Classification tree algorithms for grouped variables

Audrey Poterie, Jean-François Dupuy, Valérie Monbet, Laurent Rouviere

Classification tree algorithms for grouped variables

Audrey Poterie[a,*], Jean-François Dupuy[a], Valérie Monbet[b], Laurent Rouvière[c]

[a]*IRMAR, INSA de Rennes, 20 Avenue des Buttes de Coesmes, 35708 Rennes, France*
[b]*IRMAR, Université de Rennes 1, 263 Avenue du Général Leclerc, 35042 Rennes, France*
[c]*IRMAR, Université de Rennes 2, Place du Recteur Henri Le Moal, 35043 Rennes, France*

Abstract

The problem of predicting a binary variable is considered when a p-dimensional vector of inputs is available. These inputs are structured in a number of known groups. The objective is to take this structure into account to build a classification tree. Two tree-based approaches are proposed: the Tree Linear Discriminant Analysis algorithm (TLDA) and the Tree Penalized Linear Discriminant Analysis algorithm (TPLDA). They consist in splitting a node by repeatedly selecting a group of inputs and then applying a linear discriminant method based on this group. This process is iterated until some stopping criterion is satisfied. A pruning strategy is proposed to select an optimal tree. The two algorithms differ in the discriminant method used in the splitting process: TLDA applies a linear discriminant analysis (LDA) while TPLDA performs a regularized linear discriminant analysis. These two proposed methods are computationally less demanding than classical existing multivariate classification tree methods. Moreover, the resulting trees are more easily interpretable. The good performances of the proposed algorithms and the interest of using them in terms of classification accuracy and interpretation are demonstrated in comparison with alternative reference methods (CART, group-lasso logistic regression) though applications on simulated and real gene expression data.

*Corresponding author

Email addresses: audrey.poterie@insa-rennes.fr (Audrey Poterie), jean-francois.dupuy@insa-rennes.fr (Jean-François Dupuy), valerie.monbet@univ-rennes1.fr (Valérie Monbet), laurent.rouviere@univ-rennes2.fr (Laurent Rouvière)

Keywords: Supervised classification, variable selection, multivariate classification tree algorithms, groups of inputs, regularized linear discriminant analysis, gene expression data.

1. Introduction

Consider the supervised classification setting where the problem consists in predicting a binary variable Y, based on a vector $\mathbf{X}$ which takes values in $\mathbb{R}^p$. Suppose further that the inputs are divided into J different groups. In many situations, inputs can have an obvious group structure. A group structure can also be defined by the user to capture the underlying input associations. Moreover, in some cases, the study of groups of inputs can make more sense than the study of the inputs taken individually. For example, in the analysis of gene expression data, data sets contain the expression levels of thousands genes in a much smaller number of observations. Then, it has become frequent to describe the data using a small number of metagenes based on linear combinations of genes or clusters of genes (Tamayo et al., 2007; Lee & Batzoglou, 2003). Another example is functional data, like spectrometry data, where researchers are often more interested by identifying discriminatory parts of the curve rather that individual wave lengths (Tardivel et al., 2017). Finally, categorical inputs can be converted into a group of dummy variables that must be treated as a group. In all these situations, elaborating a classification rule based on groups of inputs rather than on the individual inputs can improve both interpretation and prediction accuracy (Gregorutti et al., 2015). Several methods have already been proposed to deal with this problem. For instance, the logistic regression regularized by the Group Lasso penalty (GL) enables to elaborate classification rules based on groups of inputs (Meier et al., 2008). As far as we know, this problem has not been studied for classification trees.

Tree-based methods are popular in statistical data classification. Classification tree algorithms elaborate classification rules by means of recursive splits. Starting with all the data, these algorithms partition the data space into two or more regions, also called nodes, and repeat the splitting procedure on the resulting nodes. The splitting process is iteratively repeated on all nodes until some stopping criteria are achieved. Each split is defined according to the values of one or more inputs. The choice of the optimal split is generally

based on the maximization of the change in an impurity function. The result of this iterative process is a partition of the input space into disjoint subregions, called terminal nodes. This partition defines a classification rule: each terminal node is assigned to the most-represented class label in the node. The entire modelling process can be represented as a tree structure and it can be summarized as a set of "if-then" rules. Then, classification trees are easy to understand and to interpret.

In supervised setting, tree-based methods were firstly studied in the 1960s and 1970s (Kass, 1980; Hunt et al., 1966). A comprehensive study about classification tree algorithms was presented by Breiman et al. (1984). It introduced the popular CART algorithm. Since then, other classification tree algorithms have been developed such as for instance ID3 (Quinlan, 1986) and C4.5 (Quinlan, 1993). All the algorithms mentioned above are univariate classification tree algorithms. It means that, each node is determined according to the value of one single input. Multivariate classification trees algorithms, which split each node according to the value of a subset of inputs, have also been studied. For most of these algorithms, splits are linear and are often defined according to the value of a linear combination of a subset of inputs. Generally, the multivariate classification trees algorithms using linear splits differ in the way they search for the optimal split. For instance, CART-LC (Breiman et al., 1984), HHCART (Wickramarachchi et al., 2016) and OC1 (Murthy et al., 1993) use optimization techniques whereas other methods such as FACT (Wei-Yin Loh, 1988), QUEST (Loh & Shih, 1997) or LTDS (Li et al., 2003) use linear discriminant analysis (LDA) and some standard variable selection methods. These algorithms generally have higher accuracy and build a smaller tree size than the univariate classification trees (Lim et al., 2000). However, they suffers from two major drawbacks. First of all, the search for the optimal split often involves greedy algorithms such as the deterministic hill-climbing (Breiman et al., 1984) or the tabu search (Li et al., 2003). Secondly, the subset of inputs used to define a split is automatically selected by the algorithm with respect to an impurity criterion. Consequently, this input combination cannot sometimes make sense. Thus, multivariate classification trees are more difficult to interpret than univariate trees.

As mentioned previously, in many situations, inputs can have a known group structure. In this context, as far as we know, any multivariate classifica-

tion tree algorithm enables to take account of the input structure. This led us to develop the Tree Linear Discriminant Analysis algorithm (TLDA), a new multivariate classification tree algorithm involving linear splits and well adapted to grouped inputs. In this new tree-based approach, to split a node, the algorithm first estimates a split for each group of inputs by performing a LDA. Next, the algorithm selects the optimal split with respect to an impurity criterion. This splitting procedure is then repeated until pre-determined stopping criteria are satisfied. This results in a fully-grown tree which can be prone to overfitting. Thus, a pruning strategy is proposed to select an optimal tree. This new multivariate classification tree algorithm overcomes the two major drawbacks of the other multivariate classification tree algorithms. Indeed, the proposed algorithm is less time-consuming than classical multivariate classification tree algorithms. Besides, the algorithm does not need to perform a greedy search to determine the subsets of inputs used to define the optimal splits since the inputs are already structured in groups. Furthermore, interpretation is easy since the algorithm uses the group structure which make sense.

During the tree elaboration, as the depth of the tree increases, the number of observations in each created node becomes smaller and smaller whereas the size of each group of inputs does not change. Furthermore, it is well known that LDA does not perform well when the number of inputs is large relative to the number of observations (Shao et al., 2011; Friedman, 1989). To circumvent this problem, we propose to replace the classical LDA by a regularized linear discriminant method. We call this "modified" version of the TLDA algorithm the Tree Penalized Linear Discriminant Analysis algorithm (TPLDA).

Performances of the proposed algorithms are analysed through a detailed simulation study. TLDA and TPLDA are compared to CART since they are very similar to CART when inputs are not grouped. Moreover, they are also compared to GL which is one the reference method to elaborate classification rules with groups of inputs.

Moreover, TLDA and TPLDA are also assessed and compared to CART and GL on a real problem of tumors classification using gene expression data (Golub et al., 1999). Nowadays, the ability to classify tumors subtypes using gene expression data is still challenging. Indeed, the nature of both high

dimensionality and small size associated with gene expression data (i.e. a large number of variables relative to the much small number of observations) implies the use of features selection, clustering and/or regularized methods. Moreover, the resulted model must be easily understandable to allow both the identification of "marker" genes and the characterization of the tumor subtypes. Up until now, lots of approaches have been used such as committee neural networks (Sewak et al., 2009), support vector machine (SVM) (Guyon et al., 2002), k-nearest neighbors (Dudoit et al., 2002), etc. In this paper, we propose to apply TLDA and TPLDA in order to elaborate a classification achieving a good trade-off between prediction accuracy and interpretation while highlighting some relevant groups of genes.

To simply matters, in this paper, we restrict our attention to the classification problems involving binary response which already captures many of the main features of more general problems. Nonetheless, our algorithms can also be applied on classification problems involving more than two classes. Indeed, the splitting process allows to split a node into as many nodes as there are classes.

The paper is organized as follows. Section 2 describes the TLDA and TPLDA algorithms. Next, performances of the proposed algorithms are emphasized and are compared with the two competitive methods CART and GL in a simulation study in Section 3 and on the real Golub data set in Section 4. The time complexity of TLDA and TPLDA are detailed in Appendix A. Additional figures about the simulation study are given in Appendix B.

2. The Tree Group algorithms

Let $(\mathbf{X}, Y)$ be a random vector taking values in $\mathbb{R}^p \times \{0, 1\}$, where $\mathbf{X}$ is a vector of inputs and Y is the class label. Let $(\mathbf{X}_1, Y_1), \ldots, (\mathbf{X}_{n+m}, Y_{n+m})$ be independent copies of $(\mathbf{X}, Y)$. We randomly split the data into a training set $(\mathbf{X}_1, Y_1), \ldots, (\mathbf{X}_n, Y_n)$ of size n and a validation set $(\mathbf{X}_{n+1}, Y_{n+1}), \ldots, (\mathbf{X}_{n+m}, Y_{n+m})$ of size m. A discrimination rule is a measurable function $g : \mathbb{R}^p \times (\mathbb{R}^p \times \{0, 1\})^{n+m} \to \{0, 1\}$ which classifies a new observation $\mathbf{x} \in \mathbb{R}^p$ into the class $g\left(\mathbf{x}, (\mathbf{X}_1, Y_1), \ldots, (\mathbf{X}_{n+m}, Y_{n+m})\right)$. In what follows, we will write $g(\mathbf{x})$ for the sake of convenience. Consider the situation where $\mathbf{X}$ is structured into J known groups. For $j = 1, \ldots, J$, let p_j denote the cardinality of the

j-th group and $\mathbf{X}^j$ denotes the j-th group. To simplify matter, the J groups are ordered such that $\mathbf{X} = (\mathbf{X}^1, \ldots, \mathbf{X}^J)$.

Classification tree algorithms elaborate a model by recursively partitioning the input space. The entire modelling process can be represented as a tree and a classification rule can be deduced from the model. Figure 1 shows a tree representation. Let T be a classification tree built on the training set $(\mathbf{X}_1, Y_1), \ldots, (\mathbf{X}_n, Y_n)$. The i-th node and the j-th terminal node of T are denoted by N_i and $\widetilde{N}_j$ respectively. Let $\widetilde{N}(\mathbf{x})$ be the terminal node of T containing the input value $\mathbf{x}$ with $\mathbf{x} \in \mathbb{R}^p$. The classification rule g_T associated to the classification tree T is defined as:

$$g_T(\mathbf{x}) = \begin{cases} 1 & \text{if } n_{1,\widetilde{N}(\mathbf{x})} \geq n_{0,\widetilde{N}(\mathbf{x})} \\ 0 & \text{otherwise.} \end{cases}$$

$n_{k,\tilde{N}(\mathbf{x})} = \text{card}(R_{k,\tilde{N}(\mathbf{x})})$ is the cardinality of $R_{k,\tilde{N}(\mathbf{x})}$ which is the set $R_{k,\tilde{N}(\mathbf{x})} = \{i = 1, \ldots, n : \mathbf{X}_i \in \widetilde{N}(\mathbf{x}) \text{ and } Y_i = k\}$, $k = 0, 1$ of indices of the observations belonging $N(\mathbf{x})$.

2.1. The Tree-Linear Discriminant Analysis algorithm

A node N is usually obtained by splitting the input space according to the value of one or more inputs. In the case where the vector $\mathbf{X}$ of inputs is naturally structured into J groups, we propose the Tree Linear Discriminant Analysis algorithm (TLDA) which consists in the following iterative two-steps strategy: i) for each group, split the input space according to a linear combination of the inputs belonging the group, ii) select the best split with respect to an impurity criterion (which is equivalent to select the splitting group). These steps are now described in greater details.

