. Therefore, we may not know how the model performs under totally different circumstances. This implies that the conclusions drawn about the performance of regularized JLDA are limited to the scope of this simulation study. We did, however, try to make this scope as general as possible. For example, a sample size of 500 is not used in the simulation study design. Nevertheless, the performance of the model with this sample size could be expected to be somewhere in between that of the models with sample sizes 100 and 1000. This does not apply to the other between-design factors. That is, the performance of regularized JLDA is unknown for data with a high true dimensionality, or for data with a more complex covariance structure of the predictor variables. Future research should take this into consideration and could further investigate this.

Concluding, the results of the simulation study look very promising but the performance of the proposed regularized JLDA model should be evaluated by more general simulation studies. Also, different between-design and within-design factors could be used, as could the values of these factors.

Appendix A

R-code main functions

JLDA

```
jlda = function(X, Y, M = 2, ord = 1){
  # Joint Linear Discriminant Analysis for multi-labelled data
  # INPUT
  # X: matrix of size n x P
  # Y: matrix of size n x R
  # M: Number of dimensions
  # order:  1 - main effects
  #         2 - two-way interactions
  #         ...
  #         R - full
  # OUTPUT
  # -------------------------------------------

  library(nnet)
  n = nrow(X)
  P = ncol(X)
  R = ncol(Y)

  ################################################
  ################## center X ####################
  ################################################

  X. = X
  # which variables have more than 2 values (i.e. are continuous)
  cont = lapply(apply(X., 2, unique), length) > 2

  Xc = scale(X.[, cont], center = TRUE, scale = TRUE)
  mx = attr(Xc, "scaled:center")
  sdx = attr(Xc, "scaled:scale")
```

```
X.[, cont] = Xc
Xs = as.matrix(X.)

################################################
################## create Yi ###################
################################################

profile = matrix(NA, n, 1)
for(i in 1:n){
  profile[i, 1] = paste(Y[i,], collapse = "")
}
Yi = class.ind(profile)

################################################
################## create Z ####################
################################################

A = unique(Y)
Aprofile = matrix(NA, nrow(A),1)
for(i in 1:nrow(A)){
  Aprofile[i, 1] = paste(A[i,], collapse = "")
}
Ai = t(class.ind(Aprofile))
aidx = rep(NA, nrow(A))
for(j in 1:nrow(A)){
  aidx[j] = which(Ai[j,] == 1, arr.ind = TRUE)
}
A = A[aidx,]
A = as.data.frame(A)
rownames(A) = colnames(Yi)
colnames(A) = colnames(Y)
if (ord == 1){
  Z = model.matrix(~ ., data = A)
}
else if (ord == 2){
  Z = model.matrix(~ .^2, data = A)
}
else if (ord == 3){
  Z = model.matrix(~ .^3, data = A)
}
else if(ord == R){
  Z = diag(nrow(A))
}

################################################
############ Dzy and Dzy^(-1/2) ################
################################################

Dy = t(Yi) %*% Yi
```

```
Dzy = t(Z) %*% Dy %*% Z
eig.dzy = eigen(Dzy)
Dzy.invsqrt = eig.dzy$vectors %*% diag(sqrt(1/eig.dzy$values))

################################################
################## U and V #####################
################################################

U = t(Xs) %*% Yi %*% Z %*% Dzy.invsqrt
V = (1/n) * t(Xs) %*% Xs

################################################
################## V^(-1/2) ####################
################################################

out.evd = eigen(V)
V2 = out.evd$vectors %*% diag(sqrt(1/out.evd$values))

################################################
################## B and G #####################
################################################

out.svd = mysvd(t(U) %*% V2, M = M)
B = V2 %*% out.svd$v
G = Z %*% solve(Dzy) %*% t(Z) %*% t(Yi) %*% Xs %*% B

################################################
################## loss ########################
################################################

Loss = ssdif(Xs %*% B, Yi %*% G)

################################################
################## output ######################
################################################

results = list(
  Xoriginal = X,
  Yoriginal = Y,
  Yprofile = profile,
  continuous = cont,
  mx = mx,
  sdx = sdx,
  X = Xs,
  Y = Yi,
  V = V2,
  U = U,
  B = B,
  G = G,
```

```
    Z = Z,
    Bg = solve(t(Z) %*% Z) %*% t(Z) %*% G,
    N = Xs %*% B,
    Loss = Loss
  )
}
```

