Arbres de décision et forêts aléatoires pour variables groupées

Audrey Poterie

UNIVERSITE BRETAGNE LOIRE / MATHSTIC

INSA RENNES

THESE DE DOCTORAT DE

L'INSA RENNES

COMUE UNIVERSITE BRETAGNE LOIRE

ECOLE DOCTORALE N° 601
Mathématiques et Sciences et Technologies de l'Information et de la Communication
Spécialité : *Mathématiques et leurs Interactions*

Par

Audrey POTERIE

Arbres de décision et forêts aléatoires pour variables groupées

Thèse présentée et soutenue à l'Amphi Bonnin de l'INSA Rennes, le 18 octobre 2018
Unité de recherche : IRMAR
Thèse N° : 18ISAR 19 / D18-19

Rapporteurs avant soutenance :

Christophe BIERNACKI	PR, Université de Lille 1
Jean-Michel POGGI	PR, Université Paris-Sud

Composition du Jury :

Président :	
Gérard BIAU	PR, Sorbonne Université
Christophe BIERNACKI	PR, Université de Lille 1
Christine TULEAU	MCF, Université de Nice
Directeur de thèse	
Jean-François DUPUY	PR, INSA Rennes
Co-directeur de thèse	
Valérie MONBET	PR, Université Rennes 1
Co-encadrant de thèse	
Laurent ROUVIERE	MCF, Université Rennes 2

Intitulé de la thèse :

Arbres de décision et forêts aléatoires pour variables groupées

Audrey POTERIE

En partenariat avec :

Remerciements

Je tiens en premier lieu à remercier Valérie Monbet, Laurent Rouvière et Jean-François Dupuy, mes directeurs de thèse. Je vous remercie sincèrement pour la confiance que vous m'avez accordez durant ces trois années. Vous m'avez orientée tout en me laissant libre de mes choix. Vous m'avez beaucoup appris et j'espère être un jour aussi expérimentée et clairvoyante que vous. Merci d'être toujours encourageant et rassurant. Ce fut un réel plaisir que de travailler avec vous, et j'espère que cela pourra continuer.

Je voudrais adresser mes remerciement chaleureux à Nicolas Klutchnikoff et à nouveau à Laurent Rouvière pour m'avoir donné l'opportunité de travailler sur le projet clustering. Ce projet m'enthousiasme. Je suis ravie de travailler avec vous.

Merci à Christophe Biernacki et Jean-Michel Poggi qui ont accepté d'être les rapporteurs de cette thèse. Je suis également très honorée de la présence de Christine Tuleau et Gérard Biau dans mon jury. Je vous en remercie.

Un grand merci à Pierre Navaro pour sa gentillesse et ses précieux conseils en programmation.

J'adresse également mes remerciements à l'ensemble des membres du département Génie Mathématiques de l'INSA Rennes avec un merci tout particulier à Martine et Patricia, sans qui le département ne tournerait pas aussi bien, ainsi qu'à Pierrette, Mounir et Olivier pour leur écoute et leurs précieux conseils. Je remercie aussi plus largement l'équipe de statistique de l'IRMAR.

Je ne peux écrire ces remerciements sans évoquer Emilie et Tangi, mes deux acolytes de thèse et amis. J'ai passé avec vous de très bons moments (mariages, voyages, appartement de l'INSA, soirées, réveillons et j'en passe...). J'attends avec impatience nos retrouvailles (le lieu reste cependant à définir : Paris ? Guelph ? Melbourne ?) et espère encore voyager aux quatre coins de monde avec vous. Je pense aussi aux jolies amitiés que j'ai tissées au cours de ces trois années : Tuan (que je remercie aussi pour son accueil chaleureux à Danang), Florent, les 2 Marie, Margot. Merci à Valérie qui a en quelques sortes jouée le rôle de grande sœur à la fin de la thèse. Merci à Khang, Tuyen, Trinh pour leurs conseils avisés lors de l'organisation de notre périple vietnamien. Et enfin merci à ceux qui ont pris la relève dans notre grand bureau: Othman, Bilel. J'adresse aussi une pensée à mes amis, Odile (pour nos

conversations où l'on refait le monde), Sarah (pour nos escapades marocaines), aux copains de l'athlétisme, aux copains rennais et parisiens, ainsi qu'à mes deux miss « discussions et débriefs du dimanche soir ».

Pour finir, je remercie de tout cœur ma famille, mes parents qui me soutiennent toujours sur tout et ma petite sœur (qui n'est plus petite depuis longtemps et dont je suis très fière). Je vous aime infiniment.

Resumé

L'apprentissage supervisé consiste à expliquer et/ou prédire une sortie y par des entrées x. Dans de nombreux problèmes, les entrées x ont une structure de groupes connue et/ou clairement identifiable. Le regroupement des variables peut être naturel ou bien défini dans le but de modéliser les relations entre les différentes variables. Par exemple, en biologie, lorsque l'on souhaite étudier la composition chimique d'un sérum à l'aide de la spectrométrie de masse, les variables explicatives, de nature fonctionnelle, peuvent être divisées en groupes représentants différentes parties de la courbe. Dans ce contexte, l'élaboration d'une règle de prédiction prenant en compte cette structure de groupes peut se révéler plus pertinente qu'un algorithme effectué sur les variables individuelles tant au niveau des performances prédictives que de l'interprétation. Des algorithmes supervisés construits sur des groupes de variables ont déjà été proposés. Un des plus connus est certainement le *Group lasso*.
L'objectif du présent travail de thèse est de développer des méthodes par arbres adaptées aux variables groupées. En effet, les arbres sont des algorithmes très utilisés en statistique. Ils permettent de construire des algorithmes simples et interprétables. De plus, de nombreux algorithmes d'agrégation tels que le boosting et les forêts alétoires sont définis à partir d'arbres. Ces algorithmes figurent souvent parmi les plus performants dans des problèmes concrets d'apprentissage supervisé. Nous proposons deux approches qui utilisent la structure groupée des variables pour construire des arbres de décision. Ces deux méthodes élaborent tout d'abord un arbre maximal au moyen d'un partitionnement récursif de l'espace des données. La première méthode permet de construire des arbres binaires en classification. Une coupure est définie par le choix conjoint d'un groupe de variables et d'une combinaison linéaire des variables du dit groupe. La seconde approche, qui peut être utilisée à la fois en régression et en classification, construit un arbre non-binaire dans lequel chaque coupure est un arbre binaire. Pour ces deux approches, l'arbre maximal est ensuite élagué. Nous proposons pour cela deux stratégies d'élagage dont une est une généralisation de la méthode *minimal cost-complexity pruning* aux arbres non binaires. Les arbres de décision étant connus pour être instables, nous introduisons également une méthode de forêts aléatoires pour variables groupées. Outre l'aspect prédiction, ces trois nouvelles méthodes peuvent aussi être utilisées dans un objectif de sélection de groupes de variables grâce à l'introduction pour chacune d'elles d'indices d'importance adaptés aux groupes de variables.
Ce travail de thèse est complété par une partie indépendante des autres dans laquelle nous nous plaçons dans un cadre d'apprentissage non supervisé. Nous introduisons un nouvel algorithme de clustering basé sur la classification hiérarchique *single linkage*. Sous certaines

hypothèses classiques concernant la séparabilité et la régularité des clusters, nous obtenons des vitesses de convergence pour le *risque de clustering* de l'algorithme proposé.

Mots-clefs : apprentissage statistique, groupes de variables, arbres de décision, forêts aléatoires, sélection de groupes de variables, clustering.

Abstract

Supervised learning consists in explaining and/or predicting an output y by using some inputs x. In many problems, inputs have a known and/or obvious group structure. These groups can naturally exist or they can be defined to capture the underlying input associations. For instance, in biology, when we want to study the chemical composition of a serum based on spectrometry data, the inputs, which are functional, can be clustered into groups representing the different parts of the curve. In this context, elaborating a prediction rule that takes into account the group structure can be more relevant than using an approach based only on the individual variables for both prediction accuracy and interpretation. Some supervised algorithms which build prediction rules based on groups of inputs have been already proposed. One of the best-known methods is certainly the *Group Lasso*.
The goal of this thesis is to develop some tree-based methods adapted to grouped variables. Indeed, tree-based approaches are commonly used in statistics. These methods allow to readily construct prediction rules easily understandable. Moreover, many aggregation algorithms such that the booting methods and the random forests are based on decision trees. These algorithms are often part of the list of the most successful methods currently use to handle prediction problems. Here, we propose two new tree-based approaches which use the group structure to build decision trees. These two methods begin with constructing a maximal tree by means of recursive partitioning of the data space. The first approach allows to build binary decision trees for classification problems. A split of a node is defined according to the choice of both a splitting group and a linear combination of the inputs belonging to the splitting group. The second method, which can be used for prediction problems in both regression and classification, builds a non-binary tree in which each split is a binary tree. In these two methods, the maximal tree is next pruned. To this end, we propose two pruning strategies, one of which is a generalization of the *minimal cost-complexity pruning* algorithm to non-binary trees. Since decisions trees are known to be unstable, we also introduce a method of random forests that deals with groups of inputs.
In addition to the prediction purpose, these three new methods can be also use to perform group variable selection thanks to the introduction for each of them of some measures of group importance.
This thesis work is supplemented by an independent part in which we consider the unsupervised framework. We introduce a new clustering algorithm which is based on the hierarchical clustering algorithm named *single linkage*. Under some classical regularity and sparsity assumptions, we obtain the rate of convergence of the *clustering risk* for the proposed

algorithm.

Keywords: supervised learning, groups of variables, decision trees, random forests, group variable selection, clustering.

Contents

1 Introduction **1**
- 1.1 Contexte et enjeux de la thèse . . . 1
- 1.2 Arbres de décision et forêts aléatoires . . . 4
- 1.3 Organisation du manuscrit et contributions . . . 13

2 Classification tree algorithm for grouped variables **24**
- 2.1 Introduction . . . 25
- 2.2 The Penalized Tree Group algorithm . . . 27
- 2.3 Evaluation of the method by simulation studies . . . 35
- 2.4 Application to tumor classification using gene expression data . . . 43
- 2.5 Conclusion . . . 47
- 2.6 Appendices . . . 48

3 Decision trees and random forests for grouped variables **58**
- 3.1 Introduction . . . 59
- 3.2 CARTGV: CART for grouped variables . . . 61
- 3.3 Random forests for grouped variables . . . 81
- 3.4 Numerical experiments . . . 86
- 3.5 Conclusion . . . 98
- 3.6 Appendices . . . 98

4 Statistical Analysis of a robust hierarchical clustering algorithm **117**
- 4.1 Introduction . . . 118
- 4.2 Mathematical framework . . . 119
- 4.3 Agglomerative clustering . . . 124
- 4.4 Simulation study . . . 129
- 4.5 Proofs . . . 137
- 4.6 A brief review of the Hausdorff measure . . . 143

5 Conclusion **145**
- 5.1 Bilan . . . 145
- 5.2 Perspectives . . . 146

Bibliography 149

Chapitre 1

Introduction

Table des Matières

1.1 Contexte et enjeux de la thèse 1

1.1.1 Présentation générale 1

1.1.2 Apprentissage supervisé 3

1.2 Arbres de décision et forêts aléatoires 4

1.2.1 Arbres CART 4

1.2.2 Forêts aléatoires 9

1.3 Organisation du manuscrit et contributions 13

1.3.1 Chapitre 2 : Arbres de classification pour variables groupées 13

1.3.2 Chapitre 3 : Arbres de décision et forêts aléatoires pour variables groupées 15

1.3.3 Chapitre 4 : Analyse statistique d'un algorithme de clustering hiérarchique en présence d'outliers 19

1.1 Contexte et enjeux de la thèse

1.1.1 Présentation générale

L'apprentissage statistique (Vapnik, 1995) désigne un vaste ensemble de méthodes permettant d'extraire l'information pertinente de données, dans un but principalement explicatif et/ou prédictif. De manière générale, on distingue deux types d'apprentissage : l'apprentissage supervisé et l'apprentissage non supervisé.