Step 1: within group LDA. We first perform a LDA (Friedman, 1989) on each group $\mathbf{X}^j = (X_1^j, \ldots, X_{p_j}^j), j = 1, \ldots, J$ of inputs. A new observation $\mathbf{x} \in \mathbb{R}^p$ is then assigned to class 1 if

$$\hat{\delta}_{1,N}^j(\mathbf{x}) - \hat{\delta}_{0,N}^j(\mathbf{x}) \geq 0, \tag{1}$$

where $\hat{\delta}_{k,N}^j$ and $\hat{\delta}_{0,N}^j$ are the linear discriminant function estimated from the training set $(\mathbf{X}_1, Y_1), \ldots, (\mathbf{X}_n, Y_n)$ by:

$$\hat{\delta}_{k,N}^j(\mathbf{x}) = {\mathbf{x}^j}^\top (\widehat{\Sigma}_N^j)^{-1} \hat{\mu}_{k,N}^j - \frac{1}{2} (\hat{\mu}_{k,N}^j)^\top (\widehat{\Sigma}_N^j)^{-1} \hat{\mu}_{k,N}^j + \log \hat{\pi}_{k,N}, \quad k = 0, 1 \tag{2}$$

with $\top$ that stands for the transpose vector and

$$\begin{aligned}
\hat{\pi}_{k,N} &= \frac{n_{k,N}}{n_N}, \\
\hat{\mu}^j_{k,N} &= \frac{1}{n_{k,N}} \sum_{i \in R_{N,k}} \mathbf{X}^j_i, \\
\widehat{\Sigma}^j_N &= \frac{1}{n_N - 2} \sum_{k=0}^{1} \sum_{i \in R_{N,k}} (\mathbf{X}^j_i - \hat{\mu}^j_{k,N})(\mathbf{X}^j_i - \hat{\mu}^j_{k,N})^\top.
\end{aligned} \tag{3}$$

$n_N = \text{card}(R_N)$ is the cardinality of R_N which is the set $R_N = \{i = 1, \ldots, n : \mathbf{X}_i \in N\}$ of indices of the observations belonging N. LDA splits the node N into two child nodes:

$$\begin{aligned}
N_0(j) &= \{\mathbf{x} \in N \mid \hat{\delta}^j_{1,N}(\mathbf{x}) - \hat{\delta}^j_{0,N}(\mathbf{x}) < 0\} \\
&\text{and} \\
N_1(j) &= \{\mathbf{x} \in N \mid \hat{\delta}^j_{1,N}(\mathbf{x}) - \hat{\delta}^j_{0,N}(\mathbf{x}) \geq 0\}.
\end{aligned} \tag{4}$$

Step 2: choosing the splitting group. In the first step of the splitting process, the algorithm finds a linear split for each group of inputs. In order to select the splitting group, we use Gini impurity function which is estimated on the training set by

$$\widehat{\mathcal{I}}(N) = \hat{\pi}_{1,N}(1 - \hat{\pi}_{1,N}).$$

The algorithm selects the splitting group j, $j \in \{1, \ldots, J\}$, which maximizes the impurity decrease defined by

$$\Delta_j(N) = \widehat{\mathcal{I}}(N) - \frac{n_{N_0(j)}}{n_N} \hat{\mathcal{I}}(N_0(j)) - \frac{n_{N_1(j)}}{n_N} \hat{\mathcal{I}}(N_1(j)). \tag{5}$$

At the very beginning of the whole procedure, steps 1 and 2 are applied to partition the entire data space into two sub-regions. Then, these steps are repeated recursively on each sub-region until every sub-region denoted by N satisfies one of the following stopping criteria:

- N is homogeneous (or near so) with respect to a particular class, i.e.

$$\hat{\pi}_{1,N} < \epsilon \quad \text{or} \quad \hat{\pi}_{1,N} > 1 - \epsilon,$$

for a small given value ϵ,

- no further partition can reduce the impurity of N, that is:

$$\Delta_j(N) = 0, \text{ for all } j = 1, \ldots, J.$$

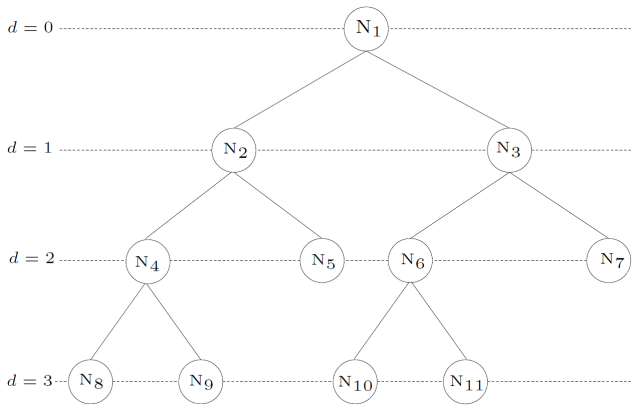


Figure 1: Example of a classification tree. Circles indicate the nodes. d refers to the depth of the nodes. Here $D(T) = 3$. The terminal nodes are: $\widetilde{N}_1(T) = N_8$, $\widetilde{N}_2(T) = N_9$, $\widetilde{N}_3(T) = N_{10}$ and $\widetilde{N}_4(T) = N_{11}$. The node N_1 denotes the tree root.

Iterating the splitting process described above yields a fully-grown tree $T_{\max}$. It is well known that maximal classification trees are generally not optimal with respect to any performance criterion (such as the misclassification error) (Breiman et al., 1984). Indeed, an excessive large number of nodes is prone to overfitting. Thus, we propose a pruning strategy that allows to select an optimal tree. This strategy is described below.

Pruning strategy. Let T be a subtree of $T_{\max}$, with terminal nodes $\widetilde{N}_1(T), \ldots, \widetilde{N}_{|T|}(T)$, where $|T|$ is the number of terminal nodes of T. Let $d(N)$ be the depth of node N, that is, the number of conditions that an observation $\mathbf{x} \in \mathbb{R}^p$ has to satisfy from the root to the node N. Then, the depth $D(T)$ of the tree T is defined as:

$$D(T) = \max_{\ell=1,\ldots,|T|} d(\widetilde{N}_\ell(T)).$$

See Figure 1 for an illustration of the notions of nodes, terminal nodes and depth. We define the sequence

$$N_1 = T_0 \subset T_1 \subset \ldots \subset T_{D(T_{\max})} = T_{\max} \tag{6}$$

of nested trees such that T_k, $k = 1, \ldots, D(T_{\max})$ is the subtree of $T_{\max}$ which maximizes over all trees $T \subset T_{\max}$ the quantity

$$\sum_{\ell=1,\ldots,|T|} d(\widetilde{N}_\ell(T)) \quad \text{subject to} \quad d(\widetilde{N}_\ell(T)) \leq k.$$

In other words, T_k is the deeper subtree of $T_{\max}$ whose terminal nodes have a depth less than or equal to k. For example, Table 1 gives the terminal nodes for the sequence of subtrees of the tree described in Figure 1.

Table 1: Terminal nodes for the subtrees in Figure 1.

Tree	**Terminal Nodes**
T_0	N_0
T_1	N_2, N_3
T_2	N_4, N_5, N_6, N_7
T_3	$N_8, N_9, N_5, N_{10}, N_{11}, N_7$

Each tree T_k, $k = 1, \ldots, D(T_{\max})$ defines a classification rule g_k:

$$g_k(\mathbf{x}) = \begin{cases} 1 & n_{1,\widetilde{N}(\mathbf{x},T_k)} \geq n_{0,\widetilde{N}(\mathbf{x},T_k)} \\ 0 & \text{otherwise,} \end{cases} \tag{7}$$

where $\widetilde{N}(\mathbf{x}, T_k)$ is the terminal node of T_k containing $\mathbf{x}$. Note that the classification rules g_k, $k = 1, \ldots, D(T_{\max})$ depend only on the training set $(\mathbf{X}_1, Y_1), \ldots, (\mathbf{X}_n, Y_n)$. Our pruning strategy selects the rule $g_{\hat{K}}$ which minimizes the misclassification error $\mathbf{P}(g_k(\mathbf{X}) \neq Y)$. This error is estimated on the validation set $(\mathbf{X}_{n+1}, Y_{n+1}), \ldots, (\mathbf{X}_{n+m}, Y_{n+m})$. Precisely, we choose

$$\hat{K} \;=\; \underset{k=1,\ldots,D_{T_{\max}}}{\operatorname{argmin}} \frac{1}{m} \sum_{i=n+1}^{n+m} \mathbf{1}_{g_k(\mathbf{X}_i) \neq Y_i}.$$

The final tree retained by our procedure is the subtree $T_{\widehat{K}}$. The cardinality of the sequence $\{g_1, \ldots, g_{D(T_{\max})}\}$ of classifiers is finite and bounded by the size n of the training set. Therefore, using classical empirical minimization tools (Devroye et al., 2013, chapter 26), we show that the selected rule $g_{\hat{K}}$ satisfies the following inequality:

$$\mathbb{E}\left[|\mathbf{P}(g_{\widehat{K}}(\mathbf{X}) \neq Y) - \inf_{k \in \{1,\ldots,D(T_{\max})\}} \mathbf{P}(g_k(\mathbf{X}) \neq Y)| \right] \leq 2\sqrt{\frac{\log(2n)+1}{2m}} \tag{8}$$

where m is the test sample size. Thus, since in most cases of interest $\log(m) \ll n$, inequality (8) means that the selected classification rule $g_{\hat{K}}$ classifies as well as the best classifier in the sequence $\{g_1, \ldots, g_{D(T_{\max})}\}$ (with respect to the misclassification error).

Remark 2.1.

- During the splitting procedure, when choosing the splitting group (Step 2, page 7) other impurity criteria, such as the information criterion, could be used.

- The time complexity of TLDA at a non-terminal node N of size n_N is in the worst case $\mathcal{O}(Jn_N p_{\max} t_{\min})$ where J refers to the number of groups, $p_{\max} = \max_j(p_j)$ with p_j denoting the size of $\mathbf{X}^j$ and $t_{\min} = \min(p_{\max}, n_N)$. The computation is detailed in Appendix A. As TLDA performs a group selection rather than a variable selection, the time complexity increases when the number of groups J increases. However, whatever the number of groups J and the size of the largest groups $p_{\max}$, TLDA is less consuming than most of multivariate classification algorithms such as for instance HHCART (time complexity $= \mathcal{O}(n_N^2 p^3)$ with $p = \sum_{j=1}^{J} p_j$ is the number of inputs) and OC1 (time complexity $= \mathcal{O}(n_N^2 \log(n_N) p)$) (Wickramarachchi et al., 2016).

The TLDA algorithm is illustrated by the following simple example:

Example 2.1. Let's consider the random vector $(\mathbf{X}, Y)$ which takes values in $\mathbb{R}^2 \times \{0, 1\}$ with

$$\begin{gathered} X_i \sim \mathcal{N}(0,1), \quad i = 1, 2 \\ \text{and} \\ \mathcal{L}(Y \mid \mathbf{X} = \mathbf{x}) = \begin{cases} \mathcal{B}(0.9) & \text{if} \quad x_2 > 2(x_1)^2 + 0.20 \quad \text{or} \quad x_2 < 0.5 + x_1 \\ \mathcal{B}(0.1) & \text{otherwise.} \end{cases} \end{gathered} \tag{9}$$

The aim is to predict the class label Y according to the unique and single group $\mathbf{X}^1 = \mathbf{X} = (X_1, X_2)$. In this scenario, the Bayes classification rule $g^*(\mathbf{x})$, i.e. the rule which minimizes the misclassification error $\mathbf{P}(g(\mathbf{X}) \neq Y)$ is defined by:

$$g^*(\mathbf{x}) = \begin{cases} 1 & \text{if} \quad x_2 > 2(x_1)^2 + 0.20 \quad \text{or} \quad x_2 < 0.5 + x_1 \\ 0 & \text{otherwise.} \end{cases} \tag{10}$$

CART and TLDA have been applied on a training sample of 50 observations. For TLDA, a validation set of 50 observations is used to perform the proposed pruning strategy based on the depth while CART maximal tree has been pruned by applying the classical cost-complexity pruning (Breiman et al., 1984). Finally, the predictive performances of the two final trees have been measured by the area under the ROC curve (AUC) estimated on an independent test sample of 1000 observations. Here, TLDA allows to elaborate a less complex partition of the input space without lost of accuracy (Figure 2).