rJLDA

```
rjlda <- function(X, Y, M = 6, ord = 1, crititer = 1e-8, maxiter =
    100, lambda.max = 1, lambda.min = 0){
  # regularized Joint Linear Discriminant Analysis for multi-
     labelled data
  # using alternating least squares algorithm
  # INPUT
  # X: matrix of size n x P
  # Y: matrix of size n x R
  # M: Number of dimensions
  # order:  1 - main effects
  #         2 - two-way interactions
  #         ...
  #         R - full
  # crititer: critical value for convergence
  # maxiter: maximum number of iterations for convergence
  # lambda.max: biggest lambda of a grid of values for lambda
  # lambda.min: smallest lambda of a grid of values for lambda
  # OUTPUT
  # ------------------------------------------

  library(nnet)
  library(MASS)
  n = nrow(X)
  P = ncol(X)
  R = ncol(Y)

  ################################################
  ################## center X ####################
  ################################################

  X. = X
  cont = lapply(apply(X., 2, unique), length) > 2
  Xc = scale(X.[, cont], center = TRUE, scale = TRUE)
  mx = attr(Xc, "scaled:center")
  sdx = attr(Xc, "scaled:scale")
  X.[, cont] = Xc
  Xs = as.matrix(X.)

  ################################################
```

```
################## create Yi ###################
################################################

profile = matrix(NA, n, 1)
for(i in 1:n){
  profile[i, 1] = paste(Y[i,], collapse = "")
}
Yi = class.ind(profile)

################################################
################## create Z ####################
################################################

A = unique(Y)
Aprofile = matrix(NA, nrow(A),1)
for(i in 1:nrow(A)){
  Aprofile[i, 1] = paste(A[i,], collapse = "")
}
Ai = t(class.ind(Aprofile))
aidx = rep(NA, nrow(A))
for(j in 1:nrow(A)){
  aidx[j] = which(Ai[j,] == 1, arr.ind = TRUE)
}
A = A[aidx,]
A = as.data.frame(A)
rownames(A) = colnames(Yi)
colnames(A) = colnames(Y)
if (ord == 1){
  Z = model.matrix(~ ., data = A)
}
else if (ord == 2){
  Z = model.matrix(~ .^2, data = A)
}
else if (ord == 3){
  Z = model.matrix(~ .^3, data = A)
}
else if(ord == R){
  Z = diag(nrow(A))
}

################################################
############ Dzy and Dzy^(-1/2) ################
################################################

Dy = t(Yi) %*% Yi
Dzy = t(Z) %*% Dy %*% Z
Dzy.inv = ginv(Dzy)

# part of G
```

```
iYYYX = Z %*% Dzy.inv %*% t(Z) %*% t(Yi) %*% Xs

# part of B
iXXXY = solve(t(Xs) %*% Xs) %*% t(Xs) %*% Yi

# lists to store all Bg's, B's, G's, and losses
Bg.all = list()
B.all = list()
G.all = list()
loss.all = list()

# S columns of Z
S = ncol(Z)

# make a sequence of lambda's from smallest to biggest
lambdaseq = seq(lambda.min, lambda.max, 0.01)

################################################
## get B and Bg for first (smallest lambda) ##
################################################

# initialize B
B = matrix(0, P, M)

# initialize (random) Bg
Bg = matrix(rnorm(S*M), S, M)

# initialize G
G = Z %*% Bg
# which columns of G are not all 0
dims = which(apply(G, 2, function(x)(all(x == 0)))==FALSE)

# initialize loss
Loss = numeric(maxiter + 1)
Loss[1] = Inf
iter = 2