En apprentissage supervisé, la base de données consiste en un ensemble de copies indépendantes $\{(\mathbf{X}_i, Y_i)\}_{1\leq i\leq n}$ d'un vecteur aléatoire $(\mathbf{X}, Y)$, où $\mathbf{X}$ désigne le vecteur des variables

explicatives et Y est appelée la variable réponse. Dans ce contexte, l'objectif est de construire, à partir de la base de données, une règle de prédiction qui à toute nouvelle observation $\mathbf{X}_{n+1}$ associe la réponse Y_{n+1}.

Dans le cadre non-supervisé, les réponses $\{(Y_i)\}_{1\leq i\leq n}$ sont "cachées" ou non observées. La base de données consiste seulement en les observations $\{(\mathbf{X}_i)\}_{1\leq i\leq n}$ et l'objectif est de discriminer/trier en un certain sens les observations $\mathbf{X}_i$. En d'autres termes, il s'agit de regrouper les observations en classes (ou *clusters*) homogènes les plus différentes possibles. L'apprentissage non-supervisé est aussi appelé problème de classification (*clustering* en anglais).

De nos jours, l'apprentissage statistique joue un rôle important dans de nombreux domaines notamment en biologie et en santé (puces à ADN, essais cliniques, classification des maladies, etc.) ou encore en environnement (prévisions climatiques, risques environnementaux, etc.). De plus, avec l'essor des nouvelles technologies ainsi que le développement de moyens de stockage puissants, l'apprentissage statistique est aujourd'hui confronté à des données de plus en plus abondantes (*big data*) et diverses (images, textes, courbes, etc.). Dans ce contexte, il est essentiel d'exploiter la structure des données. En particulier, les données peuvent avoir une structure groupée.

On peut citer par exemple l'analyse du génome humain qui amène à traiter des bases de données comprenant l'expression de plusieurs milliers de gènes. Dans ces études, il est devenu fréquent de procéder à un regroupement au préalable des gènes en ensembles représentant par exemple différents processus biologiques (Tai & Pan, 2007; Tamayo et al., 2007). Également, en biologie, lorsque l'on souhaite étudier la composition chimique d'un sérum à l'aide de la spectrométrie, les variables explicatives, de nature fonctionnelle, peuvent êtres divisées en groupes représentant différentes parties de la courbe (Tardivel et al., 2017).

La problématique de la sélection de groupes de variables a été étudiée par de nombreux auteurs comme par exemple Zhang et al. (2008),Chakraborty & Pal (2008) ou encore Grimonprez (2016). Récemment, Gregorutti et al. (2015) ont proposé, dans le cadre de données fonctionnelles, un algorithme de sélection de groupes de variables utilisant les forêts aléatoires de Breiman.

D'autre part, l'élaboration d'une règle de prédiction sur les groupes plutôt que sur les variables individuelles peut être plus pertinent tant au niveau des performances prédictives que de l'interprétation. Ce problème a principalement été étudié dans le cadre paramétrique et semi-paramétrique, avec notamment les nombreuses approches régularisées utilisant la pénalité *Group lasso* introduite par Yuan & Lin (2006) (voir par exemple Huang et al. (2012) pour une synthèse des principales méthodes).

Dans le cadre non paramétrique et plus précisément dans le contexte des arbres de décisions

et des forêts aléatoires, il n'existe pas à notre connaissance de méthode permettant de construire des règles de prédiction à partir de groupes de variables.

L'objectif du présent travail de thèse est de développer des méthodes par arbres et de forêts aléatoires adaptées aux données ayant une structure de groupe. Ce travail est aussi complété par une partie indépendante où nous proposons un nouvel algorithme de clustering et nous étudions ses propriétés mathématiques.

Les arbres de décision et les forêts aléatoires sont aujourd'hui des méthodes de référence en apprentissage supervisé. Elles offrent de nombreux avantages (large applicabilité, une facilité d'utilisation, de bonnes performances, etc.) et sont aujourd'hui couramment utilisées dans de nombreux domaines notamment en biologie (Boulesteix et al., 2003; Geurts et al., 2009) ou encore en écologie (Cutler et al., 2007). Avant de présenter plus en détails ces méthodes, nous définissons le cadre mathématique dans lesquels s'inscrivent ces travaux de thèse.

1.1.2 Apprentissage supervisé

Soit $\mathcal{D}_n = \{(\mathbf{X}_1, Y_1), \ldots, (\mathbf{X}_n, Y_n)\}$ un échantillon d'apprentissage, c'est-à-dire n copies indépendantes du couple de variables aléatoires $(\mathbf{X}, Y)$. Le couple $(\mathbf{X}, Y)$ est indépendant de $\mathcal{D}_n$ et sa loi est inconnue. Notons $\mathcal{X}$ et $\mathcal{Y}$ les espaces mesurables dans lesquels vivent respectivement les variables aléatoires $\mathbf{X}$ et Y. Dans ce manuscrit, nous considérons le cas $\mathcal{X} = \mathbb{R}^d$. La variable $\mathbf{X} = (X_1, \ldots, X_d)$ désigne le vecteur des variables explicatives et Y est la variable réponse.

Dans le cadre de la régression, la réponse Y est continue, $\mathcal{Y} = \mathbb{R}$, et le modèle statistique s'écrit sous la forme $Y = f^\star(\mathbf{X}) + \varepsilon$ où ε appelé bruit est une variable aléatoire supposée centrée conditionnellement à $\mathbf{X}$, et $f^\star$ est la fonction de régression inconnue définie sur $\mathcal{X}$ par $f^\star(\mathbf{x}) = \mathbb{E}[Y|\mathbf{X} = \mathbf{x}]$.

Dans le cadre de la classification supervisée, Y désigne la classe avec $\mathcal{Y} = \{1, \ldots, K\}$, $K \geq 2$, et $f^\star$ est le classifieur de Bayes (inconnue), définie sur $\mathcal{X}$ par $f^\star(\mathbf{x}) = \text{argmax}_{k \in \{1,\ldots,K\}} \mathbb{P}[Y = k|\mathbf{X} = \mathbf{x}]$.

Dans chacun de ces deux contextes, le problème consiste à estimer le lien entre le vecteur $\mathbf{X}$ et la variable réponse Y, c'est-à-dire à estimer la fonction $f^\star$ à partir des données de l'échantillon d'apprentissage $\mathcal{D}_n$. Un estimateur de $f^\star$ est une fonction mesurable $\hat{f} : \mathcal{X} \times (\mathcal{X} \times \mathcal{Y})^n \to \mathcal{Y}$ qui, prédit pour toute nouvelle observation $\mathbf{x}$ la valeur de la réponse Y par $\hat{f}(\mathbf{x}, \mathcal{D}_n)$. Dans la suite, on notera par commodité $\hat{f}(\mathbf{x})$. La fonction $\hat{f}$ est appelée règle de prédiction ou règle de décision. Un ensemble d'ouvrages de référence traite de la problématique de l'apprentissage supervisé, voir par exemple Devroye et al. (1996), Vapnik (1995, 1998) et Hastie et al. (2009).

Dans de nombreux problèmes en apprentissage supervisé, les variables explicatives peuvent avoir une structure de groupe. Le regroupement des variables peut être naturel ou bien défini dans le but de capturer/modéliser les relations entre les différentes variables. Les variables explicatives peuvent agir en groupes sur la variable réponse. Ainsi l'exploitation d'une telle structure peut s'avérer très utile pour construire une règle de prédiction.

Dans ce présent travail de thèse, nous nous intéressons au cas où le vecteur $\mathbf{X}$ est structuré en J groupes connus. On définit le j-ième groupe $\mathbf{X}^j$, $j = 1, \ldots, J$, par :

$$\mathbf{X}^j = (X_{j_1}, X_{j_2}, \ldots, X_{j_{d_j}}),$$

où l'ensemble $\{j_1, j_2, \ldots, j_{d_j}\} \subseteq \{1, \ldots, d\}$ désigne les d_j indices des variables explicatives appartenant au groupe j, $d_j \leq d$. On remarquera que les groupes ne sont pas forcément disjoints. L'objectif est d'utiliser cette structure pour construire une règle de prédiction $\hat{f}$.

1.2 Arbres de décision et forêts aléatoires

Cette section présente les arbres de décision (plus précisément la méthode CART) et les forêts aléatoires. Genuer & Poggi (2018) proposent un exposé clair et concis de ces méthodes.

Ces méthodes d'apprentissage supervisé ne tiennent pas compte de la structure groupée des données. Ainsi dans cette section, nous ignorons l'existence des groupes.

Les arbres de décision ou méthodes de partitionnement récursif ont été introduits dès les années 60. De nombreuses approches ont été proposées. La méthode CART (pour Classification And Regression Trees), introduite par Breiman et al. (1984) est la plus connue. Nous présentons cette méthode dans la section suivante.

1.2.1 Arbres CART

CART est une méthode non-paramétrique efficace, simple à implémenter et utilisable à la fois en régression et en classification. Le principe général de CART est de construire une règle de prédiction au moyen d'un partitionnement récursif et binaire de l'espace des données. La partition ainsi obtenue peut être représentée sous la forme d'un arbre binaire facilement interprétable. La Figure 1.1 illustre la correspondance entre une partition dyadique et un arbre binaire.

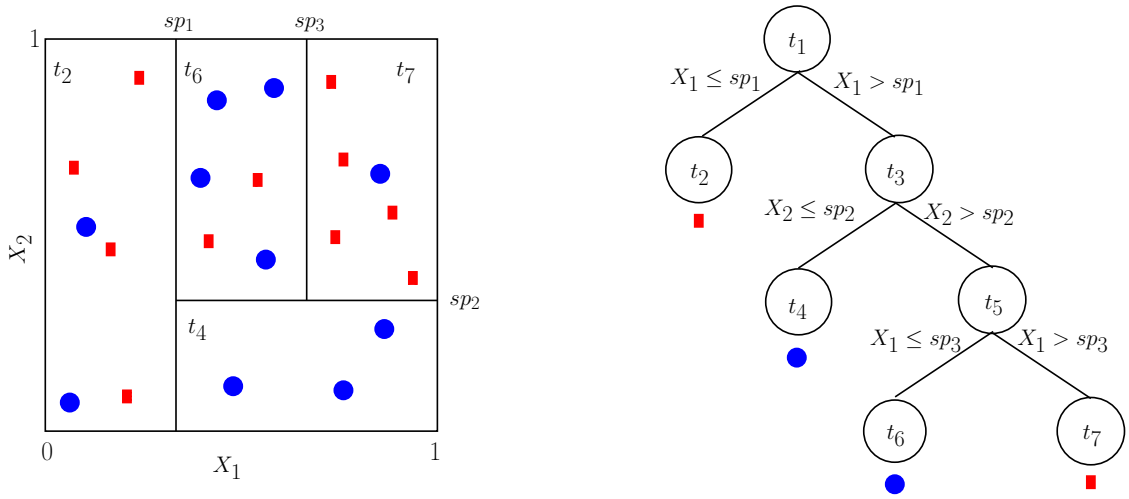


Figure 1.1: Un exemple d'arbre CART en classification binaire. A chaque feuille est associée la classe la mieux représentée.