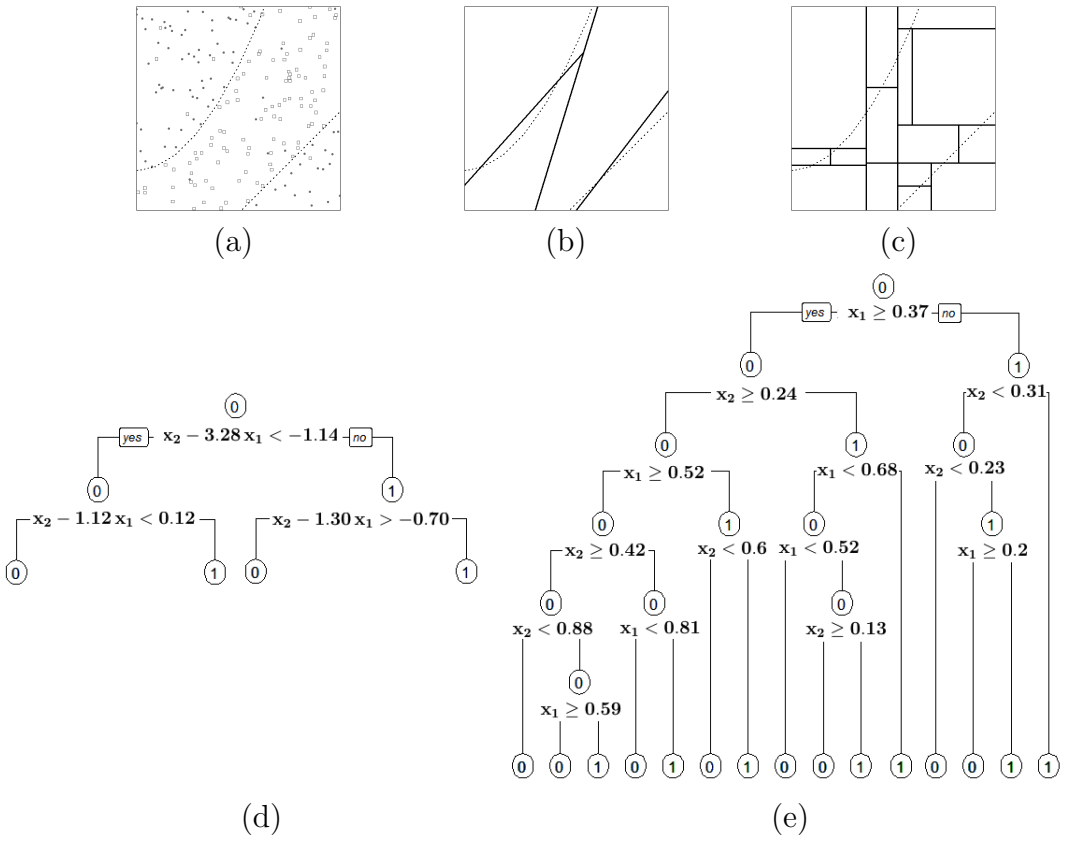


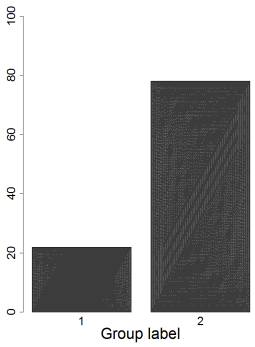
Figure 2: Example 1 - a simple binary classification problem in $\mathbb{R}^2$. From left to right: (a) representation of 200 observations defined by model (9), (b) a TLDA partition (AUC=0.90), (c) a CART partition (AUC=0.89), (d) the tree associated to the TLDA partition, (e) the tree associated to the CART partition. On each graph, the Bayes decision boundaries are represented by the two dotted boundaries. For the trees, circles defines the nodes and the figure in each node are the node label. The splitting rule is indicated below each node.

LDA is known to be very sensitive to the sample size and tends to not perform well in high-dimension, i.e. when in a node N some groups are large compared to the number of observations (Shao et al., 2011; Friedman, 1989; Xu et al., 2009; Bouveyron et al., 2007). Yet, the splitting procedure described above (page 6) creates nodes that becomes smaller and smaller whereas the sizes of inputs groups remain unchanged. Then LDA may not be appropriate for estimating recursively the hyperplane splits. Example 2.2 illustrates it.

Example 2.2. Consider the previous example 2.1 with an additional group of twenty noisy inputs. This second group is denoted by the random vector $\mathbf{X}^2$ which takes values in $\mathbf{R}^{20}$:

$$\mathbf{X} = (\mathbf{X}^1, \mathbf{X}^2) \quad \text{and} \quad X_i \sim \mathcal{N}(0,1), \quad i = 1, \ldots, 22.$$

Inputs are considered mutually independent. The conditional distribution $\mathcal{L}(Y \mid \mathbf{X} = \mathbf{x})$ of the response Y, the Bayes classification rule $g^*(\mathbf{x})$, the size of the training, validation and test sets remain unchanged compared to Example 2.1 (see equations (9) and (10)). The aim is to predict the class label Y according to the two groups $\mathbf{X}^1$ and $\mathbf{X}^2$. Figure 3 outlines the results based on 500 repetitions of this classification problem. In this example, TLDA mainly selects the second groups $\mathbf{X}^2$ which is ten times larger that the first and single predictive group $\mathbf{X}^1$. Consequently, as the pruning procedure deletes the uninformative nodes, the final TLDA tree is trivial (i.e. the depth of the final tree equals zero) in 51% of the cases and so the predictive performance of TLDA are low.



(a)

	TLDA	**CART**
AUC	0.50 [0.50 ; 0.92]	0.62 [0.60 ; 0.68]
Tree depth	0 [0 ; 1]	4 [2 ; 5]

(b)

Figure 3: Example 2 - Sensitivity of LDA to the group size. From left to right: (a) group selection frequencies for the first split for TLDA (in %), (b) median of the AUC and the tree depth with the first and the third quartiles in brackets. Note that 51% of the final TLDA trees are trivial.

To address the issue illustrated by the previous example, we propose to use a regularized discriminant method instead of LDA. This modified version of the TLDA algorithm is called the Penalized Linear Discriminant Analysis algorithm (TPLDA) and is introduced below.

2.2. The Penalized Linear Discriminant Analysis algorithm

It is established that performing LDA is equivalent to performing Fisher's linear discriminant analysis (FDA) (Friedman et al., 2001). For instance, consider the split of the node N based on the group $\mathbf{X}^j$ (Step 1, page 6). FDA seeks a one-dimensional projection of the observations in the node N that maximizes the ratio of the between-class variance to the within-class variance. Then, FDA solves the problem

$$\text{maximize}_{\beta^j \in \mathbb{R}^p} \left\{ (\beta^j)^\top \widehat{B}_N^j \, \beta^j \right\} \text{ s.t. } (\beta^j)^\top \widehat{\Sigma}_N^j \, \beta^j \leq 1 \tag{11}$$

where $\widehat{\Sigma}_N^j$ denotes the standard estimate of the within-class covariance matrix of $\mathbf{X}^j$ in the node N defined in equation (3) and $\widehat{B}_N^j$ is the standard estimate for the between-class covariance matrix of $\mathbf{X}^j$ in the node N:

$$\widehat{B}_N^j = \frac{1}{n_N - 2} \sum_{k=0}^{1} \sum_{i \in R_{N,k}} (\hat{\mu}_N^j - \hat{\mu}_{k,N}^j)(\hat{\mu}_N^j - \hat{\mu}_{k,N}^j)^\top. \tag{12}$$

The solution of problem (11) is denoted $\hat{\beta}^j$ and is called the discriminant vector. A new observation $\mathbf{x} \in \mathbb{R}^p$ is assigned to class 1 if $\mathbf{x}^j$ is closer to the centroid $\hat{\mu}_{1,N}^j$ of the class $Y = 1$ than to the centroid $\hat{\mu}_{0,N}^j$ of the class $Y = 1$ in the projected space, which is equivalent to

$$\begin{aligned} \|\hat{\beta}^{j\top} \mathbf{x}^j - \hat{\beta}^{j\top} \hat{\mu}_{1,N}^j\|_2 + \log(\hat{\pi}_{1,N}) &\geq \|\hat{\beta}^{j\top} \mathbf{x}^j - \hat{\beta}^{j\top} \hat{\mu}_{0,N}^j\|_2 + \log(\hat{\pi}_{0,N}) \\ \beta^{j\top} \left(\mathbf{x}^j - \frac{(\hat{\mu}_{1,N}^j - \hat{\mu}_{0,N}^j)}{2} \right) + \log\left(\frac{\hat{\pi}_{1,N}}{\hat{\pi}_{0,N}} \right) &\geq 0, \end{aligned} \tag{13}$$

with $\hat{\mu}_{k,N}^j$, $k = 0, 1$ denoting the empirical estimate of the prior probability of belonging the class k in the node N. Then, as LDA (4), FDA splits the

node N into two child nodes:

$$N_0(j) = \left\{ \mathbf{x} \in N \mid \hat{\beta}^{j\top} \left(\mathbf{x}^j - \frac{(\hat{\mu}^j_{1,N} - \hat{\mu}^j_{0,N})}{2} \right) + \log \left(\frac{\hat{\pi}_{1,N}}{\hat{\pi}_{0,N}} \right) < 0 \right\}$$
$$\text{and} \tag{14}$$
$$N_1(j) = \left\{ \mathbf{x} \in N \mid \hat{\beta}^{j\top} \left(\mathbf{x}^j - \frac{(\hat{\mu}^j_{1,N} - \hat{\mu}^j_{0,N})}{2} \right) + \log \left(\frac{\hat{\pi}_{1,N}}{\hat{\pi}_{0,N}} \right) \geq 0 \right\}.$$

As illustrated in Example 2.2, FDA or LDA performs badly in high-dimensional situations. Therefore, FDA may not be appropriate for estimating recursively the hyperplane splits. To address this issue, we propose to use regularized discriminant analysis (a.k.a penalized Fisher's linear discriminant analysis, see Witten & Tibshirani, 2011) instead of the LDA. In penalized Fisher's linear discriminant analysis (PLDA), the FDA problem is modified by imposing a L_1-penalty on the discriminant vector β^j. The penalized discriminant vector $\hat{\beta}^j_{pen}$ is then the solution to the PLDA problem defined as:

$$\text{maximize}_{\beta^j \in \mathbb{R}^p} \left\{ (\beta^j)^\top \widehat{B}^j_N \, \beta^j - \lambda_j \sum_{l=1}^{p_j} |\widehat{\sigma}^j_{N,l} \beta^j_l| \right\} \text{ s.t. } (\beta^j)^\top \widetilde{\Sigma}^j_N \, \beta^j \leq 1 \tag{15}$$

where $\widetilde{\Sigma}^j_N$ is the diagonal positive estimate of the within-group covariance matrix of $\mathbf{X}^j$ in the node N:

$$\widetilde{\Sigma}^j_N = \text{diag} \left((\hat{\sigma}^j_{N,1})^2, \ldots, (\hat{\sigma}^j_{N,p_j})^2 \right) \tag{16}$$

with $\hat{\sigma}^j_{N,l}$, $l = 1, \ldots, p_j$ denoting the within-class standard deviation estimate for the l-th input of X^j. This diagonal estimate has been used for instance by Friedman (1989) and shown good performances in high-dimensional situations (Bickel & Levina, 2004; Dudoit et al., 2002). In the PLDA problem (15), λ_j is a shrinkage parameter taking values in $\mathbb{R}^+$: the larger λ_j is, the more shrunk the components of β^j are. Moreover, with the inclusion of the within-class standard deviation estimates $\widehat{\sigma}^j_{N,l}$, $l = 1, \ldots, p_j$ in the penalty, the inputs in the group $\mathbf{X}^j$ that vary more within each class $Y = 1$ and $Y = 0$ are more penalized. Consequenthly, large values for λ_j and $\widehat{\sigma}^j_{N,l}$, $l = 1, \ldots, p_j$ can force some components of β^j_{pen} to be set to zero. In practice, the value of λ^j is chosen by cross-validation from a set of guided values provided by the

user.

As with FDA, a new observation $\mathbf{x} \in \mathbb{R}^p$ is assigned to the nearest class centroid in the projected space (see equation (13)). Thus, PLDA splits the node N into two child nodes:

$$N_0(j) = \left\{ \mathbf{x} \in N \mid \hat{\beta}_{pen}^{j\top} \left(\mathbf{x}^j - \frac{(\hat{\mu}_{1,N}^j - \hat{\mu}_{0,N}^j)}{2} \right) + \log \left(\frac{\hat{\pi}_{1,N}}{\hat{\pi}_{0,N}} \right) < 0 \right\}$$
$$\text{and} \tag{17}$$
$$N_1(j) = \left\{ \mathbf{x} \in N \mid \hat{\beta}_{pen}^{j\top} \left(\mathbf{x}^j - \frac{(\hat{\mu}_{1,N}^j - \hat{\mu}_{0,N}^j)}{2} \right) + \log \left(\frac{\hat{\pi}_{1,N}}{\hat{\pi}_{0,N}} \right) \geq 0 \right\}.$$

The use of PLDA instead of LDA leads to the modified version of the TLDA algorithm called the Tree Penalized Linear Discriminant Analysis algorithm (TPLDA). As TLDA, TPLDA firstly uses a two-step splitting procedure to built a maximal tree and next applies the pruning strategy described above. The pruning procedure remains unchanged compared to TLDA whereas the splitting process is slightly different:

Step 1: within group PLDA. Before applying a PLDA on a given group $\mathbf{X}^j = (X_1^j, \ldots, X_{p_j}^j), j = 1, \ldots, J$ of inputs in a node N, a K-fold cross-validation is performed to select the value of the shrinkage parameter λ_j. indeed, from a set of L guided values, the algorithm selects the value for λ_j which maximizes the cross-validated estimate of the decrease in impurity (5). Next, PLDA can be performed on $\mathbf{X}^j$ which results in the split of N into the two child nodes $N_0(j)$ and $N_1(j)$ defined by equation (17).