# convergence is either TRUE or FALSE. It indicates whether the
   stopping criteria have been met
converged = FALSE

# start the algorithm
while(!converged){
  # update B
  B = iXXXY %*% G

  # rescale B
  svd.out = mysvd(Xs %*% B, max(dims))
  B[, dims] = sqrt(n) * B %*% svd.out$v %*% ginv(svd.out$d)
```

```
  # update Bg
  Bg = soft.thres(ginv(t(Z) %*% t(Yi) %*% Yi %*% Z) %*% t(Z) %*%
      t(Yi) %*% Xs %*% B,
                  lambda.min)

  # update G
  G = Z %*% Bg
  # which columns are not all 0
  dims = which(apply(G, 2, function(x)(all(x == 0)))==FALSE)

  # check convergence
  Loss[iter] <- ssdif(Xs %*% B[, dims], Yi %*% G[, dims])
  converged <- ((Loss[iter-1]-Loss[iter]) < crititer | iter-1 >=
      maxiter)

  # some in between feedback
  #cat(iter, Loss[(iter - 1)], Loss[iter], "\n")

  # increase the iteration number
  iter <- iter + 1
}

# store B, G, loss and Bg for smallest lambda in the lists
Bg.all[[1]] = Bg
B.all[[1]] = B
G.all[[1]] = G
loss.all[[1]] = Loss[2:(iter-1)]

#####################################################
## use these results (Bg, B and G) from smallest ##
## lambda as start values for other lambdas      ##
#####################################################

for(lamb in 2:length(lambdaseq)){
  # previous matrix for B
  B = B.all[[(lamb-1)]]

  # previous matrix for Bg
  Bg = Bg.all[[(lamb-1)]]

  # initialize loss
  Loss = numeric(maxiter + 1)
  Loss[1] = Inf
  iter = 2

  # convergence is either TRUE or FALSE. It indicates whether
     the stopping criteria have been met
  converged = FALSE
```

```
  # start the algorithm
  while(!converged){
    # update B
    B = iXXXY %*% G

    # if the max dims is 0, it means everything is zero so
       continuing the function would be useless
    if(length(dims) == 0){
      cat("Stop, the dimension for lambda ", lambdaseq[lamb], "
         is 0. \n")
      break
    }

    # rescale B
    svd.out = mysvd(Xs %*% B, length(dims))
    B[, dims] = sqrt(n) * B %*% svd.out$v %*% ginv(svd.out$d)

    # update Bg
    Bg = soft.thres(ginv(t(Z) %*% t(Yi) %*% Yi %*% Z) %*% t(Z)
       %*% t(Yi) %*% Xs %*% B,
                    lambdaseq[lamb])

    # update G
    G = Z %*% Bg
    # which columns are not all 0
    dims = which(apply(G, 2, function(x)(all(x == 0)))==FALSE)

    # check convergence
    Loss[iter] <- ssdif(Xs %*% B[, dims], Yi %*% G[, dims])
    converged <- ((Loss[iter-1]-Loss[iter]) < crititer | iter-1
       >= maxiter)

    # some in between feedback
    #cat(iter, lamb, Loss[(iter - 1)], Loss[iter], "\n")

    # increase the iteration number
    iter <- iter + 1
  }

  # store B, G, loss and Bg for different lambdas in the lists
  Bg.all[[lamb]] = Bg
  B.all[[lamb]] = B
  G.all[[lamb]] = G
  loss.all[[lamb]] = Loss[2:(iter-1)]

}
################################################
################## output ######################
```

```
  ##############################################

  results = list(
    Xoriginal = X,
    Yoriginal = Y,
    Yprofile = profile,
    continuous = cont,
    mx = mx,
    sdx = sdx,
    X = Xs,
    Y = Yi,
    dims = dims,
    B.all = B.all,
    Bg.all= Bg.all,
    G.all = G.all,
    Z = Z,
    lambda.sequence = lambdaseq,
    Loss.all = loss.all
    )
}
```