Construction d'un arbre CART

Pour construire un arbre CART à partir des données de l'échantillon $\mathcal{D}_n$ d'apprentissage, l'algorithme procède en deux étapes.

- Étape 1 : Élaboration d'un arbre maximal.
  Cette étape consiste en un partitionnement récursif et dyadique de l'espace des données $\mathcal{X}$. Au départ, l'espace $\mathcal{X}$ tout entier est associé à la racine de l'arbre, que l'on note t_1. L'algorithme commence par diviser la racine t_1 en deux sous-espaces disjoints t_{1L} et t_{1R} (appelés nœuds fils) comme suit :

$$t_{1L} = \{\mathbf{X}_i, i \leq n : X_{ij} \leq sp\} \quad \text{and} \quad t_{1R} = \{\mathbf{X}_i, i \leq n : X_{ij} > sp\},$$

  où $j = 1, \ldots, p$ et $sp \in \mathbb{R}$. Une division δ est donc définie par un couple $\delta = (j, sp)$ où j désigne l'indice de la variable de coupure et sp désigne une valeur seuil pour cette variable. Le choix de ce couple repose sur la définition d'une fonction Q d'impureté. La méthode sélectionne la coupure $\delta^\star_{t_1}$ qui maximise la décroissance d'impureté définie par

$$\Delta \mathrm{Q}(t_1, \delta) = n_{t_1} \mathrm{Q}(t_1) - n_{t_{1R}} \mathrm{Q}(t_{1R}) - n_{t_{1L}} \mathrm{Q}(t_{1L}), \tag{1.1}$$

  où t_{1L} et t_{1R} désignent les deux nœuds fils de t_1 définis par la coupure δ et n_{t_1} (respectivement $n_{t_{1L}}$ et $n_{t_{1R}}$) désigne le nombre d'observations dans la racine t_1 (respectivement dans les nœuds fils t_{1L} et t_{1R}). En régression, la fonction d'impureté $\mathrm{Q}(t)$ correspond le plus souvent à la variance du nœud t :

$$\mathrm{Q}(t) = \frac{1}{n_t} \sum_{i:\mathbf{X}_i \in t} (Y_i - \bar{Y}_t)^2,$$

où $\bar{Y}_t$ est la moyenne des Y_i des observations contenues dans le nœud t. En classification, l'indice de Gini est généralement utilisé pour définir l'impureté d'un nœud t :

$$\mathrm{Q}(t) = \sum_{k=1}^{K} \pi_k(t)(1 - \pi_k(t)),$$

où $\pi_k(t) = \frac{1}{n_t} \sum_{i:\mathbf{X}_i \in t} \mathbb{1}_{Y_i=k}$ est la proportion d'observations de la classe k dans le nœud t. Dans les deux cas, l'objectif est de partager les observations de l'échantillon $\mathcal{D}_n$ en deux groupes disjoints les plus homogènes possible au sens de la variable réponse Y.

Une fois la racine de l'arbre découpée, la procédure est répétée sur chaque nœud fils, puis de manière récursive sur tous les autres nœuds jusqu'à ce que chaque nœud soit homogène, c'est-à-dire que toutes les observations contenues dans le nœud partagent la même valeur pour Y. Les nœuds terminaux, qui ne sont pas découpés sont appelés feuilles. A la fin du découpage, les feuilles forment une partition fine de l'espace des données $\mathcal{X}$, qui peut être représentée sous la forme d'un arbre maximal, noté $T_{\max}$. Une prédiction $\hat{y}_t$ est associée à chaque feuille t de l'arbre $T_{\max}$ (la moyenne empirique de la réponse Y dans le nœud t en régression ou en classification, la classe de Y la mieux représentée dans le nœud t). De l'arbre $T_{\max}$, on déduit alors la règle de prédiction notée $\hat{f}_{T_{\max}}$ et définie, pour toute observation $\mathbf{x} \in \mathcal{X}$, par

$$\hat{f}_{T_{\max}}(\mathbf{x}) = \sum_{t \in \widetilde{T}_{\max}} \hat{y}_t \mathbb{1}_t(\mathbf{x}),$$

où $\widetilde{T}_{\max}$ désigne l'ensemble des nœuds terminaux de $T_{\max}$ et $\mathbb{1}_t(\mathbf{x})$ désigne la fonction indicatrice égale à 1 si $\mathbf{x} \in t$ et 0 sinon (voir Figure 1.1).

- Étape 2 : Élagage et sélection de l'arbre final.
  L'arbre maximal T_{max} souvent trop complexe n'est généralement pas optimal au sens d'un critère de performance choisi (par exemple en classification, l'erreur de classification). Un nombre excessif de coupures conduit à un arbre qui a tendance à sur-ajuster. Pour éviter cela, $T_{\max}$ est élagué suivant la méthode *minimal cost-complexity pruning* introduite par (Breiman et al., 1984).

Ce procédé consiste à extraire une suite de sous-arbres de $T_{\max}$ par minimisation du critère pénalisé défini pour tout sous-arbre T de $T_{\max}$, noté $T \preceq T_{\max}$, et pour tout $\alpha \in \mathbb{R}^+$ par

$$\mathcal{R}_\alpha(T) \;=\; \mathcal{R}(T, \mathcal{D}_n) \;+\; \alpha|\widetilde{T}|, \tag{1.2}$$

où $|\widetilde{T}|$ désigne le nombre de feuilles de l'arbre T et $\mathcal{R}(T, \mathcal{D}_n)$ correspond à l'erreur empirique du modèle T estimée à partir des données de l'échantillon $\mathcal{D}_n$. En régression, $\mathcal{R}(T, \mathcal{D}_n)$ désigne le critère des moindres carrés

$$\mathcal{R}(T, \mathcal{D}_n) = \frac{1}{n} \sum_{i:(\mathbf{X}_i, Y_i) \in \mathcal{D}_n} (Y_i - \hat{f}_T(\mathbf{X}_i))^2,$$

et en classification, $\mathcal{R}(T, \mathcal{D}_n)$ désigne l'erreur de classification

$$\mathcal{R}(T, \mathcal{D}_n) = \frac{1}{n} \sum_{i:(\mathbf{X}_i, Y_i) \in \mathcal{D}_n} \mathbb{1}_{Y_i \neq \hat{f}_T(\mathbf{X}_i)}.$$

Dans l'équation (1.2), α est un paramètre à régler/à choisir. Il permet de contrôler la complexité de l'arbre. Plus α est grand, plus les arbres ayant beaucoup de feuilles sont pénalisés.

La méthode d'élagage consiste à trouver pour toute valeur $\alpha \in \mathbb{R}$, le plus petit sous-arbre de $T_{\max}$ optimal au sens du critère pénalisé (1.2). Une recherche exhaustive de chaque arbre optimal se révèle souvent trop coûteuse. Aussi, Breiman et al. (1984) propose une stratégie efficace, qui repose sur le résultat suivant.

Theorem 1.2.1. *(Breiman et al., 1984). Pour tout arbre maximal $T_{\max}$, il existe une suite finie et strictement croissante de paramètres*

$$0 = \alpha_1 < \ldots < \alpha_K$$

associée à une suite de sous-arbres emboités $T_{\max} \succeq T_1 \succ \ldots \succ T_K = \{t_1\}$ tous élagués de $T_{\max}$ et vérifiant pour tout $1 \leq k < K$,

$$\textit{pour tout } \alpha \in [\alpha_k; \alpha_{k+1}[, \quad T_k \ = \ \underset{T \preceq T_{\max}}{\operatorname{argmin}} \mathcal{R}_\alpha(T),$$

et

$$\textit{pour tout } \alpha \geq \alpha_K, \quad T_K \ = \ \underset{T \preceq T_{\max}}{\operatorname{argmin}} \mathcal{R}_\alpha(T).$$

Ainsi, l'extraction de la suite d'arbres optimaux repose sur un nombre fini de valeurs pour α et chaque arbre de la suite est obtenu par élagage du précédent. En d'autres termes, pour chaque $k = 1, \ldots, K$, T_k est le plus petit sous arbre de T_{k-1} minimisant $\mathcal{R}_{\alpha_k}$ (en posant ici $T_0 = T_{\max}$). De plus, la suite $\{T_k\}_{1 \leq k \leq K}$ contient toute l'information puisque pour tout $\alpha \geq 0$, le plus petit sous-arbre optimal au sens de $\mathcal{R}_\alpha$ est contenu dans la suite.

L'arbre final est le meilleur sous-arbre de la suite $\{T_k\}_{1 \leq k \leq K}$ au sens d'un critère donné et évalué sur un échantillon témoin ou par validation croisée. Des garanties théoriques justifiant la stratégie d'élagage et la sélection de l'arbre final ont été obtenues en régression (Gey & Nedelec, 2005) et en classification (Gey, 2012).

Importance des variables

En parallèle, CART fournit aussi une mesure d'importance de chaque variable explicative X_j, avec $j = 1, \ldots, J$. Ce score défini par Breiman et al. (1984), permet, relativement à un arbre CART donné T, de hiérarchiser les variables explicatives $X_1, \ldots, X_J$. Le calcul de ce score est détaillé ci-dessous.

Pour tout nœud t non-terminal de l'arbre T, et toute variable X_j, avec $J = 1, \ldots, J$, on détermine la division par substitution, notée $\delta_t^s(j) = (j, sp^s)$, pour la variable X_j et le nœud t comme la coupure qui *se rapproche* le plus de la coupure optimale $\delta_t^\star$, c'est-à-dire

$$\delta_t^s(j) \;=\; \underset{\delta=(j,sp):sp\in\mathbb{R}}{\operatorname{argmax}} \; \max\left\{ p_{LL}(\delta, \delta_t^\star) + p_{RR}(\delta, \delta_t^\star)\,,\; p_{RL}(\delta, \delta_t^\star) + p_{LR}(\delta, \delta_t^\star) \right\},$$

où :

- $p_{LL}(\delta, \delta_t^\star)$ (respectivement $p_{RR}(\delta, \delta_t^\star)$) est l'estimateur de la probabilité que les divisions δ et $\delta_t^\star$ envoient une observation du nœud t dans son nœud fils gauche t_L (respectivement droit t_R).
- $p_{LR}(\delta, \delta_t^\star)$ (respectivement $p_{RL}(\delta, \delta_t^\star)$) est l'estimateur de la probabilité qu'une observation du nœud t soit envoyée dans le nœud gauche (respectivement droit) par la division δ et envoyée dans le nœud droit (respectivement gauche) par la division $\delta_t^\star$.

L'importance de la variable j dans le nœud t correspond alors au gain d'homogénéité induit par la division $\delta_t^s(j)$, défini par :

$$\mathcal{I}_{\text{cart}}(j, t) \;=\; \Delta\text{Q}(t, \delta_t^s(j)).$$

Ainsi, on définit l'importance de la variable X_j, relativement à l'arbre T, comme la somme sur les nœuds non-terminaux de T des importances "locales", c'est-à-dire

$$\mathcal{I}_{\text{cart}}(j, T) \;=\; \sum_{t \,\in\, T\setminus\widetilde{T}} \Delta\text{Q}(t, \delta_t^s(j)), \tag{1.3}$$

où on rappelle que $\widetilde{T}$ désigne l'ensemble des nœuds terminaux de l'arbre T. Généralement, on ramène cette importance sur une échelle comprise entre 0 et 100 par :

$$\widetilde{\mathcal{I}}_{\text{cart}}(j, T) = 100 \times \frac{\mathcal{I}_{\text{cart}}(j, T)}{\max\limits_{j'=1,\ldots,d} \mathcal{I}_{\text{cart}}(j', T)}.$$

Ceci induit un ordre "d'importance" sur les variables explicatives. En effet, on considère comme importantes les variables dont l'importance est supérieure à un seuil choisi.