Step 2: choosing the splitting group. As TLDA, TPLDA uses Gini impurity function and selects the group $j, j = 1, \ldots, J$ which maximizes the impurity decrease in the node N (equation (5)).

Remark 2.2.

- The term $\log \left(\frac{\hat{\pi}_{1,N}}{\hat{\pi}_{0,N}} \right)$ in equations (14) and (17) allows to take account of the empirical estimates of the prior probabilities of class membership $\hat{\pi}_{0,N}$ and $\hat{\pi}_{1,N}$.

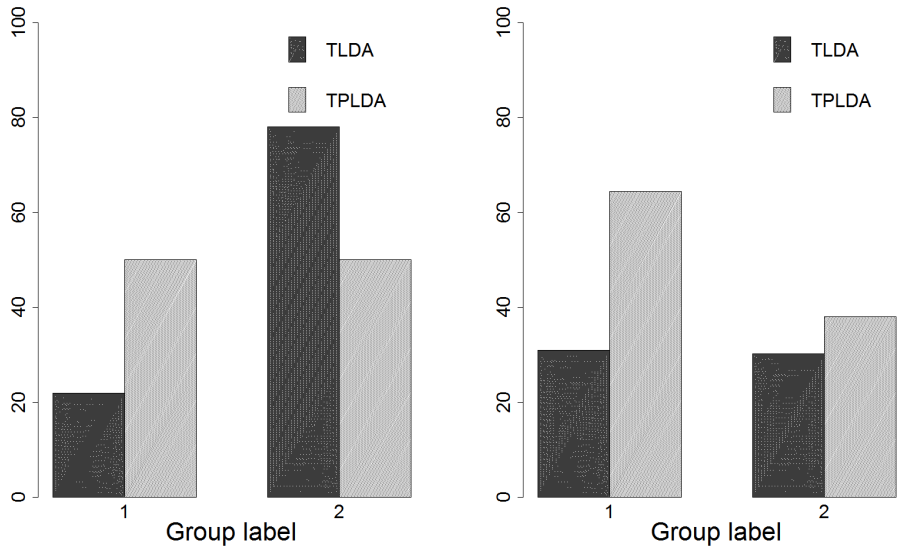
- In practice, the PLDA problem (15) is solved by using a minimization algorithm (Hunter & Lange, 2004). More details about PLDA are available in the original paper of Witten & Tibshirani (2011). A R package is also available (Witten, 2015).

- In the PLDA problem (15), if λ_j is equal to zero and the diagonal positive estimate $\widetilde{\Sigma}_N^j$ of the within-group covariance matrix of $\mathbf{X}^j$ in the node N (equation (16)) is replaced by the standard estimate $\widehat{\Sigma}_N^j$ (equation (3)), the PLDA problem (15) corresponds to the FDA problem (11).

- In a node N, if the inputs in the group $\mathbf{X}^j$ are mutually independent or if the size p_j of the group $\mathbf{X}^j$ is 1, then $\tilde{\Sigma}_N^j = \widehat{\Sigma}_N^j$.

- The time complexity of TPLDA at a node N of size n_N is in the worst case $\mathcal{O}\left(JLKn_N p_{\max}^2\right)$ with J referring to the number of groups, K being the number of folds in the cross-validation, L denoting the number of guided values for λ^j in the cross-validation and $p_{\max} = \max_j(p_j)$ with p_j being the size of $\mathbf{X}^j$. The computation is detailed in Appendix A. As TPLDA performs, for each group $\mathbf{X}^j$, $j = 1, \ldots, J$, a K-fold cross-validation to calibrate λ_j, the time complexity increases as an increasing function of L, K and J. However, as the inequality $K \leq n_N$ is always satisfied, TPLDA remains less time consuming than lots of multivariate classification tree algorithms such as for instance HHCART (time complexity $= \mathcal{O}\left(n_N^2 p^3\right)$ with $p = \sum_{j=1}^J p_j$ is the total number of inputs) and OC1 (time complexity $= \mathcal{O}\left(n_N^2 \log(n_N) p\right)$), excepted in very small nodes (i.e. $Lp_{\max} > \log(n_N)$).

Example 2.3. TPLDA is applied on the classification problem introduced in Example 2.2. Figure 4 describes the results based on 500 repetitions of this classification problem. Compared to TLDA, TPLDA is less sensitive to the sizes of the groups. TPLDA selects more frequently the true predictive group $\mathbf{X}^1$ than the noisy group $\mathbf{X}^2$ and elaborates a trivial tree in only 28% cases whereas TLDA builts a trivial tree in 51% cases. In this case, TPLDA outperforms both TLDA and CART (see Figure 3, page 12).



(a) (b)

	TPLDA
AUC	0.64 [0.50 ; 0.75]
Tree depth	1 [0 ; 2]

(c)

Figure 4: Example 3 - Application of the TPLDA algorithm to the problem introduced in Example 2.2. From left to right: (a) Group selection frequencies for the first split (in %), (b) Group selection frequencies (in %), (c) Median of the AUC and the tree depth with the first and the third quartiles in brackets for TPLDA. Note that 28% of the final TPLDA trees are trivial.

3. Numerical experiments

In this section, several numerical experiments are presented. In all the experiments, the TLDA and TPLDA algorithms are compared to CART and GL. Experiments are based on a model inspired by Friedman et al. (2001) and described below.

Presentation of the general simulation design

First, a sample of outcome variable Y is simulated from a balanced Bernoulli distribution $Y \sim \mathcal{B}(0.5)$. The vector $\mathbf{X}$ of inputs is defined conditionally

to the variable Y and is structured in J groups: $\mathbf{X} = (\mathbf{X}^1, \ldots, \mathbf{X}^J)$. When $Y = 0$, for $j = 1, \ldots J$, every component X_l^j, $l = 1, \ldots, p_j$ of the vector $\mathbf{X}^j$, follows a standard Gaussian distribution:

$$\mathcal{L}(X_l^j \mid Y = 0) \;=\; \mathcal{N}(0,1), \quad l = 1, \ldots, p_j.$$

When $Y = 1$, every component X_l^j, $l = 1, \ldots, p_j$ of the vector $\mathbf{X}^j$, is defined conditionally to the value of the standard uniform random variable U:

$$\mathcal{L}(X_l^j \mid Y = 1, U = u) = \begin{cases} \mathcal{N}(-\mu_j, 1) & \text{if } u < u_1; \\ \mathcal{N}(\mu_j, 1) & \text{if } u_1 \leq u < u_2; \\ \mathcal{N}(0,1) & \text{otherwise.} \end{cases} \tag{18}$$

where u_1, u_2 are two fixed real numbers such that $0 \leq u_1 < u_2 \leq u_1 + u_2 \leq 1$. The components $\mu_j \geq 0$, $j = 1, \ldots J$ can be interpreted as the discriminatory power of the group j: the higher the value of μ_j is, the more the class-conditional distributions of $\mathbf{X}^j$ differ. Note that if μ_j is zero, whatever the values of Y and U, all inputs in group $\mathbf{X}^j$ are distributed according to a standard Gaussian distributions: $\mathcal{L}(X_l^j | Y = 0) = \mathcal{L}(X_l^j | Y = 1, U = u) = \mathcal{N}(0,1)$, $l = 1, \ldots, p_j$. Then, when μ_j is zero, the group $\mathbf{X}^j$ is not relevant to predict Y. The vector of the components μ_j $(j = 1, \ldots J)$ is denoted $\mu = (\mu_1, \ldots, \mu_J)$.

The covariance between two inputs X_l^j and $X_{l'}^{j'}$ $(j, j' = 1, \ldots, J, l = 1, \ldots, p_j, l' = 1, \ldots, p_{j'})$ is defined according to both the group belonging by each of the two inputs and the distance $d_1(X_l^j, X_{l'}^{j'})$ between the two inputs:

$$\mathrm{Cov}\left(X_l^j, X_{l'}^{j'}\right) = \begin{cases} c_w^{\,d_1(X_l^j, X_{l'}^{j'})} & \text{if } j = j', \\ c_b^{\,d_1(X_l^j, X_{l'}^{j'})} & \text{otherwise.} \end{cases}$$

where c_w and c_b are two real numbers such that $0 \leq c_w, c_b < 1$ and

$$d_1(X_l^j, X_{l'}^{j'}) = \sum_{i=\min(j,j')}^{\max(j,j')-1} p_i + (l - l')\mathbf{1}_{\min(j,j')=j'} + (l' - l)\mathbf{1}_{\min(j,j')=j}.$$

Thus, in this simulation design, the group structure of the inputs comes from both the discriminatory power μ_j, $j = 1, ..., J$ of the inputs and the block structure of the covariance matrix of $\mathbf{X}$.

For all experiments, $n+m+q$ observations are randomly divided into three independent subsamples: a training sample of size n, a validation sample of size m and a test sample of size q. The size of the training and the validation samples are set to be equal. In all experiments, the test sample size is set to $q = 1000$. Moreover, $J = 10$ groups are simulated and the vector μ is set to $\mu = (1.25, 0, 1, 0, 0.75, 0, 0.5, 0,\ 0.25, 0)$. In this way, only groups with an odd index are relevant and the discriminatory power of each even group (i.e. in each relevant group) is a linear decreasing function of the group index. Moreover, we choose $(u_1, u_2) = (0.25, 0.90)$. Four experiments are considered here by varying:

- the sizes n and m of the training set and the validation set,
- the group sizes p_j, $j = 1, \ldots, J = 10$,
- the correlations c_w and c_b.

In this way, we embrace a variety of situations: independent and ungrouped inputs, correlated groups of moderate and equal size, large correlated groups and correlated groups with large noisy groups.

In each experiment, the algorithms are compared and assessed on two main criteria: 1) the predictive performance of the classification rule and 2) the ability to identify the true relevant groups/inputs. The first criterion is assessed by using the AUC computed on the test set. The ability to identify the true relevant groups/inputs is measured by the group selection frequencies which has not the same meaning for all the assessed algorithms. As CART ignores the group structure, for this algorithm, the selection frequency of a given group is defined as the number of times that at least one input in the group is included in the final tree. For TLDA, TPLDA and GL, as the algorithms use the group structure to build a classification rule, the selection frequency of a given group refers to the number of times that a group is included at least once in the final model. Due to these two different definitions, no direct comparison is made between the group selection frequencies of CART algorithm and those of the other algorithms.

Furthermore, the complexity of the classification rule is also studied by using two criteria: the tree depth for the classification tree algorithms (i.e. TLDA, TPLDA and CART) and the number of groups included in the model for

GL. For the classification tree algorithms, interpretation of a large tree is harder than the one of a small tree. So larger a tree is, more complex the classification rule is. For GL, the complexity increases with the number of groups included in the model.
Finally, for the classification tree algorithms, the decreasing of the misclassification error is displayed according to the tree depth in the training and the validation set.
All criteria are averaged over 500 iterations of each experiment.

Remark 3.1. In this simulation design, the covariance structure looks like the one of gene expression data: genes included in a same biological pathway are correlated, and the correlation decreases as a function of the distance between any two genes.