The cross-validation function

```
cv.rjlda = function(X, Y, M = 6, K = 5, myseed, ord = 1,
                    lambda.min, lambda.max,
                    stderror = T, PE.2by2 = T){
  # cross-validated regularized Joint Linear Discriminant Analysis
  # for multi-labelled data
  # INPUT
  # X: matrix of size n x P
  # Y: matrix of size n x R
  # M: Number of dimensions
  # K: Number of folds for cross-validation
  # myseed: a seed for reproducibility
  # order:  1 - main effects
  #         2 - two-way interactions
  # lambda.max: biggest lambda of a grid of values for lambda
  # lambda.min: smallest lambda of a grid of values for lambda
  # stderror: boolean indicating usage of one-standard-error rule
  # PE.2by2: boolean indicating usage of marginal prediction error
  # OUTPUT
  # ------------------------------------------------------------

  ##############################################
  ##### results on the complete data set #####
  ##############################################

  set.seed(myseed)
```

```
results.complete = rjlda(X, Y, M = M, ord = ord,
                         lambda.min = lambda.min,
                         lambda.max = lambda.max)
lambdas = length(results.complete$lambda.sequence)

##############################################
############# cross-validation #############
##############################################

n = nrow(X)
R = ncol(results.complete$Yoriginal)

# random folds
folds <- cut(seq(1, n), breaks = K, labels=FALSE)
folds = sample(folds)

# extract the response profiles and the original Y
Yprofile = results.complete$Yprofile
Yoriginal = results.complete$Yoriginal

# initialize predictions matrix
big.preds = matrix(NA, nrow(Y), ncol = lambdas)

# initialize cross-validated prediction error matrix
PE = matrix(nrow = lambdas, ncol = 1)

# initialize prediction error per fold matrix
PE.per.K = matrix(nrow = lambdas, ncol = K)

# start cross-validation
for(k in 1:K){
  # extract indices of current fold
  idx = which(folds == k, arr.ind = TRUE)

  # initialize predictions for current fold matrix
  predic = matrix(NA, nrow = length(idx), ncol = lambdas)

  # perform regularized JLDA on remaining folds
  out = rjlda(X[-idx, ], Y[-idx, ], M = M, ord = ord,
              lambda.max = lambda.max, lambda.min = lambda.min)

  # predict outcome for current fold
  predic = predict.rjlda(newx = X[idx, ], out)
  rownames(predic) = as.character(idx)

  # store predictions for current fold in predictions matrix
  big.preds[as.numeric(rownames(predic)), ] = predic

  # JOINT PREDICTION ERROR
```

```
    if(PE.2by2 == F){
      # initialize matrix with ones if predicted correctly, 0
         otherwise
      # (for every lambda)
      preds.01 = matrix(nrow = length(idx), ncol = lambdas)

      # for every subject in the current fold
      for(i in 1:length(idx)){
        # for every value for lambda
        for(j in 1:lambdas){
          # if predicted response profile matches true response
             profile, add 1, otherwise 0
          if( predic[i,j] == Yprofile[
            as.numeric(rownames(predic)[i]),] ){
            preds.01[i,j] = 1
          }
          else{
            preds.01[i,j] = 0
          }
        }
      }

      # for every lambda, count how many times response profile is
          predicted correctly
      sums.28by28 = colSums(preds.01)/length(idx)

      # joint prediction error for the current fold
      PE.per.K[, k] = 1-sums.28by28
      } else{

    # MARGINAL PREDICTION ERROR

        # for every lambda, transform the predictions for current
           fold matrix (predic)
        # into matrix with 5 columns, one for each R
        for(l in 1:lambdas){
          predic.mat = matrix(nrow = length(idx), ncol = R)
          colnames(predic.mat) = colnames(Yoriginal)
          rownames(predic.mat) = as.numeric(names(
            strsplit(predic[,l], "")))

          for(i in 1:length(idx)){
            predic.mat[i, ] = as.numeric(strsplit(
              predic[i,l], "")[[1]])
          }

          # initialize matrix with ones if predicted correctly, 0
             otherwise
          # (for every lambda)
```

```
        preds.01 = matrix(nrow = length(idx), ncol = R)

        # for every subject in the current fold
        for(i in 1:length(idx)){
          # for every response variable R
          for(r in 1:R){
            # if predicted outcome for R matches true outcome
               for R, add 1, otherwise 0
            if( predic.mat[i,r] == Yoriginal[
              as.numeric(rownames(predic.mat)[i]),r] ){
              preds.01[i,r] = 1
            }
            else{
              preds.01[i,r] = 0
            }
          }
        }

        # for every lambda, count how many times outcome for R
           is predicted correctly