Ce score a été utilisé par quelques auteurs pour faire de la sélection de variables (Questier et al., 2005; Tuleau & Poggi, 2006). Souvent considéré comme instable (Ghattas et al., 2000), cet indice d'importance est aujourd'hui peu utilisé.

Extensions : d'autres arbres de décision

Bien que CART soit la méthode par arbre la plus utilisée, il existe d'autres méthodes de partitionnement comme par exemple CHAID (Kass, 1980), ID3 (Quinlan, 1986) ou encore C4.5 (Quinlan, 1993) (voir Loh (2014) pour une synthèse des principaux arbres de décision). Comme CART, ces méthodes sont basées sur un partitionnement récursif de l'espace des données. Cependant, la stratégie de découpage et les règles d'arrêt utilisées peuvent être différentes. Des approches utilisant notamment des coupures multivariées, c'est-à-dire des coupures définies en fonction d'un sous-ensemble de variables explicatives, ont également été développées, par exemple CART-LC (Breiman et al., 1984), FACT (Wei-Yin Loh, 1988), QUEST (Loh & Shih, 1997), LTDS (Li et al., 2003) ou encore HHCART (Wickramarachchi et al., 2016). Dans la plupart des ces méthodes, la règle de coupure d'un nœud est une combinaison linéaire des variables explicatives. Ces algorithmes ont souvent de meilleures performances prédictives que les approches utilisant des coupures univariées (Brodley & Utgoff, 1995; Lim et al., 2000). Cependant, ils possèdent deux défauts majeurs. Tout d'abord, dans ces méthodes, la recherche de la coupure optimale, c'est-à-dire de la meilleure combinaison linéaire de variables, implique souvent le recours à des algorithmes coûteux. On peut citer par exemple le *Tabu search algorithm* utilisé par la méthode LTDS. D'autre part, les coupures optimales, qui sont des combinaisons linéaires de variables, sont souvent difficiles à interpréter. Ces deux faiblesses expliquent en partie pourquoi les arbres de décision basés sur des coupures multivariées sont moins souvent utilisés en pratique.

1.2.2 Forêts aléatoires

Les méthodes de partitionnement et particulièrement la méthode CART connaissent un succès important. Elles sont maintenant couramment utilisées dans de nombreux domaines notamment dans le domaine médical (Sathyadevi, 2011) ou en écologie (Pesch et al., 2011). Cependant, ces méthodes s'avèrent être très instables. En effet, une simple perturbation de quelques observations dans l'échantillon d'apprentissage peut modifier complètement l'arbre ainsi construit. Les forêts aléatoires de Breiman (2001) permettent de résoudre cette faiblesse des arbres de décision et en améliorent les performances prédictives. Nous rappelons maintenant les grands principes de cette méthode. Il existe plusieurs méthodes de forêts aléatoires, aussi pour fixer les idées, dans le manuscrit, le terme forêts aléatoires désigne la méthode des forêts aléatoires introduite par Breiman (voir les thèses de Genuer (2010) et de Scornet (2015) pour un panorama complet des différents modèles de forêts).

Algorithme des forêts aléatoires

Les forêts aléatoires sont basées sur le bagging (Breiman, 1996), approche qui consiste à agréger une collection d'estimateurs construits à partir d'échantillons bootstrap. Une forêt aléatoire est une agrégation d'arbres aléatoires. Le principe de construction d'une forêt

est tout d'abord de générer indépendamment un grand nombre (noté `ntree`) d'échantillons bootstrap $\mathcal{D}_n^1, \ldots, \mathcal{D}_n^{\texttt{ntree}}$ en tirant aléatoirement, pour chacun d'eux, a_n observations (avec ou sans remise) dans l'échantillon $\mathcal{D}_n$ d'apprentissage. Ensuite, `ntree` arbres de décision $T^1, \ldots, T^{\texttt{ntree}}$ sont construits à partir des échantillons bootstrap $\mathcal{D}_n^1, \ldots, \mathcal{D}_n^{\texttt{ntree}}$ et en utilisant une variante de CART. En effet chaque arbre est ici construit de la façon suivante. Pour découper un nœud, l'algorithme choisi aléatoirement et sans remise un nombre `mtry` de variables explicatives, puis il détermine la meilleure coupure uniquement suivant les `mtry` variables sélectionnées. De plus, l'arbre construit est pleinement développé et n'est pas élagué. La forêt aléatoire, que l'on note $\{T^b\}_1^{\texttt{ntree}}$, est enfin obtenue en agrégeant les `ntree` arbres ainsi construits. Elle définit une règle de prédiction qui correspond à la moyenne empirique des prédictions en régression et au vote majoritaire en classification. La construction des forêts aléatoires de Breiman est décrite dans par l'Algorithme 1.

Algorithm 1 Forêts aléatoires.

Input : Échantillon d'apprentissage $\mathcal{D}_n$, `ntree` $\in \mathbb{N}$, $a_n \in \{1, \ldots, n\}$, `mtry` $\in \{1, \ldots, d\}$, `nodesize` $\in \{1, \ldots, n\}$.

For $b = 1$ **to** `ntree` **do**

1. Construction de l'échantillon bootstrap $\mathcal{D}_n^b$: tirer uniformément et avec remise a_n observations dans $\mathcal{D}_n^b$.
2. Construction de l'arbre T^b à partir de $\mathcal{D}_n^b$: répéter de manière récursive le procédé suivant sur chaque nœud, jusqu'à ce que chaque nœud soit homogène ou contienne moins de `nodesize` observations :
   (a) Tirer un sous-ensemble $\mathcal{M}_{\texttt{mtry}} \subset \{X_1, \ldots, X_d\}$ de cardinal `mtry` uniformément et sans remise.
   (b) Choisir la meilleure coupure au sens du critère de coupure de CART (1.1) et en se basant uniquement sur le sous-ensemble $\mathcal{M}_{\texttt{mtry}}$ de variables.
   (c) Diviser le nœud en deux nœuds fils selon la coupure précédemment choisie.
3. Définition de la règle de prédiction $\hat{f}_b$ à partir de l'arbre maximal T^b.

Output: la collection d'arbres $T^1, \ldots, T^{\texttt{ntree}}$ et la collection associée de règles de prédiction $\hat{f}_1, \ldots, \hat{f}_{\texttt{ntree}}$.

Prédiction de la forêt aléatoire en $\mathbf{x} \in \mathcal{X}$:

En régression : $\hat{f}_{\texttt{rf}}(\mathbf{x}) = \frac{1}{\texttt{ntree}} \sum_{b=1}^{\texttt{ntree}} \hat{f}_b(\mathbf{x})$,

En classification : $\hat{f}_{\texttt{rf}}(\mathbf{x}) = \underset{k=1,\ldots,K}{\operatorname{argmax}} \left(\sum_{b=1}^{\texttt{ntree}} \mathbb{1}_{\hat{f}_b(\mathbf{x})=k} \right)$.

L'algorithme des forêts aléatoires comporte plusieurs paramètres.

- Le nombre d'arbres `ntree` de la forêt. Sa valeur par défaut est 500. Notons que ce paramètre n'est pas vraiment un paramètre à calibrer dans le sens où une plus grande valeur de ce paramètre mènera toujours à des prédictions plus stables qu'une plus petite valeur de ce paramètre.

- Le nombre `mtry` de variables choisies pour le découpage de chaque nœud. Sa valeur par défaut est $\texttt{mtry} = d/3$ en régression et $\texttt{mtry} = \sqrt{d}$ en classification. C'est sans doute le paramètre le plus important à calibrer puisqu'il peut grandement influencer les performances de la forêt.

- Le nombre minimum d'observations `nodesize` en dessous duquel un nœud n'est plus découpé. La valeur par défaut de ce paramètre est $\texttt{nodesize} = 1$ en classification et $\texttt{nodesize} = 5$ en régression. En général, ce paramètre est laissé à sa valeur par défaut.

- Le nombre d'observations a_n dans chaque échantillon bootstrap. Par défaut, chaque échantillon bootstrap contient $a_n = n$ observations tirées avec remise dans l'échantillon initial $\mathcal{D}_n$.

Plusieurs auteurs se sont intéressés au choix et à l'influence de ces paramètres (Breiman, 2001; Díaz-Uriarte & Alvarez de Andrés, 2006; Genuer, 2010; Bernard et al., 2008; Biau & Scornet, 2016). En général, les valeurs par défaut des paramètres donnent de bons résultats.

Les forêts aléatoires connaissent aujourd'hui un large succès. La méthode a permis de résoudre efficacement un grand nombre de problèmes dans des domaines variés comme par exemple en écologie (Prasad et al., 2006), en bioinformatique (Díaz-Uriarte & Alvarez de Andrés, 2006), ou encore en analyse d'image (Shotton et al., 2011). Outre ses très bonnes performances et sa large applicabilité, la méthode ne dépend que d'un petit nombre de paramètres ce qui la rend aussi facilement utilisable. D'un point de vue théorique, l'étude des propriétés mathématiques des forêts aléatoires se révèle plus délicate. En effet, il existe peu de résultats théoriques disponibles pour les forêts aléatoires de Breiman. On peut néanmoins citer un résultat majeur établi récemment par Scornet et al. (2015) et portant sur la convergence des forêts aléatoires dans le modèle additif. Des garanties théoriques ont également été obtenues pour des versions simplifiées de la méthode (Breiman, 2001; Biau, 2012; Genuer, 2012; Wager, 2014). Une synthèse des principaux résultats théoriques est disponible dans Genuer & Poggi (2018) et Biau & Scornet (2016).

Importance des variables

Les forêts aléatoires sont en général plus performantes que les simples arbres de décision mais possèdent l'inconvénient d'être plus difficilement interprétables. Afin de pallier à cela, plusieurs indices d'importance des variables sont définis. Ces scores permettent d'établir une

hiérarchie des variables explicatives fondée sur l'importance par rapport à la réponse Y. La méthode des forêts aléatoires propose principalement deux critères : l'importance de Gini et l'importance par permutation.

L'indice d'importance de Gini se rapproche du score d'importance (1.3) proposé dans CART, excepté qu'il n'utilise pas les coupures par substitution. Cet indice est défini à partir du critère d'impureté (1.1) utilisé lors de la construction d'un arbre. L'importance d'une variable est d'abord évaluée sur chaque arbre de la forêt. Ainsi, pour un arbre donné, elle correspond à la réduction *globale* d'impureté c'est-à-dire à la somme pondérée des réductions d'impureté induites lorsque que la variable est utilisée pour découper un nœud du dit arbre. L'importance de Gini d'une variable est alors définie par la moyenne (sur tous les arbres de la forêt) des réductions *globales* d'impureté. En pratique, cet indice est moins utilisé que l'indice par permutation. Différents auteurs ont montré par des études de simulations que cet indice avait tendance à favoriser les variables catégorielles qui ont beaucoup de modalités ou bien dont les effectifs sont déséquilibrés (Strobl et al., 2007; Nicodemus, 2011; Boulesteix et al., 2011).