Experiment 1: ungrouped inputs

This first scenario illustrates the similarity of TLDA and TPLDA to CART when inputs are not grouped. The scenario follows the general simulation design presented before. Each group includes only a single input (i.e. $p_j = 1$, for $j = 1, \ldots, 10$), so a group is an input. The size of the training and the validation sets are set to $n = m = 500$. Moreover, inputs are not correlated (i.e. $c_w = 0$ and $c_b = 0$) in order to assess the ability of the algorithms to identify the true relevant groups/inputs when data are mutually independent and not grouped.
Results. Figure 5 displays the AUC distribution, the tree-depth distribution and the group selection frequencies over the 500 iterations. Note that as all groups include only a single input, CART selection frequency of a given group has the same meaning than the selection frequency of the group for TLDA, TPLDA and GL. In this situation, TLDA and TPLDA lead to almost the same results and can be then used interchangeably. These big similarities can be explained by the fact that as the size of the groups is 1: the FDA (or LDA) problem and the PLDA problem are identical (see Remark 2.2).
Moreover, the two proposed algorithms identify the same groups (or inputs) and have similar predictive performances than CART (Figures 5a, Figures 5c and Table 2). Even though, CART elaborates larger trees. TLDA and TPLDA do not exactly give the same results than CART because they do not optimize the same criteria. Indeed, to split a node, CART tries to find the splitting input and the value for this inputs that maximizes the decrease in impurity in the node (see Breiman et al., 1984). In our approach, the

splitting process is composed of two steps (see the description of the two-step splitting procedure page 6). First, TLDA and TPLDA estimate a split for every group based on a maximization of the ratio of the between-class covariance matrix and the within-group covariance matrix (see equation (11) and equation (15)). Next, they select the split that maximizes the decrease in impurity in the node (see equation (5)). So, TLDA and TPLDA do not directly maximizes the decrease in impurity.

Table 2: Experiment 1. AUC distribution according to the methods.

	TLDA	**TPLDA**	**CART**	**GL**
AUC	0.66	0.66	0.67	0.64
	[0.64;0.67]	[0.65;0.68]	[0.65;0.68]	[0.62;0.65]

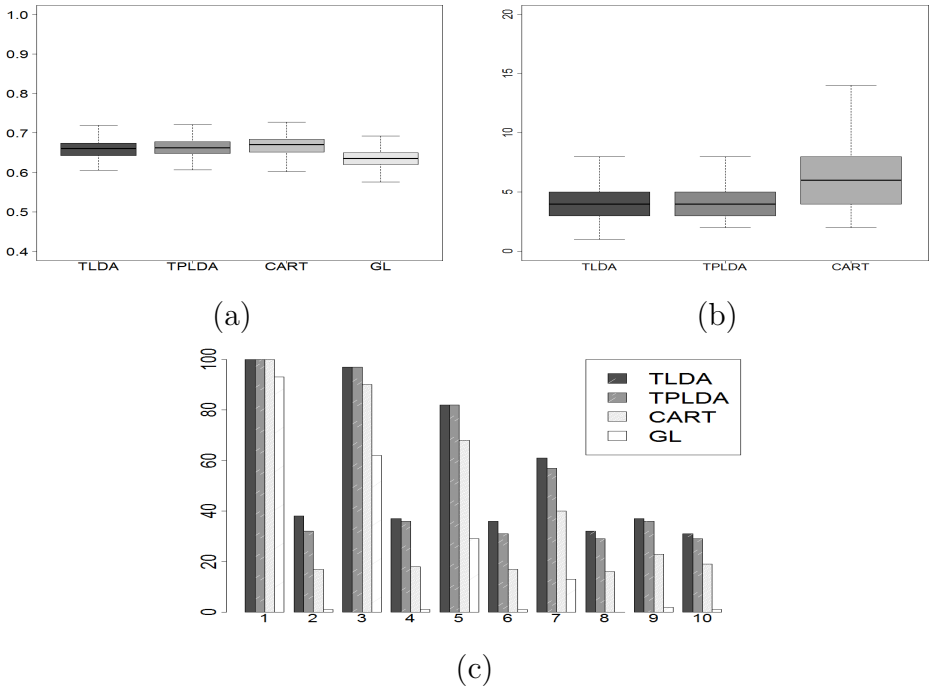


Figure 5: Experiment 1, ungrouped inputs. From top left to bottom right: (a) Boxplots of the AUC, (b) Boxplots of the tree depth, (c) Groups selection frequencies according to the method.

Table 3: Experiment 1. Number of groups included in the GL model.

Model size	0	1	2	≥ 3
Frequency (in %)	1.80	31.00	39.80	27.4

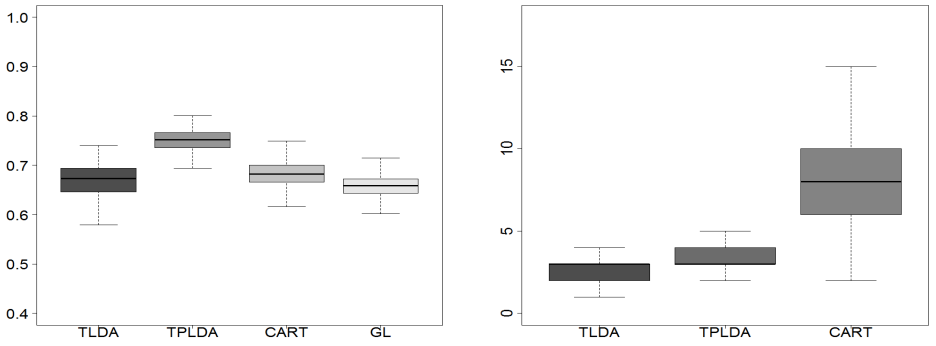
Experiment 2: groups of moderate and equal size

In contrast to the previous simulation, in this scenario, inputs are structured in groups of equal and moderate sizes. Moreover, correlation between and within groups are considered. This scenario is also simulated according to the general simulation design presented before, by taking:

- $n = m = 500$,
- $p_j = 10$ for $j = 1, \ldots 10$,
- $(c_w, c_b) = (0.85, 0.8)$.

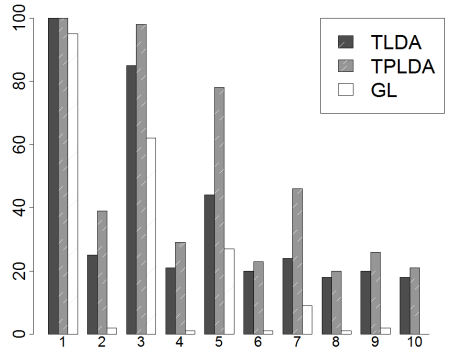
Results. As previously, TLDA, TPLDA and GL select more the relevant groups than the noisy groups and the selection frequency of a given group behaves as an increasing function of the discriminatory power of the group (Figure 6c and Figure 6d). Moreover, TPLDA identifies and selects more relevant group than TLDA and GL (Figure 6c). It selects the three most predictive groups in about 80% of the simulations and the fourth most predictive group in more than 40% of the cases whereas TLDA (GL respectively) selects the three most predictive groups in less than 50% (less than 30% respectively) and the fourth most relevant group in less than 40% (in less than 10% respectively). Even though the group selection frequencies using CART and TPLDA are not comparable because CART selects individual inputs whereas TPLDA select a group of inputs, the group selection frequencies of the two algorithms are very similar. Then, CART identifies inputs belonging the groups selected by TLDA and this in the same frequencies.
Figure 7a and Figure 7b display the misclassification error according to the tree depth in the training and the validation set for TLDA, TPLDA and CART. In both the training set and the validation set, the misclassification error decreases quicker with TPLDA and TLDA than with CART because the two proposed Tree Group algorithms use multivariate splits which are more informative (Lim et al., 2000). This also explains the smaller size of the final tree for TLDA and TPLDA compared to CART (Figure 6b). Then,

TPLDA and TLDA trees are more easily interpretable than CART because first splits are defined according to the groups which makes more sense that inputs taken individually and secondly final trees are smaller.

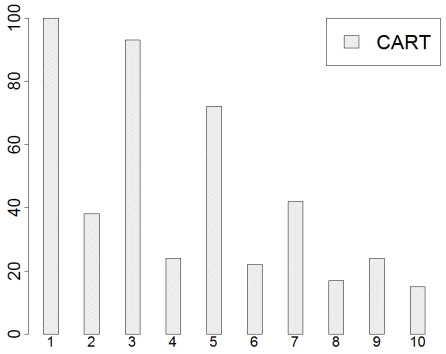


(a)

(b)



(c)

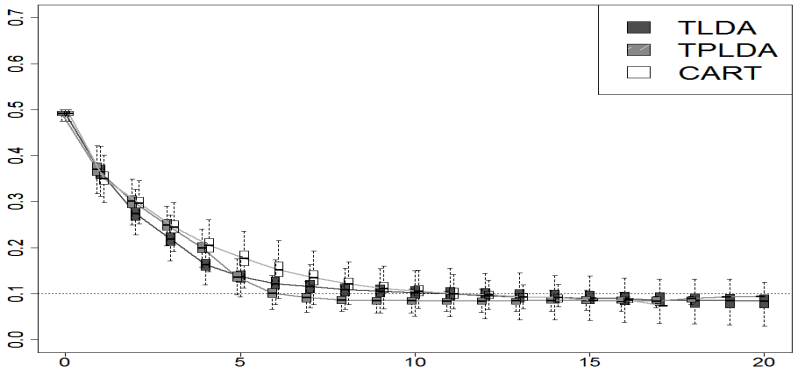


(d)

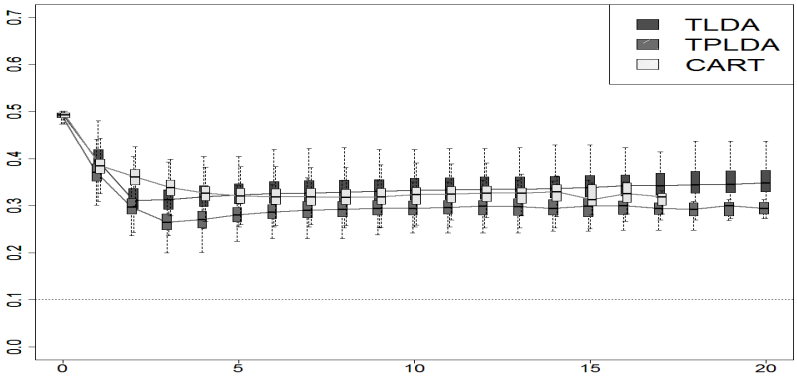
	TLDA	**TPLDA**	**CART**	**GL**
AUC	0.67 [0.65;0.69]	0.75 [0.74;0.77]	0.68 [0.67;0.70]	0.66 [0.64;0.67]

(e)

Figure 6: Experiment 2, groups of moderate and equal sizes. From top left to bottom right: (a) Boxplots of the AUC, (b) Boxplots of the tree depth, (c) Groups selection frequencies for TLDA, TPLDA and GL, (d) Groups selection frequencies for CART, (e) AUC distribution according to the methods.



(a)



(b)

Figure 7: Experiments 2. Misclassification error estimate on the training set (a) and on the validation set (b) versus the tree depth for TLDA, TPLDA and CART. The dotted lines indicates the value of the Bayes error. The Bayes error is 10%.

Moreover, TPLDA gives the best predictive performances (Figure 6a and Table 6e). Thus, TPLDA outperforms all the other algorithms by elaborating more accurate and more easily understandable classification rules.
The lower performances of TLDA and GL can be explained by the fact that the algorithms do not identify enough relevant groups (Figure 6c). As previously mentioned, TLDA is sensitive to the dimension of the training sample and performs badly when the number of observations is small compared to the number of inputs. Consequently, splits of the deeper nodes (i.e. splits in

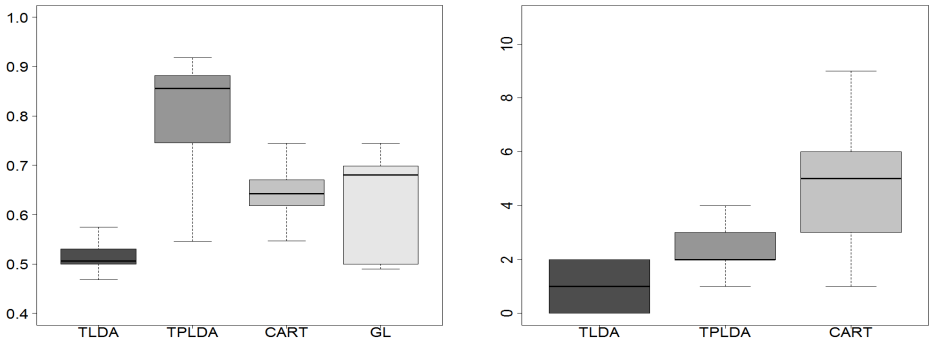
small nodes) can be less informative and so are deleted during the pruning step, leading to smaller final TLDA trees than the final TPLDA trees.

Experiment 3: large groups
This simulation deals with the high-dimensional problem introduced previously (see Example 2.2 and Subsection 2.2). It follows the general simulation design presented above by setting:

- $n = m = 100$,
- $p_j = 50$ for $j = 1, \dots 10$.

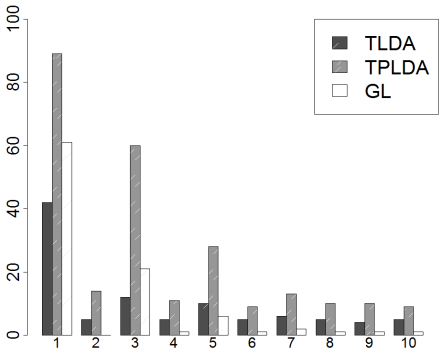
The correlation parameters remain unchanged: $(c_w, c_b) = (0.85, 0.80)$.