        sums.2by2 = colSums(preds.01)/length(idx)

        # marginal prediction error for the current fold
        PE.per.K[l,k] = mean(1-sums.2by2)
      }
    }
} # end of cross-validation

# cross-validated prediction error for every lambda
PE[,1] = round(rowMeans(PE.per.K), 7)

# index for lambda with best results
idx.lambda = which.min(PE)

# if the one-standard-error rule is applied
if(stderror == T){
  # initialize matrix to store the standard errors
  SE = matrix(nrow = lambdas, ncol = 1)

  # for every lambda
  for(i in 1:lambdas){
    # standard deviation per lambda
    result = sd(PE.per.K[i,])

    # standard error per lambda
    SE[i,] = result/sqrt(K)
  }

  # range of one standard error below the best PE and
```

```
  # one standard error above the best PE
  range = c(PE[idx.lambda, ] - SE[idx.lambda],
            PE[idx.lambda, ] + SE[idx.lambda])

  PE. = PE

  # if PE is one SE below the best PE and one SE above the best
     PE
  # set to 1 (:lambdas because most sparse model is desired)
  PE.[idx.lambda:lambdas,][
    PE.[idx.lambda:lambdas,] >= range[1] &
      PE.[idx.lambda:lambdas,] <= range[2] ] = 1

  # which of indices that are one SE below the best PE and one
     SE above the best PE
  # is biggest
  idx.stderror = max(which(PE.[,1] == 1.0))
}

##############################################
############ confusion matrices ############
############     per lambda     ############
##############################################

# initialize list to store confusion matrices
list.Cj.matrix = list()

# get names of the response profiles
Yprofile.uniq = colnames(results.complete$Y)

# number of response profiles
n.Cj = ncol(results.complete$Y)

# for every lambda
for(i in 1:lambdas){
  # initialize empty matrix of size (in this case) 28x28
  list.Cj.matrix[[i]] = matrix(0, nrow = n.Cj, ncol = n.Cj)

  # get names of the response profiles
  colnames(list.Cj.matrix[[i]]) = Yprofile.uniq
  rownames(list.Cj.matrix[[i]]) = Yprofile.uniq

  # for every response profile
  for(r in 1:n.Cj){
    for(j in 1:n.Cj){
      # check if predicted response profile matches the actual
         response profile
      # count how many times this is the case and put this sum
         in confusion matrix
```

```
      list.Cj.matrix[[i]][r, j] = sum(
        Yprofile == Yprofile.uniq[r] &
          big.preds[,i] == Yprofile.uniq[j])
    }
  }
}

##############################################
############ confusion matrices ##############
########## per response variable ############
############    per lambda     ##############
##############################################

# initialize list to store confusion matrices
list.rCj.matrix = list()

# for every lambda
for(i in 1:lambdas){
  # initialize list of size R
  rCj.matrix = vector('list', ncol(results.complete$Yoriginal))

  # for every R
  for(z in 1:ncol(results.complete$Yoriginal)){
    # matrix Z for a single R
    Z.matrix = cbind(results.complete$Z[, 1+z],
                     1 - results.complete$Z[,1+z])

    # check if predicted response profile that contains the
       single R matches the actual
    # response profile that contains the single R and count how
       many times this is the case
    # put this sum in 2x2 confusion matrix
    r.Cj = t(Z.matrix) %*% as.matrix(list.Cj.matrix[[i]]) %*%
      Z.matrix
    dimnames(r.Cj) = list(actual = c('1', '0'),
                          predicted = c('1', '0'))
    names(rCj.matrix)[z] = colnames(results.complete$Z)[z+1]

    # store 2x2 confusion matrix in list of size R
    rCj.matrix[[z]] = r.Cj
  }
  # for every lambda store the list of size R
  list.rCj.matrix[[i]] = rCj.matrix
}

##############################################
################## output ####################
##############################################
```

```
  # if the one-standard-error rule is applied
  if(stderror == T){
    results = list(
      indx.lambda = idx.stderror,
      lambda = results.complete$lambda.sequence[idx.stderror],
      prediction.errors = PE,
      confusion.matrices = list.Cj.matrix,
      confusion.matrices.per.R = list.rCj.matrix,
      best.PE = PE[idx.stderror, ],
      best.confusion.matrix = list.Cj.matrix[[idx.stderror]],
      best.confusion.matrix.per.R = list.rCj.matrix[[idx.stderror
         ]]
      )
    }
  else{
  # if the one-standard-error rule is NOT applied
    results = list(
      indx.lambda = idx.lambda,
      lambda = results.complete$lambda.sequence[idx.lambda],
      prediction.errors = PE,
      confusion.matrices = list.Cj.matrix,
      confusion.matrices.per.R = list.rCj.matrix,
      best.PE = PE[idx.lambda, ],
      best.confusion.matrix = list.Cj.matrix[[idx.lambda]],
      best.confusion.matrix.per.R = list.rCj.matrix[[idx.lambda]]
      )
  }
}
```