L'indice d'importance par permutation (Breiman, 2001) repose sur l'idée qu'une variable explicative peut être considérée comme importante pour prédire la réponse Y si briser le lien entre cette variable et la réponse Y détériore la qualité de la prédiction. En ce sens, des permutations aléatoires des valeurs de la variable sont utilisées pour imiter la rupture de ce lien. Formellement, le calcul de la mesure d'importance par permutation pour une variable X_j (avec $j = 1, \ldots, d$) consiste tout d'abord à définir l'échantillon *out-of-bag* (OOB) associé à chaque échantillon bootstrap. Pour $b = 1, \ldots, \texttt{ntree}$, le b-ième échantillon OOB est noté $\bar{\mathcal{D}}_n^b$ et est défini comme le complémentaire du b-ième échantillon bootstrap $\mathcal{D}_n^b$ dans $\mathcal{D}_n$: $\bar{\mathcal{D}}_n^b = \mathcal{D}_n \setminus \mathcal{D}_n^b$. En d'autres termes, $\bar{\mathcal{D}}_n^b$ contient les observations de $\mathcal{D}_n$ n'appartenant pas à $\mathcal{D}_n^b$. Cet échantillon est utilisé pour mesurer l'erreur du b-ème arbre T^b. On note $\mathcal{R}(T^b, \bar{\mathcal{D}}_n^b)$ l'erreur empirique de T^b estimée sur $\bar{\mathcal{D}}^b$. (Comme mentionné dans la section 1.2.1, l'erreur empirique d'un arbre correspond au critère des moindres carrés en régression ou à l'erreur de classification en classification). Ensuite, l'algorithme définit l'échantillon OOB permuté $\bar{\mathcal{D}}_n^{bj}$ qui est obtenu en permutant aléatoirement les valeurs de la variable X_j. L'erreur de l'arbre est à nouveau calculée mais cette fois-ci en utilisant l'échantillon permuté $\bar{\mathcal{D}}_n^{bj}$. Elle est notée $\mathcal{R}(T^b, \bar{\mathcal{D}}_n^{bj})$.
Ces étapes sont répétées sur tous les arbres de la forêt. L'indice d'importance correspond alors à la moyenne sur tous les arbres de l'augmentation de l'erreur :

$$\mathcal{I}_{\text{perm}}(X_j\,,\,\{T^b\}_1^{\texttt{ntree}}) \;=\; \frac{1}{\texttt{ntree}} \sum_{b=1}^{\texttt{ntree}} \mathcal{R}(T^b, \bar{\mathcal{D}}_n^b) - \mathcal{R}(T^b, \bar{\mathcal{D}}_n^{bj}),$$

Si la permutation aléatoire de la j-ème variable induit une forte augmentation de l'erreur alors $\mathcal{I}_{\text{perm}}(X_j\,,\,\{T^b\}_1^{\texttt{ntree}})$ est grand et la variable est considérée comme importante. A l'inverse, si les perturbations n'affectent pas l'erreur, alors l'indice d'importance par permutation de X_j est proche de zéro et la variable est considérée comme peu importante pour prédire la

réponse Y. L'indice d'importance par permutation a été étudié par de nombreux auteurs (Archer & Kimes, 2008; Altmann et al., 2010; Gregorutti et al., 2013) et s'est notamment avéré être très utile pour faire de la sélection de variables (Díaz-Uriarte & Alvarez de Andrés, 2006; Genuer et al., 2010). Récemment, Gregorutti et al. (2015) ont adapté cet indice aux groupes de variables, dans le but de sélectionner des variables fonctionnelles. Cette extension sera définie et utilisée dans la suite du manuscrit.

1.3 Organisation du manuscrit et contributions

Dans les sections qui suivent, nous résumons le contenu des différents chapitres ainsi que les résultats obtenus.

1.3.1 Chapitre 2 : Arbres de classification pour variables groupées

Dans ce chapitre, nous proposons une nouvelle approche par arbres permettant de construire des règles de classification à partir de variables groupées. Cette méthode, appelée TPLDA (pour Tree Penalized Linear Discriminant Analysis), construit d'abord un arbre maximal au moyen d'un partitionnement récursif et binaire de l'espace des données $\mathcal{X}$. La division d'un nœud, qui est définie par un groupe de variables et une combinaison linéaire (aussi appelée *règle de division*) des variables appartenant au dit groupe, se fait en deux temps.

- Étape 1 : Choix d'une coupure pour chaque groupe.
  Tout d'abord l'algorithme définit, pour chaque groupe, une règle de division en utilisant l'analyse linéaire discriminante régularisée proposée par Witten & Tibshirani (2011). Le recours à cette approche régularisée s'appuie sur des résultats numériques et théoriques de la littérature. Une étude par simulations est aussi effectuée pour justifier ce choix.

- Étape 2 : Choix de la meilleure coupure.
  Dans CART comme dans beaucoup de méthodes de partitionnement, la sélection de la coupure repose sur la maximisation de la réduction d'impureté (1.1). Lorsque les coupures sont définies en fonction d'un groupe de variables, ce critère n'est plus approprié puisqu'il a tendance à favoriser les plus grands groupes (Boulesteix et al., 2012). Afin de corriger ce biais de sélection, nous proposons de corriger la réduction d'impureté par une pénalité pen dépendant de la taille du groupe. Soit δ_j une coupure définie à partir des variables du groupe j et t un nœud. La réduction d'impureté pénalisée induite par la division de t par la coupure δ_j est définie par

$$\Delta_p \mathrm{Q}(t, \delta_j) = \Delta \mathrm{Q}(t, \delta_j) \mathsf{pen}(d_j), \tag{1.4}$$

où $\mathsf{pen}(d_j)$ est la fonction de pénalité évaluée en la taille d_j du groupe j. La forme de la fonction pen sera définie précisément dans le **Chapitre 2**. Plusieurs fonctions de pénalités sont notamment proposées. Une analyse de simulation est effectuée pour évaluer ce critère. Nous montrons que la pénalité permet de contrôler le biais de sélection et que le critère pénalisé est adapté à la sélection de coupures multivariées.

Les deux étapes précédentes sont appliquées tout d'abord sur l'espace des observations $\mathcal{X}$. Il en résulte la création de deux nœuds fils. Ces deux étapes sont ensuite répétées sur ces deux nœuds, puis de manière récursive sur tous les autres nœuds, et ce jusqu'à atteindre un critère d'arrêt. La partition fine des données ainsi construite peut être représentée sous la forme d'arbre maximal, qui a tendance à sur-ajuster. Afin d'éviter cela, une méthode d'élagage est proposée pour sélectionner un arbre optimal. Cette procédure construit une suite de sous-arbres emboités, tous élagués de $T_{\max}$ par maximisation d'un critère basé sur la profondeur de l'arbre. L'algorithme sélectionne ensuite le meilleur arbre dans la suite en s'appuyant sur un critère de performance donné et estimé sur un échantillon indépendant. Un exemple jouet où l'on considère un seul groupe à deux variables est donné dans la Figure 1.2. La figure montre les partitions obtenues à partir des méthodes TPLDA et CART.

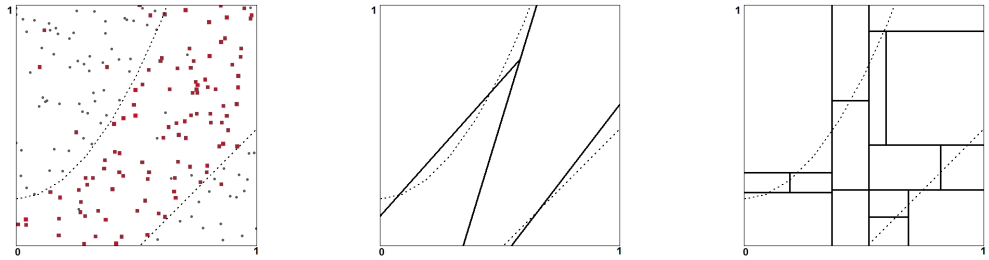


Figure 1.2: Illustration de la méthode TPLDA - un problème de classification avec un seul groupe contenant deux variables. (a) les données d'apprentissage, (b) la partition obtenue avec la méthode TPLDA, (c) la partition obtenue avec la méthode CART. Sur chaque graphe, la règle de Bayes est représentée par deux lignes en pointillés.

Ensuite, comme il est souvent utile d'avoir une information sur les groupes de variables et plus précisément sur leur capacité à expliquer la variable réponse, nous introduisons un nouvel indice d'importance pour les groupes de variables. Cet indice qui est associé à un arbre TPLDA s'appuie sur la réduction d'impureté pénalisée (1.4) évaluée en chaque nœud non terminal de l'arbre.

Un ensemble de simulations est ensuite effectué pour évaluer les performances de la méthode TPLDA et de la mesure d'importance. La méthode TPLDA est comparée à la méthode CART et à la régression logistique régularisée par la pénalité *Group lasso* (GL), qui est une des méthodes de référence pour construire des règles de classification à partir de données groupées. Ces analyses de simulation montrent que TPLDA est une méthode efficace pour construire de règles de prédiction à partir de variables groupées. De plus, la méthode

s'avère être beaucoup moins coûteuse (en terme de complexité informatique) que la plupart des méthodes classiques de partitionnement utilisant des coupures multivariées (par exemple les méthodes HHCART et OC1). D'autre part, dans les analyses de simulation, l'indice d'importance identifie correctement les groupes discriminants, même lorsque les tailles de groupes sont très déséquilibrées ou lorsqu'il y a beaucoup de bruits.

Pour finir ce chapitre, les méthodes TPLDA, CART et GL sont appliquées sur trois jeux de données publiques d'expression de gènes. Cet exemple souligne à nouveau les bonnes performances de la méthode TPLDA et de son indice d'importance des groupes.

Les résultats de ce chapitre font l'objet d'un article soumis dans la revue *Computational Statistics*. De plus, la méthode TPLDA ainsi que l'indice d'importance des groupes ont été implémentés en langage `R` pour la classification binaire.

Contributions du chapitre : développement d'une nouvelle méthode par arbre permettant de construire des règles de classification à partir de variables groupées et introduction d'une mesure d'importance pour les groupes de variables.

1.3.2 Chapitre 3 : Arbres de décision et forêts aléatoires pour variables groupées

Le Chapitre 3 commence par présenter une nouvelle approche par arbre appelée CARTGV (pour Classification And Regression Trees for Grouped Variables) adaptée aux variables groupées. Contrairement à la méthode TPLDA introduite dans le Chapitre 2, cette nouvelle méthode ne fait aucune hypothèse sur la forme de la relation entre les variables au sein d'un groupe. De plus, la méthode CARTGV permet de construire des règles de prédiction en classification ainsi qu'en régression.

La méthode construit un arbre maximal au moyen d'un partitionnement récursif et non binaire de l'espace des données. De façon similaire à la méthode TPLDA, découper un nœud se fait en deux temps.

- Étape 1 : Choix d'une coupure pour chaque groupe.
  Tout d'abord, l'algorithme détermine une coupure pour chaque groupe, en utilisant la méthode de partitionnement de CART. Plus précisément, pour chaque groupe, la méthode construit un arbre CART sur les observations du nœud et les variables du groupe considéré. Cet arbre que l'on appelle *arbre de coupure* est peu profond (de profondeur maximale D_j) et non élagué.

- Étape 2 : Choix de la meilleure coupure.
  Ensuite, l'algorithme choisit la meilleure coupure par maximisation du critère pénalisé (1.4).

Ce procédé est répété récursivement jusqu'à obtenir un arbre pleinement développé.
Un exemple d'arbre et de coupure obtenus à partir de la méthode CARTGV est donné dans la Figure 1.3.