Results. Figure 8 gives the group selection frequencies, the tree-depth distribution and the AUC distribution over the 500 iterations for each algorithm. As in the previous scenarios, all algorithms identify more frequently the most relevant groups than the noisy groups and the selection frequency of a given group is an increasing function of the discriminatory power the group (Figure 8c and Figure 8d).
This scenario highlights the bad performances of LDA in high-dimensional situations. Indeed, in the first split, the LDA used to split the entire data space overfits the training set. This can be seen in Figures 9a and 9b: the training misclassification error decreases much faster for TLDA and becomes smaller than the Bayes error from the first split while the test misclassification error for TLDA remains stable. Consequently, after applying the pruning procedure which removes the less informative nodes, the final TLDA tree is trivial in 28.40 % of the simulations (Figure 8b and Table 4).
PLDA overcomes the weakness of LDA in high-dimensional situations. Indeed, TPLDA is not affected by the high-dimension and the algorithm still outperforms all the other assessed algorithms (Table 8e). The misclassification error in both the training and the validation test decreases more faster than CART (Figure 9). Consequently, TPLDA builds smaller trees than CART (Figure 8b). As previously, all the algorithms well identify the most true relevant groups and even though the group selection frequencies of CART and TPLDA have not the same meanings, there are very similar (Figure 8c and Figure 8d).

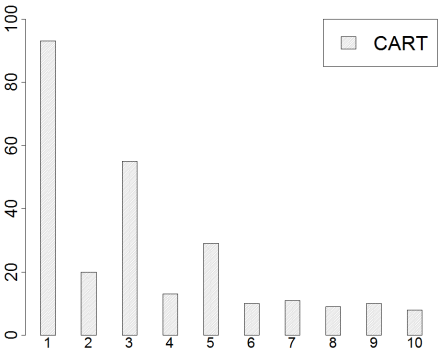


(a)

(b)



(c)

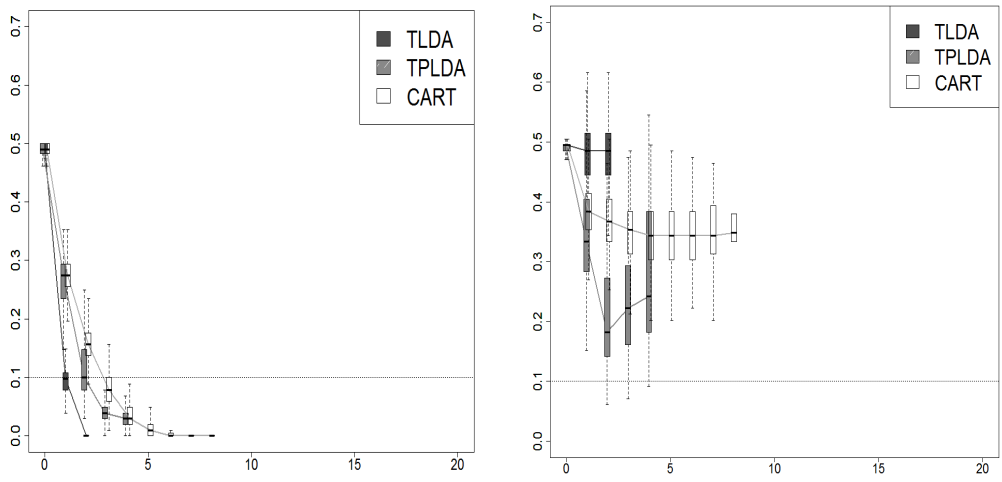


(d)

	TLDA	**TPLDA**	**CART**	**GL**
AUC	0.51 [0.50;0.53]	0.86 [0.75;0.88]	0.64 [0.62;0.67]	0.68 [0.50;0.69]

(e)

Figure 8: Experiment 3, large groups. From top left to bottom right: (a) Boxplots of the AUC, (b) Boxplots of the tree depth, (c) Groups selection frequencies for TLDA, TPLDA and GL, (d) Groups selection frequencies for CART, (e) AUC distribution according to the methods.



(a) (b)

Figure 9: Experiments 3. Misclassification error estimate on the training set (a) and on the validation set (b) versus the tree depth for TLDA, TPLDA and CART. The dotted lines indicates the value of the Bayes error. The Bayes error is 10%.

Note that compared to the previous scenarios, TPLDA and GL better performs (Figure 8a and Table 8e) because the discriminatory power of every relevant group is higher compared to the previous scenarios. Indeed, given that, in every group, all inputs share the same discriminatory power, the discriminatory power of a predictive group increases when the group size increases. Yet, it can be noticed that in this scenario, the variability of the predictive performances is higher for these two algorithms.

Table 4: Experiment 3, large groups – TLDA tree depth.

Tree depth	0	1	2
Frequency (in %)	28.40	40.40	31.20

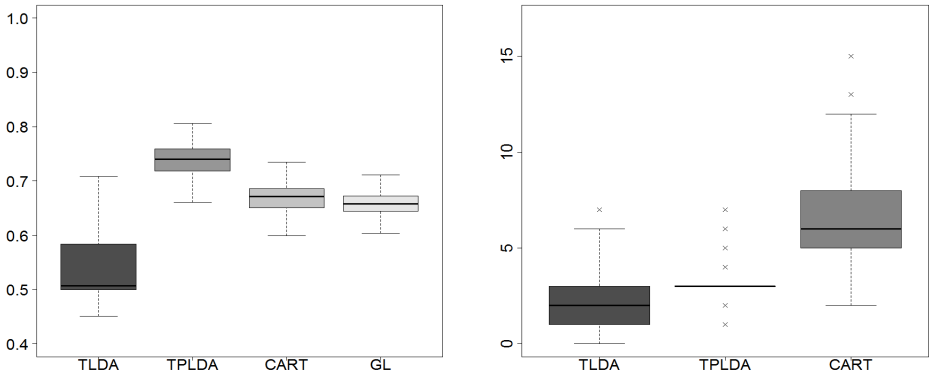
Experiment 4: large noisy groups

The aim of this scenario is to compare the algorithms in a difficult situation involving large noisy groups. The size of the noisy group is set to 50 and the size of the relevant group is set to 10:

$$p_j = \begin{cases} 10 & \text{if } j \equiv 1 \pmod 2; \\ 50 & \text{otherwise,} \end{cases}$$

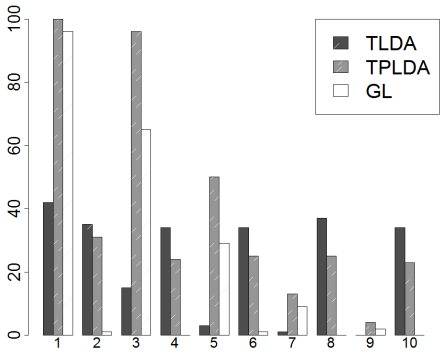
for $j = 1, \ldots, 10$. The correlation between inputs is the same that in the previous scenario (i.e. $(c_w, c_b) = (0.85, 0.8)$) and the training samples and the validation samples include $n = m = 500$ simulated observations.

Results. As in the previous scenario, TLDA is sensitive to the sizes of the groups. Indeed, TLDA selects as many Group 1 (i.e. the most relevant group) as the five large noisy groups (Figure 10c). Besides, the first splitting group chosen by TLDA is a large noisy group in 51.2% of the simulations (Figure 11 and Table 11). This explains why the misclassification error in the training set decreases fast whereas the misclassification error in the validation set stays quite stable (Figure 12). Thus, in this situation, TLDA selects the trivial tree in 22.60% of the simulations and has bad predictive performances (Figure 10a, Figure 10b and Table 10e). On the contrary, the other algorithms and especially TPLDA are not affected by the large noisy groups. Indeed, the other algorithms select mainly the most predictive groups/inputs (Figure 10c and figure 10d).
Moreover, as in the previous scenarios, the misclassification error in both the training set and the validation set decreases faster for TPLDA than for CART (Figure 12). Consequently, TPLDA builds smaller trees while having better predictive performance than the other algorithms (Figure 10a, Figure 10b and Table 10e). Thus, whatever the group sizes, TPLDA performs well in these two last difficult scenarios.

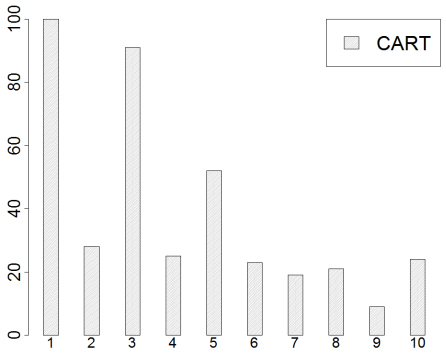


(a)

(b)



(c)

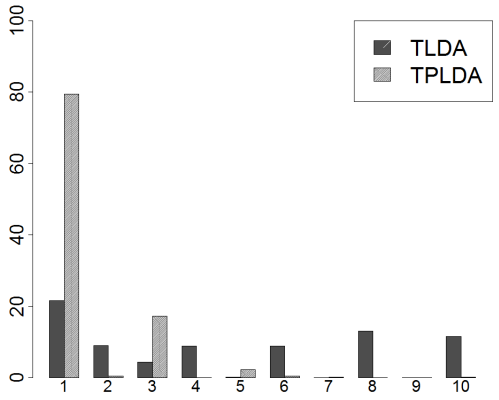


(d)

	TLDA	**TPLDA**	**CART**	**GL**
AUC	0.51	0.74	0.67	0.66
	[0.50;0.58]	[0.72;0.76]	[0.65;0.68]	[0.64;0.67]

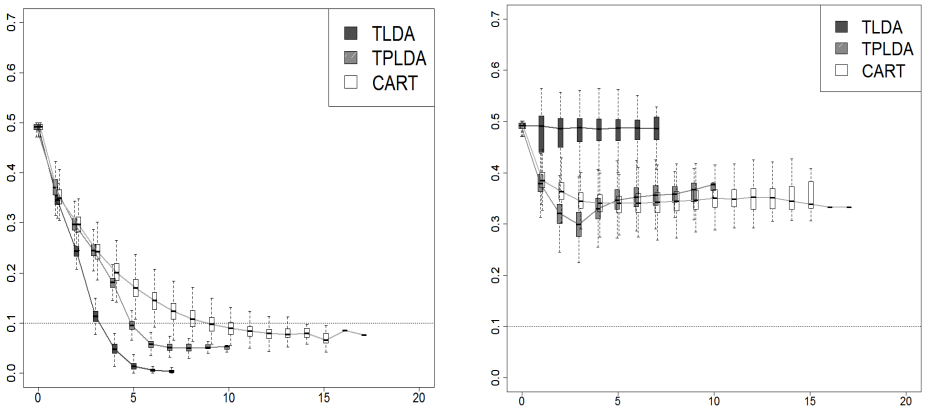
(e)

Figure 10: Experiment 4, large noisy groups. From top left to bottom right: (a) Boxplots of the AUC, (b) Boxplots of the tree depth, (c) Groups selection frequencies for group methods, (d) Groups selection frequencies for CART, (e) AUC distribution according to the methods.



	Group 1	Other relevant groups	Noisy Groups	Total
TLDA	21.60	4.60	52.10	77.40
TPLDA	79.40	19.60	1.00	100

Figure 11: Experiment 4. First splitting group (in %) for the TLDA maximal trees and the TPLDA maximal trees.



(a) (b)

Figure 12: Experiments 4. Misclassification error estimate on the training set (a) and on the validation set (b) versus the tree depth for TLDA, TPLDA and CART. The dotted lines indicates the value of the Bayes error. The Bayes error is 10%.

Remark 3.2. CART has also been pruned with our pruning strategy to assess the sensitivity to the pruning method (see Appendix B). The obtained pruned trees are slightly larger but have similar prediction accuracy than the pruned trees selected with the cost-complexity pruning. Thus, the prediction accuracy does not seem to be affected by the choice of the pruning method.

4. Application to gene expression data

In this section, TLDA and TPLDA are compared to CART and GL on the leukemia microarray study of Golub et al. (1999). This study is the first study on leukemia tumor classification using gene expression data and has then become a benchmark in cancer tumors classification. The data can be freely downloaded from `http://www.broadinstitute.org/cgi-bin/cancer/publications/pub_paper.cgi?paper_id=43`. The data set consists of 72 tumor samples from leukemia patients, with each sample giving the expression levels of 2185 genes. Based on pathological and histological criteria, 47 tumor samples are classified as acute lymphoblastic leukemias (called ALL) and the remaining 25 samples are classified as acute myeloid leukemias (called AML). The algorithms are applied on this data set to elaborate a classification rule based on the gene expression patterns. This classification rule will be mainly used both to predict the leukemia tumors and to identify groups or "marker" genes that allow to characterize the several tumors subtypes. So, the aim is to obtain a classification rule which both well classifies the leukemia tumors while being easily understandable to yield information about relationship between some groups of genes and the several tumors classes.