Appendix B

R-code additional functions

The loss function

```
ssdif = function(A,B){
  # sum of squared differences
  ss = sum((A - B)^2)

  return(ss)
}
```

The svd function

```
mysvd = function(A, M){
  # an SVD procedure that keeps matrices

  I = nrow(A)
  J = ncol(A)
  svd.out = svd(A, nu = M, nv = M)
  U = matrix(svd.out$u, I, M)
  V = matrix(svd.out$v, J, M)
  D = diag(svd.out$d, M, M)

  result = list(
    u = U,
    v = V,
    d = D
  )
  return(result)
}
```

The soft-thresholding function

```
soft.thres = function(z,gamma){
  z = sign(z) * pmax(abs(z) - gamma, 0)

  return(z)
}
```

The predict function

```
predict.rjlda = function(newx, output){
  # sequence of lambdas
  lambdas = length(output$G.all)

  # center new X
  cont = output$continuous
  Xs = scale(newx[, cont], center = output$mx, scale = output$sdx)
  X = newx
  X[, cont] = Xs

  # initialize yhat for every lambda
  yhat.all = matrix(nrow = nrow(newx), ncol = lambdas)

  # for every lambda
  for(i in 1:lambdas){
    # discriminant scores
    N = X %*% output$B.all[[i]]

    # get G
    G = output$G.all[[i]]
    rownames(G) = colnames(output$Y)

    # calculate distance between subjects (discriminant scores)
       and class coordinates
    ones.C = matrix(1, nrow(G), 1)
    ones.I = matrix(1, nrow(newx),1)
    D2 = diag(N %*% t(N)) %*% t(ones.C) + ones.I %*%
      t(diag(G %*% t(G))) - 2* N %*% t(G)
    colnames(D2) = colnames(output$Y)

    # assign person to class whose distance is smallest
    yhat.all[,i] = matrix(colnames(D2)[apply(D2, 1, which.min)],
                          nrow(newx), 1)
  }
  return(yhat.all)
}
```

References

Adachi, K. (2004). Correct classification rates in multiple correspondence analysis. *Journal of the Japanese Society of Computational Statistics*, *17*, 1–20.