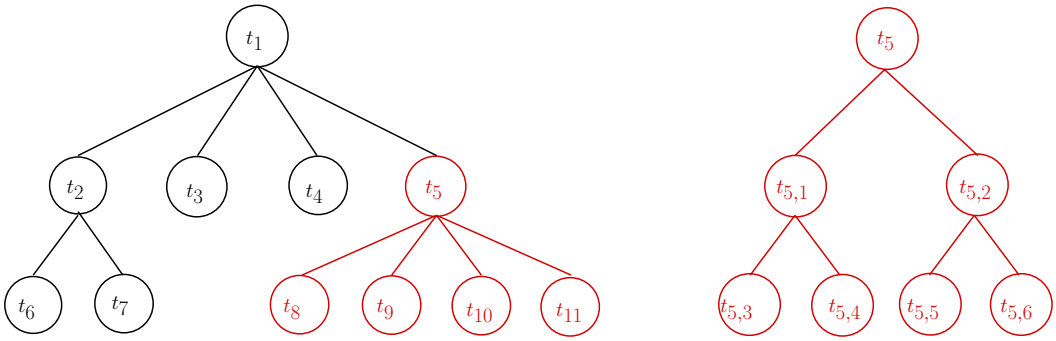


Figure 1.3: Un arbre CARTGV (à gauche) et un arbre de coupure (à droite). L'arbre de coupure divise le nœud t_5 en les nœuds fils : $t_8 = t_{5,3}$, $t_9 = t_{5,4}$, $t_{10} = t_{5,5}$, $t_{11} = t_{5,6}$.

L'arbre maximal, qui n'est plus binaire, est ensuite élagué selon la généralisation de la méthode *minical cost-complexity pruning* aux arbres non binaires. En effet, cette stratégie a initialement été proposée pour élaguer les arbres binaires (Breiman et al., 1984). Nous avons prouvé que cette approche pouvait être utilisée pour les arbres non binaires. Plus précisément, nous avons généralisé le théorème de Breiman et al. (1984).

Theorem 1.3.1. *Pour tout arbre $T_{\max}$ maximal et non nécessairement binaire, il existe une suite finie et strictement croissante de paramètres*

$$0 = \alpha_1 < \ldots < \alpha_K$$

associée à une suite de sous-arbres emboités

$$T_{\max} \succeq T_1 \succ \ldots \succ T_K = \{t_1\},$$

tous élagués de $T_{\max}$ et vérifiant pour tout $1 \leq k < K$,

$$\text{pour tout } \alpha \in [\alpha_k; \alpha_{k+1}[, \quad T_k \ = \ \underset{T \preceq T_{\max}}{\operatorname{argmin}} \mathcal{R}_\alpha(T),$$

et

$$\text{pour tout } \alpha \geq \alpha_K, \quad T_K \ = \ \underset{T \preceq T_{\max}}{\operatorname{argmin}} \mathcal{R}_\alpha(T),$$

où $\mathcal{R}_\alpha$ est le critère pénalisé défini par la formule (1.2).

Ainsi, comme dans le cas des arbres CART, on peut extraire une suite de sous-arbres emboîtés $T_1, \ldots, T_K$ tous élagués de $T_{\max}$ et optimaux au sens du critère d'élagage (1.2). Cette suite est obtenue simplement en coupant de manière itérative des branches à chaque étape. L'algorithme sélectionne ensuite le meilleur sous-arbre de la suite au sens d'un critère de

performance donné et évalué sur un échantillon de validation.

En parallèle, la méthode CARTGV fournit deux mesures d'importance pour chaque groupe de variables. Le premier indice s'appuie sur la réduction d'impureté pénalisée (1.4) évaluée en chaque nœud non terminal de l'arbre et la mesure de proximité appelée indice de Rand. Le second, qui est inspiré de l'indice d'importance proposé par Breiman et al. (1984) pour l'algorithme CART (voir Section 1.2.1 pour une description de cet indice), utilise une extension de la notion de coupure par substitution aux groupes de variables. Ces deux scores permettent de hiérarchiser l'importance de tous les groupes de variables en leur attribuant une note comprise entre 0 et 100.

Une série d'études de simulation est ensuite utilisée pour évaluer la méthode CARTGV ainsi que son indice d'importance. La méthode est comparée à CART, qui peut être considéré comme son analogue dans le cas de variables non-groupées. Ces analyses confirment l'intérêt d'utiliser, quand elle existe, la structure groupée des données pour construire une règle de prédiction. On observe de meilleures performances prédictives avec CARTGV, en particulier lorsque l'information est contenue dans un tout petit nombre de groupes.

Comme toute méthode de partitionnement, CARTGV se révèle souvent instable. Pour cette raison, dans ce chapitre nous introduisons aussi une méthode de forêts aléatoires pour les groupes de variables. A notre connaissance, c'est la première méthode qui permet de construire des forêts aléatoires à partir de données groupées. Cette méthode, que nous appelons RFGV (pour Random Forests for Grouped Variables) est une modification des forêts aléatoires introduites par (Breiman, 2001), dans le sens où la méthode perturbe l'espace des variables à deux niveaux : au niveau des groupes de variables mais aussi au niveau des variables individuelles. Une forêt RFGV consiste en une agrégation d'arbres aléatoires. Tout d'abord, l'algorithme construit un grand nombre (noté `ntree`) d'arbres CARTGV aléatoires à partir d'une collection d'échantillons bootstrap de $\mathcal{D}_n$ indépendants. Chaque arbre est construit selon une variante de CARTGV, dans le sens où à chaque coupure seul un petit nombre de groupes et de variables au sein de ces groupes est utilisé. La forêt est alors définie en agrégeant les arbres ainsi construits. L'Algorithme 2 décrit plus précisément la construction d'une forêt RFGV.

Comme pour tout modèle construit par agrégation, une forêt RFGV ne peut pas être interprétée directement. Pour cette raison, nous utilisons l'indice d'importance des groupes proposé par Gregorutti et al. (2015).

Une série d'analyses de simulations montre les bonnes performances de la méthode. De plus, nous proposons des recommandations quant aux choix des paramètres (notamment le choix de D_j, `mgrp` et $\texttt{mvar}_j$).

Algorithm 2 Forêts aléatoires pour variables groupées.

Input : Échantillon d'apprentissage $\mathcal{D}_n$, $\texttt{ntree} \in \mathbb{N}^\star$, $a_n \in \{1, \ldots, n\}$, $\texttt{mgrp} \in \{1, \ldots, J\}$, $\texttt{nodesize} \in \{1, \ldots, n\}$, $\texttt{mvar}_j \in \{1, \ldots, d_j\}$, $D_j \in \mathbb{N}^\star$, pour tout $j = 1, \ldots, J$.

For $b = 1, \ldots, \texttt{ntree}$ **do**

1. Construction de l'échantillon bootstrap $\mathcal{D}_n^b$: tirer uniformément et avec remise a_n observations dans $\mathcal{D}_n^b$.
2. Construction de l'arbre T^b à partir de $\mathcal{D}_n^b$: répéter de manière récursive le procédé suivant sur chaque nœud, jusqu'à ce que chaque nœud soit homogène ou contienne moins de $\texttt{nodesize}$ observations : considérons le nœud terminal t.
   (a) Tirer un sous-ensemble $\mathcal{J} \subset \{1, \ldots, J\}$ de cardinal $\texttt{mgrp}$ uniformément et sans remise.
   (b) **For** $j \in \mathcal{J}$ **do**
   Choix d'une coupure pour le groupe j : construction d'un *arbre de coupure* sur le nœud t et les variables $X_{j_1}, X_{j_2}, \ldots, X_{j_{d_j}}$ du groupe j, en répétant de manière récursive le procédé suivant sur chaque nœud, jusqu'à ce que l'arbre de coupure atteigne la profondeur maximale D_j :
      i. Tirer un sous-ensemble $\mathcal{M}_j \subset \{X_{j_1}, \ldots, X_{j_{d_j}}\}$ de cardinal $\texttt{mvar}_j$ uniformément et sans remise.
      ii. Choisir dans le sous-ensemble $\mathcal{M}_j$ la coupure optimale au sens du critère de coupure de CART (1.1).
      iii. Couper le nœud en deux nœuds fils selon la coupure précédemment choisie.
   End.
   (c) Choix de la meilleure coupure : choisir selon les coordonnées dans $\mathcal{J}$ le couple groupe/*arbre de coupure* optimal au sens d'un critère d'impureté pénalisé (1.4).
   (d) Couper le nœud t selon la coupure choisie.
3. Définition de la règle de prédiction $\hat{f}_b$ à partir de l'arbre maximal T^b.

End.

Output : la collection d'arbres $T^1, \ldots, T^{\texttt{ntree}}$ et la collection associée de règles de prédiction $\hat{f}_1, \ldots, \hat{f}_{\texttt{ntree}}$.

Prédiction de la forêt aléatoire en $\mathbf{x} \in \mathcal{X}$:

En régression : $\hat{f}_{\texttt{rfgv}}(\mathbf{x}) = \frac{1}{\texttt{ntree}} \sum_{b=1}^{\texttt{ntree}} \hat{f}_b(\mathbf{x})$.

En classification : $\hat{f}_{\texttt{rfgv}}(\mathbf{x}) = \underset{k=1,\ldots,K}{\operatorname{argmax}} \left(\sum_{b=1}^{\texttt{ntree}} \mathbb{1}_{\hat{f}_b(\mathbf{x})=k} \right)$.

Les résultats de ce chapitre font l'objet d'un article prochainement soumis dans la revue *Statistics & Computing*. De plus, les méthodes CARTGV, RFGV ainsi que la généralisation de l'algorithme *minimal cost-complexity pruning* ont été implémentées en langage R pour la classification binaire.

Contributions du chapitre :

- développement d'une méthode par arbre non-paramétrique adaptée aux données groupées,
- généralisation du théorème de (Breiman et al., 1984) et de l'algorithme *minimal cost-complexity pruning* aux arbres de décision non binaires,
- développement d'une méthode de forêts aléatoires pour les variables groupées.

1.3.3 Chapitre 4 : Analyse statistique d'un algorithme de clustering hiérarchique en présence d'outliers

Dans ce chapitre, indépendant des autres, nous nous plaçons dans un cadre de classification non supervisée, une des problématiques majeures en apprentissage statistique.

La classification non supervisée (ou *clustering*) consiste, comme son nom l'indique, à apprendre sans superviseur. L'objectif est d'extraire, à partir d'une population, des classes (groupes, clusters ou encore labels) d'individus présentant des caractéristiques communes, le nombre et la définition des classes n'étant pas donnés *a priori*. Beaucoup d'ouvrages traitent de cette problématique, nous renvoyons par exemple à Hartigan (1975), Rousseeuw & Kaufman (1990), Gordon (1999), ou encore Hastie et al. (2009). Aujourd'hui, il existe un très grand nombre de méthodes en classification non supervisée (les k-means, le clustering spectral, le clustering hiérarchique, l'analyse en composnate principale, etc.). Ces méthodes sont utilisées dans des domaines divers tels que la médecine (Sjostrand et al., 2007), la biologie (Yamanishi et al., 2004; Zeng et al., 2012) et le marketing (Pedrycz, 2002).

Dans ce chapitre, nous proposons un nouvel algorithme de clustering, basé sur la classification ascendante hiérarchique single linkage, et étudions ses propriétés dans un modèle original. Le modèle permet de prendre en compte le fait que les clusters puissent appartenir à des espaces de plus petites dimensions que l'espace des observations (voir Arias-Castro (2011)). De plus notre modèle permet de prendre en compte des *outliers* : des observations qui n'appartiennent à aucun cluster. Le cadre mathématique est proche de ceux proposés par Arias-Castro (2011) et Maier et al. (2009). Les résultats obtenus s'inscrivent dans la continuité de Auray et al. (2015).