Data preprocessing and genes clustering
Following Dudoit et al. (2002) and Sewak et al. (2009), a preprocessing is applied to the whole data set. First, only the first quartile of genes with the greatest variation across the sample is considered, the other genes are excluded. Next, gene expression values less than 20 and higher than 16000 are set to 20 and 16000 respectively.
After that, as proposed by Lee & Batzoglou (2003), independent component analysis (ICA) is applied to cluster the remaining genes. Genes are then clustered into non-mutually exclusive groups based on their load on each ICA component.

Evaluation of the methods

Following previous analysis of this data set (Sewak et al., 2009; Dudoit et al., 2002), the accuracy rate is used to assess the predictive performances of TLDA, TPLDA, CART and GL. It has been computed by using 5-fold cross-validation and 200 Monte Carlo replications of the whole data set.

Results

The data preprocessing keeps 545 genes. Next, based on the ICA applied on the remaining genes, ten groups are created. The choice of the number of groups has been driven by Tamayo et al. (2007) and Sewak et al. (2009). Each group includes the 5% of genes with the highest loads in absolute terms. Then, each group consists in 28 genes and the ten groups are based on 82 genes of which 32 genes belonging only a single group.

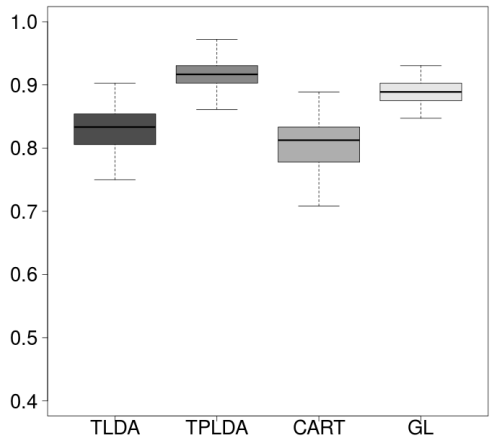


Figure 13: Application on the Leukemia data set: boxplots of the accuracy.

The distribution of the accuracy rate according to the algorithm is represented in Figure 13. On this data set, TPLDA outperforms the other methods with a median accuracy rate of 92%. GL has lightly lower predictive performances with a median accuracy rate of 89% while TLDA and CART are less efficient with a median accuracy rate of 83% and 81% respectively. Moreover, the accuracy rate seems more stable with TPLDA and GL.
As mentioned above, one of the major issue in the classification of tumors using gene expression data is the identification of relevant variables or relevant

groups of genes. Figure 14 displays the TPLDA tree obtained by applying the algorithm on the whole data set. The tree consists in two splits. The first one is defined according to the first group which includes the variables with the highest load (in absolute terms) on the first ICA component. The second split is based on the fourth group including the variables with the highest (in absolute terms) on the fourth ICA component. We are not attempted to give biological interpretation of the genes and groups of genes involved in the TPLDA tree. However, these results seem consistent with those presented in the original paper of Golub et al. (1999). Indeed, in the two splits, some of the genes that contribute the most to the split decision rule belongs the set of informative genes reported by Golub et al. (1999). Moreover, in the TPLDA tree, the shape of the splits and the tree structure enable to highlight possible relevant interactions between genes inside the first and the fourth groups and also between these groups.

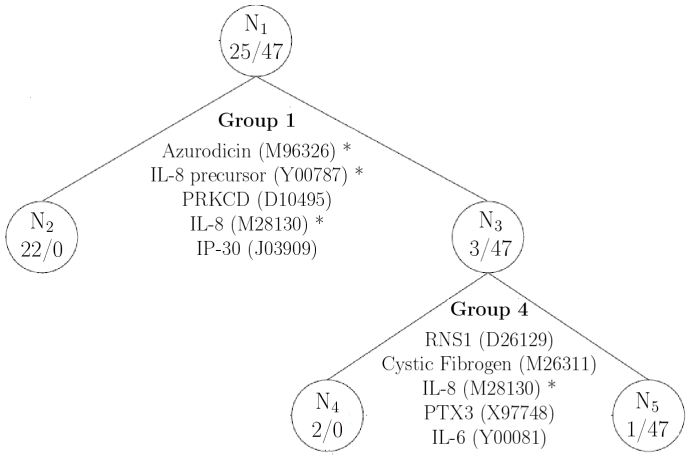


Figure 14: Application on the Leukemia data set: the TPLDA tree estimated on the whole leukemia data set. From the tree, circles refer to the nodes. The two numbers in the circles indicate the number of AML and the number of ALL belonging the node. The splitting group is indicated below the node with the five genes that contribute the most to the splitting rule. The stars are used to identify the genes belonging the set of relevant genes reported by Golub et al. (1999).

On this data set, TPLDA gives competitive results with other popular classifiers such as nearest-neighbors classifiers (Dudoit et al., 2002), committe

neural networks (CNN) (Sewak et al., 2009) or SVM (Guyon et al., 2002). Compared to Dudoit et al. (2002), our study considered twice as many genes (82 genes) and the genes are structured in groups based on ICA which enables to underline the existence of informative biological processes and genes interactions (Lee & Batzoglou, 2003). By comparison, Dudoit et al. (2002) keep only the 40 genes the most correlated with the tumor subtypes and then genes selection ignores possible genes interactions or biological processes. Sewak et al. (2009) and (Guyon et al., 2002) have applied CNN and SVM respectively which lead to high predictive performances. Yet, the resulted classification rules cannot be easily interpreted. Therefore, the TPLDA algorithm achieves comparatively a better trade-off between interpretation and prediction accuracy than CNN or SVM.

5. Conclusion

In this work we have presented a new way to classify data with grouped inputs. Our approach consists in building a classification tree based on splits of the input space defined according to the groups. Two algorithms have been introduced: TLDA which uses LDA to defined the decision splits and TPLDA which partitions the inputs space by performing regularized LDA. To our knowledge, there are the first classification trees algorithms dealing with grouped inputs.

The TLDA and TPLDA can be considered as two multivariate classification tree algorithms which use linear combinations of inputs to built the partition like lots of multivariate classification tree algorithms. However, contrary to most of the multivariate classification tree algorithms (Breiman et al. 1984; Murthy et al. 1993; Loh & Shih 1997; Li et al. 2003; Wickramarachchi et al. 2016, etc.), TLDA and TPLDA are not computationally expensive and they built classification trees easily understandable. Based on the simulation study and the application on real data, it is clear that TPLDA is well adapted to classify data with groups of inputs. Furthermore, as shown in the application on gene expression data, TPLDA is also an effective method to perform variable selection when inputs are grouped. Thus, this algorithm shows promising results in term of predictive performances, interpretation and variable selection.

Appendix A. Time complexity of the tree-groups algorithms at a node

In the following section, the maximal time complexity at a node N of TLDA and TPLDA is detailed. We assume that there are n_N observations in the node N, J groups of variables and the group j, $j = 1, \ldots, J$, includes p_j inputs.

Appendix A.1. Time complexity of TLDA

To split a node N including n_N observations, first the algorithm estimates a split for each group:

- Complexity for estimating the hyperplan split with the group j by using a LDA is $\mathcal{O}(p_j n_N t)$ with $t = min(p_j, n_N)$. See Cai et al. (2008) for a detailed calculation of LDA time complexity.
- Complexity for assigning each of the n_N observations to one of the two child nodes $N_0(j)$ and $N_1(j)$ is $\mathcal{O}(n_N p_j)$.
- Complexity for computing the decrease in impurity resulting from the split $\Delta_j(N)$ is $\mathcal{O}(n_N)$.

So, the maximal time complexity at a node to estimate the split with the group j is $\mathcal{O}(p_j n_N t) + \mathcal{O}(n_N p_j) + \mathcal{O}(n_N) = \mathcal{O}(p_j n_N t)$. The previous steps are repeated on the J groups, so the complexity for estimating a split for each group is $\mathcal{O}(J p_{\max} n_N t)$ with $p_{\max} = \max_j(p_j)$.

Next, the algorithm selects, among the J estimated splits, the one which maximizes the impurity decrease. The complexity for choosing the splitting group is then $\mathcal{O}(1)$.

Consequently, the maximal time complexity of TLDA at a node is in the worst case $\mathcal{O}(J n_N p_{\max} t_{\max}) + \mathcal{O}(1) = \mathcal{O}(J n_N p_{\max} t_{\min})$ with $p_{\max} = \max_j(p_j)$ and $t_{\min} = \min(p_{\max}, n_N)$.

Appendix A.2. Time complexity of TPLDA

To split a node N including n_N observations, first the algorithm estimates a split for each group. Compared to TLDA, the estimation of the split for a given group j consists in the two following steps: first the selection of the shrinkage parameter λ_j and next the application of a PLDA using the selected value for λ_j and the group j.

Complexity of performing PLDA:
PLDA computation steps are described in the original paper (Witten & Tibshirani, 2011). We detailed here its maximal time complexity:

- Complexity for constructing the estimated between covariance matrix $\widehat{B}_N^j$ is $\mathcal{O}\left(n_N p_j^2\right)$ with p_j, $j = 1, \ldots, J$ being the size of $\mathbf{X}^j$.
- Complexity for constructing the estimated diagonal within covariance matrix $\widetilde{\Sigma}_N^j$ is $\mathcal{O}\left(n_N p_j\right)$.
- Complexity of the eigen analysis of $(\widetilde{\Sigma}_N^j)^{-1}\widehat{B}_N^j$ is $\mathcal{O}\left(p_j^2\right)$.
- Complexity of the eigen analysis of $(\widehat{B}_N^j)^{-1}\widehat{B}_N^j$ and the research for the dominant eigenvector is $\mathcal{O}\left(p_j^2\right)$.
- Complexity for estimating the penalized discriminant vector $\hat{\beta}_{pen}^j$ by performing M iterations of the minimization-maximization algorithm (Lange et al., 2000) is $\mathcal{O}\left(Mp_j^2\right)$. Note that the minimization-maximization algorithm uses the dominant eigenvector as the initial value for the penalized discriminant vector.

So, maximal time complexity of performing PLDA on the group j in the node N is $\mathcal{O}\left(n_N p_j^2\right) + \mathcal{O}\left(n_N p_j\right) + \mathcal{O}\left(p_j^2\right) + \mathcal{O}\left(Mp_j^2\right) = \left(n_N p_j^2\right)$ by supposing that $M < n_N$.

Complexity of the selection of the shrinkage parameter:
The value of shrinkage parameter λ_j is determined by using a K-fold cross-validation and a grid $\{v_1, \ldots, v_L\}$ containing L values for λ_j.

First, before performing the K-fold cross-validation, the n_N observations in the node N into K disjoint samples $\{S_1, \ldots, S_K\}$. The complexity of this step is $\mathcal{O}\left(n_N\right)$.

Next, for each fold k, $k = 1, \ldots, K$ and each value v_l, $l = 1, \ldots, L$:

- Performing a PLDA on $N \setminus S_k$ (i.e. all the disjoint sets $\{S_1, \ldots, S_K\}$ excepted S_k) with $\lambda_j = v_l$: the complexity is $\mathcal{O}\left(\frac{K-1}{K} n_N p_j^2\right)$.
- Predicting the class of each observation of S_k using the resulted PLDA model computed on $N \setminus S_k$: the complexity is $\mathcal{O}\left(\frac{n_N}{K} p_j\right)$.

Then, the two previous steps are repeated on each fold k, $k = 1, \ldots, K$ and each value v_l, $l = 1, \ldots, L$. So the complexity of the cross-validation is $\mathcal{O}\left(L(K-1)n_N p_j^2\right) + \mathcal{O}\left(Ln_N p_j\right)$.

After that, the impurity decrease $\Delta_j(N, v_l)$ is computed for each value v_l,$l = 1, \ldots, L$: its time complexity is $\mathcal{O}\left(n_N\right)$ for one value v_l, $l = 1, \ldots, L$ and is then $\mathcal{O}\left(Ln_N\right)$ for all the values v_l.

Next, the algorithm selects the λ_j value in the grid $\{v_1, \ldots, v_L\}$ which maximizes the decrease in impurity: the complexity is $\mathcal{O}\left(1\right)$.

Therefore, the complexity for selecting the value of λ_j is $\mathcal{O}\left(n_N\right)+\mathcal{O}\left(L(K-1)n_N p_j^2\right)+ \mathcal{O}\left(Ln_N p_j\right) + \mathcal{O}\left(Ln_N\right) + \mathcal{O}\left(1\right) = \mathcal{O}\left(L(K-1)n_N p_j^2\right)$.