Adachi, K. (2016). *Matrix-based introduction to multivariate data analysis*. Springer.

Breiman, L., & Spector, P. (1992). Submodel selection and evaluation in regression. the x-random case. *International Statistical Review*, *60*(3), 291–319.

Clemmensen, L., Hastie, T., Witten, D., & Ersboll, B. (2011). Sparse discriminant analysis. *Technometrics*, *53*(4), 406–413.

Duda, R. O., Hart, P. E., & Stork, D. G. (2001). *Pattern classification*.

Fehrman, E., Mirkes, E. M., Muhammad, A. K., Egan, V., & Gorban, A. N. (2017). The five factor model of personality and evaluation of drug consumption risk. In F. Palumbo (Ed.). A. Montanari (Ed.). M. Vichi (Ed.), *Data science: Innovative developments in data analysis and clustering* (pp. 231–242). Springer. https://doi.org/10.1007/978-3-319-55723-6_18.

Fisher, R. A. (1936). The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, *7*.

Friedman, J., Hastie, T., & Tibshirani, R. (2010). Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, *33*(1).

Fukunaga, K. (2013). *Introduction to statistical pattern recognition*. Academic Press.

Gabriel, K. R. (1971). The biplot graphic display of matrices with application to principal component analysis. *Biometrika*, *58*(3), 453–467.

Gower, J. C., & Hand, D. J. (1996). *Biplots*. Chapman & Hall.

Hastie, T., Buja, A., & Tibshirani, R. (1995). Penalized discriminant analysis. *The Annals of Statistics*, *23*(1), 73–102.

Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The elements of statistical learning: Data mining, inference, and prediction*. Springer.

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). *An introduction to statistical learning: With applications in r*. Springer.

Kohavi, R. (1995). A study of cross-validation and bootstrap for accuracy estimation and model selection. *International Joint Conference on Artificial Intelligence*, *14*, 1137–1143.

Morris, T. P., White, I. R., & Crowther, M. J. (2019). Using simulation studies to evaluate statistical methods. *Statistics in Medicine*, *38*(11), 2074–2102.

Park, C. H., & Lee, M. (2008). On applying linear discriminant analysis for multi-labeled problems. *Pattern Recognition Letters*, *29*(7), 878–887.

Passer, M. W. (2014). *Research methods: Concepts and connections*. Worth Publishers.

Pushpa, M., & Karpagavalli, S. (2017). Multi-label classification: Problem transformation methods in tamil phoneme classification. *Procedia Computer Science*, *115*, 572–579.

R Core Team. (2019). *R: A language and environment for statistical computing*. R Foundation for Statistical Computing. Vienna, Austria. https://www.R-project.org

Shu, X., Xu, H., & Liang, T. (2015). A least squares formulation of multi-label linear discriminant analysis. *Neurocomputing*, *156*, 221–230.

Skrondal, A. (2000). Design and analysis of monte carlo experiments: Attacking the conventional wisdom. *Multivariate Behavioral Research*, *35*(2), 137–167.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society*, *58*(1), 267–288.

Tsoumakas, G., & Katakis, I. (2009). Multi-label classification: An overview. *International Journal of Data Warehousing and Mining*, *3*, 1–13.

Wang, H., Ding, C., & Huang, H. (2010). Multi-label linear discriminant analysis. In K. Daniilidis, P. Maragos, & N. Paragios (Eds.), *Computer vision - eccv 2010* (pp. 126–139). Springer.

Zhang, P. (1993). Model selection via multifold cross validation. *The Annals of Statistics*, *21*(1), 299–313.