Cadre mathématique

On considère $X_1, \ldots, X_n$ n variables aléatoires indépendantes à valeurs dans $\mathbb{R}^d$ et de loi $\mathbb{P}$. La loi $\mathbb{P}$ s'écrit comme un mélange de $M+1$ distributions $\mathbb{P}_0, \ldots, \mathbb{P}_M$ telles que :

$$\mathbb{P} = \varepsilon \mathbb{P}_0 + (1-\varepsilon) \sum_{i=1}^{M} \gamma_i \mathbb{P}_i, \tag{1.5}$$

où $0 \leq \varepsilon < 1$, $\gamma_i > 0$ pour tout $i = 1, \ldots, M$ et $\sum_{i=1}^{M} \gamma_i = 1$. Dans cette décomposition $\mathbb{P}_0$ représente la distribution des outliers et $\mathbb{P}_i$ celle des observations qui appartiennent au i-ème groupe. La proportion d'outliers est représentée par ε et γ_i représente le poids du i-ème cluster. La figure 1.4 propose un exemple de données générées avec $M = 3$ clusters.

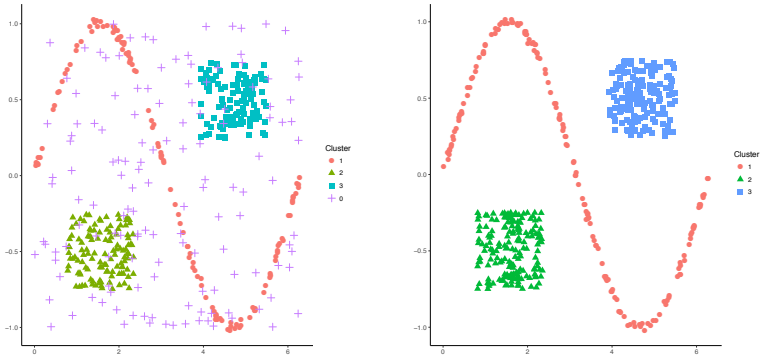


Figure 1.4: Données générées selon le modèle 1.5 avec outliers (gauche) et sans outliers (droite).

On désigne par $S_i, i = 1, \ldots, M$, les supports des lois $\mathbb{P}_i$ et par S_0 le complémentaire de $\bigcup_{i=1}^{M} S_i$ dans $\mathbb{R}^d$. On supposera que chaque $S_i, i = 1, \ldots, M$, est compact et connexe et on désigne par δ la distance (euclidienne) minimale entre deux supports définie par :

$$\delta = \min\{\|x - y\| : x \in S_i, y \in S_j, 1 \leq i < j \leq M\}.$$

Dans ce contexte un algorithme de clustering partitionnera l'ensemble des observations $\{X_1, \ldots, X_n\}$ en $M+1$ clusters $\mathcal{X}_0, \ldots, \mathcal{X}_M$ tels que

$$\bigcup_{i=0}^{M} \mathcal{X}_i = \{X_1, \ldots, X_n\} \quad \text{et} \quad \mathcal{X}_i \cap \mathcal{X}_j = \emptyset \quad \text{pour} \quad i \neq j.$$

Les outliers, qui appartiennent à l'ensemble S_0 peuvent être affectés dans un groupe $\mathcal{X}_i, i = 1, \ldots, M$, ou dans le cluster spécifique $\mathcal{X}_0$. On désigne par $[\![n]\!] = \{1, \ldots, n\}$ et pour tout $I \subseteq [\![n]\!]$, on note $X_I = \{X_i : i \in I\}$. Par conséquent $X_{[\![n]\!]}$ représente l'ensemble des

observations. L'algorithme sera efficace si il existe une permutation π de $\{1, \ldots, M\}$ telle que pour tout $i = 1, \ldots, M$,

$$X_{[\![n]\!]} \cap S_i \subseteq \mathcal{X}_{\pi(i)},$$

avec grande probabilité. Nous mesurerons ainsi la performance d'un algorithme par la probabilité que les observations issues d'un même support $S_i, i = 1, \ldots, M$, ne soient pas contenues dans un même cluster

$$\mathcal{R}_n(\mathcal{X}) = \mathbb{P}^n(\forall \pi \in \Pi_M \ \exists\, i = 1, \ldots, M,\ X_{[\![n]\!]} \cap S_i \nsubseteq \mathcal{X}_{\pi(i)}), \tag{1.6}$$

où $\mathcal{X} = \{\mathcal{X}_0, \ldots, \mathcal{X}_M\}$ désigne la partition issue de l'algorithme. Cette quantité sera appelée *risque de clustering*. Plus ce risque est petit, meilleur est l'algorithme.

L'algorithme *robust single linkage*

Nous proposons une méthode de clustering basée sur l'algorithme *single linkage*, qui permet de prendre en compte des outliers. L'approche consiste à construire une classification ascendante hiérarchique single linkage classique, puis à couper le dendogramme issu de cette classification en maximisant le cardinal du M-ème plus gros cluster. L'algorithme est défini ci-dessous.

Algorithm 3 Robust single linkage clustering

Input : les données $x_1, \ldots, x_n$, le nombre de clusters $M \geq 2$.

Initialisation : $k = 0$, $m_k = n$ et $\mathcal{P}_k = \{\{x_1\}, \ldots, \{x_n\}\}$ (chaque point défini un cluster).

While $m_k \geq M$ **do**

1. Assembler les deux plus proches clusters de $\mathcal{P}_k$ en terme de distance minimale. On obtient une partition $\mathcal{P}_{k+1}$ en m_{k+1} sous-ensembles des données que l'on note $\mathcal{X}_1^{k+1}, \ldots, \mathcal{X}_{m_{k+1}}^{k+1}$.
2. Ordonner les clusters selon leurs tailles : $\#\mathcal{X}_1^{k+1} \geq \#\mathcal{X}_2^{k+1} \geq \ldots \geq \#\mathcal{X}_{m_{k+1}}^{k+1}$.
3. Affecter k à la valeur $k + 1$.

End.

Sélectionner $\hat{k} \in \operatorname{argmax}_k \#\mathcal{X}_M^k$
Définir

$$\mathcal{X}_i = \mathcal{X}_i^{\hat{k}}, \quad i = 1, \ldots, M \qquad \text{et} \qquad \mathcal{X}_0 = \bigcup_{i=M+1}^{m_{\hat{k}}} \mathcal{X}_i^{\hat{k}}.$$

Output : la partition des données $\{\mathcal{X}_0, \mathcal{X}_1, \ldots, \mathcal{X}_M\}$ en $M + 1$ clusters disjoints.

On remarque que l'algorithme requiert la connaissance du nombre de clusters. La boucle **While** est classique. La nouveauté consiste à choisir la sous-partition de la classification hiérarchique qui maximise la taille du M-ème cluster. Les observations qui appartiennent aux clusters de tailles inférieures à ce dernier sont considérées comme des outliers. Cette approche permet notamment d'éviter que des observations qui se retrouvent isolées dans l'espace forment des clusters, reproche souvent fait à l'algorithme single linkage. C'est en ce sens que nous appelons cet algorithme *Robust single linkage clustering*. Pour simplifier le rédaction, dans l'Algorithme 3 les points sont supposés tous distincts les uns des autres.

Dans ce travail, nous étudions aussi la vitesse à laquelle le risque de clustering (1.6) tend vers 0 sous certaines hypothèses concernant :

- la séparabilité et la régularité des supports ;
- la sparsité du modèle, c'est-à-dire le rapport entre la densité des observations dans les supports $S_i, i = 1, \ldots, M$, et celle des outliers dans S_0.

L'approche proposée est également comparée avec des méthodes classiques de clustering (k-means, clustering spectral) sur différents scénarios de simulation. La figure 1.5 présente les résultats de 4 approches sur les données de la figure 1.4. Sur cet exemple, on remarque clairement que le single linkage classique identifie deux clusters de très petites tailles, les autres observations sont mises dans un cluster unique. Les autres approches se comportent mieux avec une préférence pour le clustering spectral et l'approche que nous proposons qui permet en plus d'identifier certains outliers.

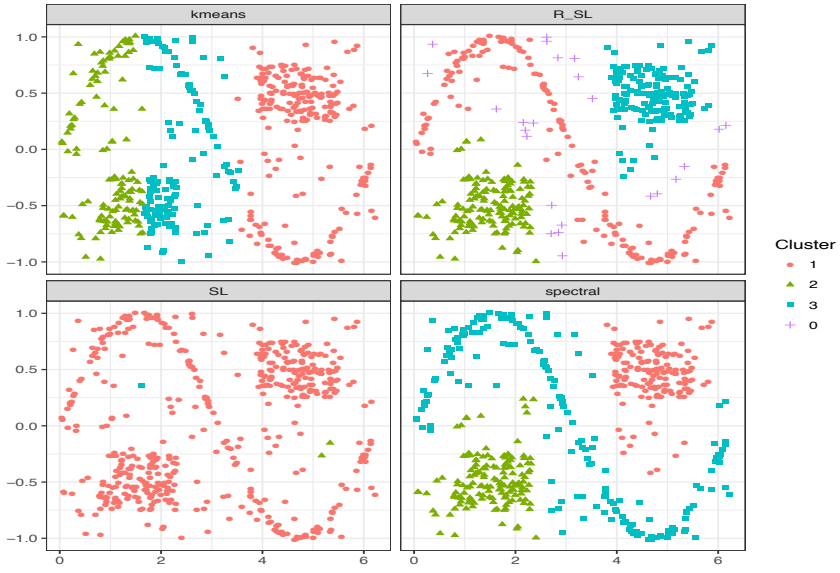


Figure 1.5: Résultats des algorithmes k-means, robust single linkage, single linkage classique et clustering spectral sur les données de la figure 1.4 (avec outliers).

Les résultats de ce chapitre font l'objet d'un article prochainement soumis dans la revue *Journal of Multivariate Analysis*. De plus, l'algorithme *Robust single linkage clustering* a été implémenté en langage `R`. Ce travail est le fruit d'une collaboration avec Nicolas Klutchnikoff (maître de conférences à l'Université Rennes 2) et Laurent Rouvière.

Contributions du chapitre :

- développement d'un algorithme de clustering permettant de prendre en compte des outliers,
- proposition d'un nouveau critère permettant de mesurer la performance d'une méthode de clustering et étude de ce critère pour l'algorithme proposé.

Chapter 2

Classification tree algorithm for grouped variables

Abstract. We consider the problem of predicting a categorical variable based on groups of inputs. Some methods have already been proposed to elaborate classification rules based on groups of variables (e.g. group lasso for logistic regression). However, to our knowledge, no tree-based approach has been proposed to tackle this issue. Here, we propose the Tree Penalized Linear Discriminant Analysis algorithm (TPLDA), a new-tree based approach which constructs a classification rule based on groups of variables. It consists in splitting a node by repeatedly selecting a group and then applying a regularized linear discriminant analysis based on this group. This process is repeated until some stopping criterion is satisfied. A pruning strategy is proposed to select an optimal tree. Compared to the existing multivariate classification tree methods, the proposed method is computationally less demanding and the resulting trees are more easily interpretable. Furthermore, TPLDA automatically provides a measure of importance for each group of variables. This score allows to rank groups of variables with respect to their ability to predict the response and can also be used to perform group variable selection. The good performances of the proposed algorithm and its interest in terms of prediction accuracy, interpretation and group variable selection are loud and compared to alternative reference methods through simulations and applications on real datasets.