The two previous steps (selection of the shrinkage parameter λ_j and PLDA computation) are repeated on all the J groups. So, the complexity for estimating a split for each group is $(Jn_N p_{\max}^2) + \mathcal{O}\left(JL(K-1)n_N p_{\max}^2\right) = \mathcal{O}\left(JLKn_N p_{\max}^2\right)$ with $p_{\max} = \max_j(p_j)$, L denoting the number of possible value for λ_j, $j = 1, \ldots, J$ and K denoting the number of folds used in the cross-validation.

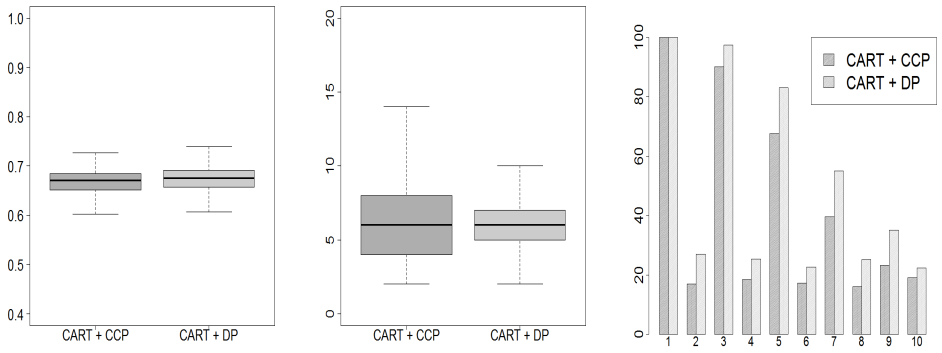
Next, the algorithm selects, among the J estimated splits, the one which maximizes the impurity decrease: the complexity of this step is $\mathcal{O}\left(1\right)$.

Consequently, the maximal time complexity of TPLDA at a node N is in the worst case $(Jn_N p_{\max}^2) + \mathcal{O}\left(JL(K-1)n_N p_{\max}^2\right) + \mathcal{O}\left(1\right) = \mathcal{O}\left(JLKn_N p_{\max}^2\right)$ with $p_{\max} = \max_j(p_j)$.

Appendix B. Additional figures from the simulation study

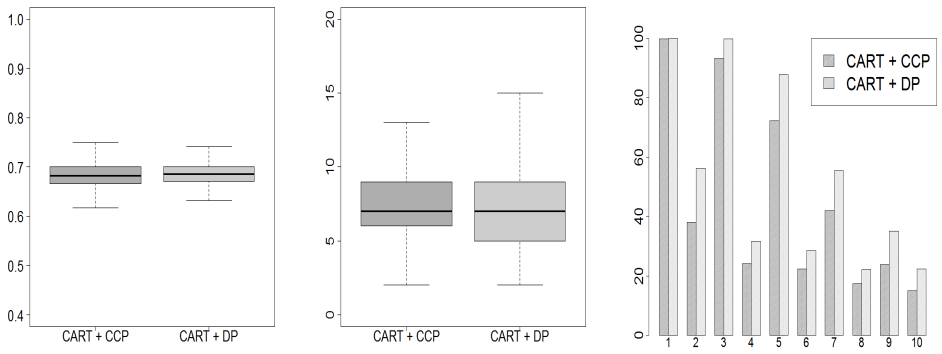
In the numerical experiment, the proposed pruning strategy based on the tree depth is also applied on the maximal CART trees, for each scenario. Figures B.15, B.16, B.17 and B.18 display the AUC distribution, the tree depth and the group selection frequencies for the final tree CART according to the pruning strategy. $\mathsf{CART} + \mathsf{CCP}$ refers to CART when applying the cost-complexity pruning strategy while $\mathsf{CART} + \mathsf{DP}$ refers to CART when applying the proposed pruning method based on the depth. Overall, the two pruning methods lead to similar CART trees and so similar classification

rules. Indeed, the predictive performances and the tree depth are very close. Moreover, the group selection frequencies do not really differ: they are lightly higher when using the proposed pruning strategy based on the depth which can be expected.



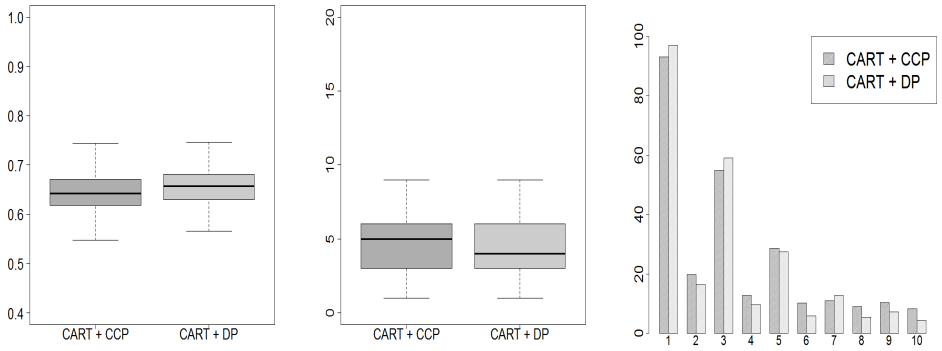
(a) (b) (c)

Figure B.15: Experiment 1, ungrouped inputs: comparison of the pruning methods. From left to right: (a) Boxplots of the AUC, (b) Boxplots of the tree depth, (c) Groups selection frequencies according to the method.



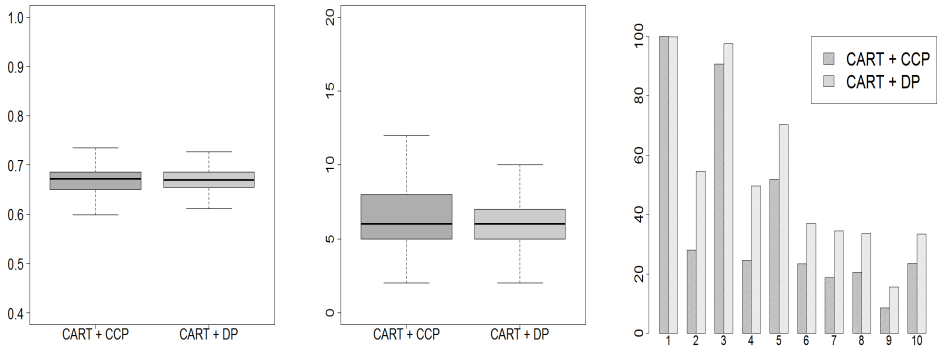
(a) (b) (c)

Figure B.16: Experiment 2, groups of moderate and equal size: comparison of the pruning methods. From left to right: (a) Boxplots of the AUC, (b) Boxplots of the tree depth, (c) Groups selection frequencies according to the method.



(a) (b) (c)

Figure B.17: Experiment 3, large groups: comparison of the pruning methods. From left to right: (a) Boxplots of the AUC, (b) Boxplots of the tree depth, (c) Groups selection frequencies according to the method.



(a) (b) (c)

Figure B.18: Experiment 4, large noisy groups: comparison of the pruning methods. From left to right: (a) Boxplots of the AUC, (b) Boxplots of the tree depth, (c) Groups selection frequencies according to the method.

References

Bickel, P. J., & Levina, E. (2004). Some theory for Fisher's linear discriminant function,'naive bayes', and some alternatives when there are many more variables than observations. *Bernoulli*, *10*, 989–1010.

Bouveyron, C., Girard, S., & Schmid, C. (2007). High-dimensional discriminant analysis. *Communications in Statistics Theory and Methods*, *36*, 2607–2623.

Breiman, L., Friedman, J., Stone, C. J., & Olshen, R. A. (1984). *Classification and regression trees*. CRC Press.

Cai, D., He, X., & Han, J. (2008). Training linear discriminant analysis in linear time. In *2008 IEEE 24th International Conference on Data Engineering* (pp. 209–217).

Devroye, L., Györfi, L., & Lugosi, G. (2013). *A probabilistic theory of pattern recognition* volume 31. Springer Science & Business Media.

Dudoit, S., Fridlyand, J., & Speed, T. P. (2002). Comparison of discrimination methods for the classification of tumors using gene expression data, . *97*, 77–87.

Friedman, J., Hastie, T., & Tibshirani, R. (2001). *The elements of statistical learning* volume 1. Springer Series in Statistics New York.

Friedman, J. H. (1989). Regularized discriminant analysis. *Journal of the American Atatistical Association*, *84*, 165–175.

Golub, T. R., Slonim, D. K., Tamayo, P., Huard, C., Gaasenbeek, M., Mesirov, J. P., Coller, H., Loh, M. L., Downing, J. R., Caligiuri, M. A. et al. (1999). Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. *Science*, *286*, 531–537.

Gregorutti, B., Michel, B., & Saint-Pierre, P. (2015). Grouped variable importance with random forests and application to multiple functional data analysis. *Computational Statistics & Data Analysis*, *90*, 15–35.

Guyon, I., Weston, J., Barnhill, S., & Vapnik, V. (2002). Gene selection for cancer classification using support vector machines. *Machine Learning*, *46*, 389–422.

Hunt, E. B., Martin, J., & Stone, P. J. (1966). *Ec*. Academic Press.

Hunter, D. R., & Lange, K. (2004). A tutorial on MM algorithms. *The American Statistician*, *58*, 30–37.

Kass, G. V. (1980). An exploratory technique for investigating large quantities of categorical data. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, *29*, 119–127.

Lange, K., Hunter, D. R., & Yang, I. (2000). Optimization transfer using surrogate objective functions. *Journal of Computational and Graphical statistics*, *9*, 1–20.

Lee, S.-I., & Batzoglou, S. (2003). Application of independent component analysis to microarrays. *Genome Biology*, *4*, R76.

Li, X.-B., Sweigart, J. R., Teng, J. T., Donohue, J. M., Thombs, L. A., & Wang, S. M. (2003). Multivariate decision trees using linear discriminants and tabu search. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, *33*, 194–205.

Lim, T.-S., Loh, W.-Y., & Shih, Y.-S. (2000). A comparison of prediction accuracy, complexity, and training time of thirty-three old and new classification algorithms. *Machine Learning*, *40*, 203–228.

Loh, W.-Y., & Shih, Y.-S. (1997). Split selection methods for classification trees. *Statistica Sinica*, *7*, 815–840.

Meier, L., Van De Geer, S., & Bhlmann, P. (2008). The group lasso for logistic regression. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, *70*, 53–71.

Murthy, S. K., Kasif, S., Salzberg, S., & Beigel, R. (1993). OC1: A randomized algorithm for building oblique decision trees. In *Proceedings of AAAI* (pp. 322–327). AAAI volume 93.

Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning*, *1*, 81–106.

Quinlan, J. R. (1993). *C4.5: programs for machine learning*. Morgan Kaufmann Publishers Inc.

Sewak, M. S., Reddy, N. P., & Duan, Z.-H. (2009). Gene expression based leukemia sub-classification using committee neural networks. *Bioinformatics and Biology Insights*, *3*, 89.

Shao, J., Wang, Y., Deng, X., Wang, S. et al. (2011). Sparse linear discriminant analysis by thresholding for high dimensional data. *The Annals of Statistics*, *39*, 1241–1265.

Tamayo, P., Scanfeld, D., Ebert, B. L., Gillette, M. A., Roberts, C. W., & Mesirov, J. P. (2007). Metagene projection for cross-platform, cross-species characterization of global transcriptional states. *Proceedings of the National Academy of Sciences*, *104*, 5959–5964.

Tardivel, P. J. C., Canlet, C., Lefort, G., Tremblay-Franco, M., Debrauwer, L., Concordet, D., & Servien, R. (2017). Asics: an automatic method for identification and quantification of metabolites in complex 1d 1h nmr spectra. *Metabolomics*, *13*, 109.

Wei-Yin Loh, N. V. (1988). Tree-structured classification via generalized discriminant analysis. *Journal of the American Statistical Association*, *83*, 715–725.

Wickramarachchi, D., Robertson, B., Reale, M., Price, C., & Brown, J. (2016). HHCART: an oblique decision tree. *Computational Statistics & Data Analysis*, *96*, 12–23.

Witten, D. (2015). *Penalized Classification using Fisher's Linear Discriminant*. URL: `http://CRAN.R-project.org/package=penalizedLDA` R package version 1.1.

Witten, D. M., & Tibshirani, R. (2011). Penalized classification using Fisher's linear discriminant. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, *73*, 753–772.

Xu, P., Brock, G. N., & Parrish, R. S. (2009). Modified linear discriminant analysis approaches for classification of high-dimensional microarray data. *Computational Statistics & Data Analysis*, *53*, 1674–1687.