Contents

2.1 Introduction **25**
2.2 The Penalized Tree Group algorithm **27**
2.2.1 Some notations 28
2.2.2 Construction of a maximal tree 29
2.2.3 Pruning strategy 31
2.2.4 A toy example 33
2.2.5 Group importance measure 34
2.3 Evaluation of the method by simulation studies **35**

2.3.1 Simulation design 35
2.3.2 Performances of TPLDA, CART and GL 37
2.3.3 Assessment of the group importance measure 40
2.3.4 Choice of the penalty function 40
2.4 Application to tumor classification using gene expression data 43
2.4.1 Data preprocessing and genes clustering 44
2.4.2 Evaluation of the methods 44
2.4.3 Results 45
2.5 Conclusion 47
2.6 Appendices 48
2.6.1 Time complexity of TPLDA 48
2.6.2 Additional figures about the illustration of the TPLDA method on a simple example 50
2.6.3 Additional information about the numerical experiments 50
2.6.4 Additional information about the application to gene expression data 56

2.1 Introduction

Consider the supervised classification setting where the problem consists in predicting a class variable Y taking values in $\{1, \ldots, K\}$, with $K \geq 2$, based on a vector $\mathbf{X}$ which takes values in $\mathbb{R}^d$. Suppose further that the inputs are divided into J different groups. In many supervised classification problems, inputs can have a group structure or groups of inputs can be defined to capture the underlying input associations. In these cases, the study of groups of variables can make more sense than the study of inputs taken individually. For example, in the analysis of gene expression data, datasets contain the expression levels of thousands genes in a much smaller number of observations. Then it has become frequent to use in the analysis only a small number of genes which can be clustered into several groups that represent putative biological processes (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 than individual wave lengths (Picheny et al., 2016). Finally, categorical inputs can be converted into a group of dummy variables that can be treated as a group. In all these situations, elaborating a classification rule based on groups of inputs rather than on the individual variables 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 input variables (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 (Genuer & Poggi, 2018; Loh, 2014). Classification tree algorithms elaborate classification rules by means of recursive partitioning of the data space. 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 applied on each resulting node until some stopping criteria are achieved or as long as the node is not pure (i.e. all observations in the node do not have the same label). 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: at each step, the algorithm splits the data space into more and more pure nodes. The terminal nodes, which are not split, are called leaves. At the end of the splitting process, the leaves define a partition of the data space which can be represented as a tree. A classification rule is associated to each leaf. In a leaf, observations are assigned to the most-represented class label in the leaf. Generally, the tree resulting from the splitting process is often not optimal with respect to a given criterion. So, a pruning method is often used to select an optimal tree (Breiman et al., 1984).

The first comprehensive study about classification tree algorithms was presented by Breiman et al. (1984), who introduced the popular CART algorithm. Since then, other classification tree algorithms have been developed, such as ID3 (Quinlan, 1986) and C4.5 (Quinlan, 1993). All these algorithms are univariate classification tree algorithms, that is, each node is determined according to the value of one single input. Multivariate classification trees algorithms that split each node according to the value of a subset of input variables, have also been studied. For most of the multivariate classification algorithms, splits are defined according to the value of a linear combination of a subset of input variables (Breiman et al. 1984, Wickramarachchi et al. 2016, Murthy et al. 1993, Wei-Yin Loh 1988, Li et al. 2003). Multivariate classification tree algorithms generally have higher accuracy and lead to smaller trees than univariate classification tree algorithms (Brodley & Utgoff, 1995; Lim et al., 2000). However, they suffer from two major drawbacks. First of all, they are generally time-consuming (Breiman et al., 1984; Li et al., 2003). Secondly, the subset of input variables used to define a split is automatically selected by the algorithm with respect to an impurity criterion and without regarding if the combination of this subset of selected variables make sense. Consequently, some splits may not make sense. Thus, multivariate classification trees are often difficult to interpret.

As mentioned previously, in many supervised classification problems, input variables can have a known group structure. In this context, as far as we know, no multivariate classification tree algorithm enables to take account of this group structure. This led us to develop the Penalized Tree Linear Discriminant Analysis algorithm (TPLDA), 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 variables by performing the regularized linear discriminant analysis proposed by Witten & Tibshirani (2011). Next, the algorithm selects the optimal split with respect to an impurity criterion. This splitting procedure is then repeated until predetermined 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. The algorithm does not need to perform a greedy search to determine the subsets of input variables used to define the optimal splits since it uses the existing group structure. Moreover, interpretation is easy because the algorithm uses the group structure which makes sense. Furthermore, as identification of relevant groups of inputs is also an important issue in many classification problems involving groups of variables, we introduce a measure of group importance. This score is based on a TPLDA tree and allows to rank all the groups of inputs according to their discriminatory power.

To simplify matters, in this paper, we restrict our attention to binary classification problems, which already captures many of the main features of more general problems. Nonetheless, our algorithm 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 2.2 describes the TPLDA algorithm and the group importance measure. In Section 2.3, performances of the proposed algorithm are analyzed through a detailed simulation study. TPLDA is compared to CART and GL, which is one of the reference methods to elaborate classification rules with groups of inputs. In Section 2.4, TPLDA is applied on three publicly available real microarray datasets. The proposed method is then compared to CART, GL and the shrunken centroid regularized discriminant analysis (SCRDA) (Guo et al., 2006) which is one of the standard methods used to analyze microarray data. The time complexity of TPLDA and additional information about the simulation study and the application on the three microarray datasets are provided in Section 2.6.1. The method has been implemented in `R` language. The functions are available at https://github.com/apoterie/TPLDA.

2.2 The Penalized Tree Group algorithm

Let $(\mathbf{X}, Y)$ be a random vector taking values in $\mathcal{X} \times \{0,1\}$, where $\mathbf{X} = (X_1, \ldots, X_d)$ is a vector of input variables with $\mathcal{X} = \mathbb{R}^d$ and Y is the class label. Let $\{(\mathbf{X}_1, Y_1), \ldots, (\mathbf{X}_{n+m}, Y_{n+m})\}$ be independent copies of $(\mathbf{X}, Y)$, which are randomly split into a training set $\mathcal{D}_n = \{(\mathbf{X}_1, Y_1), \ldots, (\mathbf{X}_n, Y_n)\}$ of size n and a validation set $\mathcal{T}_q = \{(\mathbf{X}_{n+1}, Y_{n+1}), \ldots, (\mathbf{X}_{n+m}, Y_{n+m})\}$ of size m. A discrimination rule is a measurable function $\hat{g} : \mathbb{R}^d \times (\mathbb{R}^d \times \{0,1\})^{n+m} \to \{0,1\}$ which classifies a new observation $\mathbf{x} \in \mathbb{R}^d$ into the class $\hat{g}(\mathbf{x}, (\mathbf{X}_1, Y_1), \ldots,$

$(\mathbf{X}_{n+m}, Y_{n+m})$). In what follows, we will write $\hat{g}(\mathbf{x})$ for the sake of convenience.

In this work, we consider the situation where $\mathbf{X}$ is structured into J known groups. For any $j = 1, \ldots, J$, let $\mathbf{X}^j$ denote the j-th group of size d_j, such that:

$$\mathbf{X}^j = (X_{j_1}, X_{j_2}, \ldots, X_{j_{d_j}}).$$

To simplify matters, the J groups are ordered such that

$$\mathbf{X} = (\mathbf{X}^1, \ldots, \mathbf{X}^J).$$

Note that the groups are not necessarily disjoint, some input variables can belong to several groups. The objective is to construct a classification rule $\hat{g}$ which takes into account the group structure. To do this we propose a new tree-based approach named the Tree Penalized Discriminant Analysis (TPLDA). This method elaborates a classification rule based on two steps. First, the algorithm builds a maximal classification tree which is next pruned. These two steps are described below. We need to introduce some notations before describing the TPLDA algorithm.

2.2.1 Some notations

If T is a tree, t is the general notation for a node of T and n_t is the total number of observations in t. Let k be the class label, $k \in \{0, 1\}$. We denote by $R_{k,t}$ the set of observations with the label k in the node t and $|R_{k,t}| = n_{k,t}$, such that $n_{0,t} + n_{1,t} = n_t$. The class probability in the node t is estimated by its standard empirical estimate $\pi_{k,t} = \frac{n_{k,t}}{n_t}$.

For any $j = 1, \ldots, J$, let consider the group $\mathbf{X}^j$ of inputs. In t, the standard estimate of the between-class covariance matrix B_t^j of group $\mathbf{X}^j$ is given by

$$\widehat{B}_t^j = \frac{1}{n_t - 2} \sum_{k=0}^{1} n_{k,t} (\hat{\mu}_t^j - \hat{\mu}_{k,t}^j)(\hat{\mu}_t^j - \hat{\mu}_{k,t}^j)^\top, \tag{2.1}$$

where $\top$ stands for the transpose vector and $\hat{\mu}_{k,t}^j$ is the empirical estimate of the class mean vector of $\mathbf{X}^j$ in the node t. Furthermore, the within-class covariance matrix Σ_t^j of $\mathbf{X}^j$ is estimated by its diagonal positive estimate $\widehat{\Sigma}_t^j$ defined as

$$\widehat{\Sigma}_t^j = \operatorname{diag}\left((\hat{\sigma}_{t,1}^j)^2, \ldots, (\hat{\sigma}_{t,d_j}^j)^2\right), \tag{2.2}$$

where $\hat{\sigma}_{t,\ell}^j$, with $\ell = 1, \ldots, d_j$, denotes the within-class standard deviation estimate of the ℓ-th input of $\mathbf{X}^j$.

2.2.2 Construction of a maximal tree

As for existing tree-based methods, TPLDA elaborates a maximal tree by recursively partitioning the data space. At each step, the data space is divided into smaller and smaller nodes. This splitting process, that is applied on nodes, is made of two steps. Consider the split of the node t. First, for any $j = 1, \ldots, J$, we split the input space according to a linear combination of the inputs belonging to group $\mathbf{X}^j$. Then, we select the best split with respect to an impurity criterion (which is equivalent to selecting the splitting group). These steps are now described in greater details.

- Step 1: within group PLDA.
  In the first step, the algorithm performs a penalized linear discriminant analysis (PLDA, Witten & Tibshirani, 2011) on each group $\mathbf{X}^j = (X_1^j, \ldots, X_{d_j}^j)$, with $j = 1, \ldots, J$. That is, PLDA seeks a one-dimensional projection $(\beta^j)^\top \mathbf{x}^j$, $(\beta^j = (\beta_1^j, \ldots, \beta_{d_j}^j) \in \mathbb{R}^{d_j})$, of the observations in t, that maximizes the ratio of the between-class covariance to the within-class covariance. PLDA's criterion can be defined as:

$$\max_{\beta^j \in \mathbb{R}^{d_j}} \left\{ (\beta^j)^\top \widehat{B}_t^j \, \beta^j - \lambda_j \sum_{\ell=1}^{d_j} |\widehat{\sigma}_{t,\ell}^j \beta_\ell^j| \right\} \quad \text{subject to} \quad (\beta^j)^\top \widehat{\Sigma}_t^j \, \beta^j \leq 1, \tag{2.3}$$

  where $\widehat{B}_t^j$ and $\widehat{\Sigma}_t^j$ are respectively given by (2.1) and (2.2). As for Fisher's linear discriminant analysis (Hastie et al., 2009, FDA), the solution of (2.3) is denoted by $\hat{\beta}^j$ and is called the penalized discriminant vector. In (2.3), the parameter $\lambda_j \in \mathbb{R}^+$ is a regularization parameter that can force some components of β^j to be set to zero. The use of the regularization parameter λ_j and the diagonal positive within-class covariance matrix Σ_t^j enables to solve the singularity problem occurring when the number of observations in the node t is small compared to the number of variables in the group $\mathbf{X}^j$ (for more details see Witten & Tibshirani, 2011).

  PLDA divides the node t into two child nodes according to the linear decision boundary described by the linear equation $\beta^{j\top}(\mathbf{x}^j - \frac{(\hat{\mu}_{1,t}^j - \hat{\mu}_{0,t}^j)}{2}) = 0$. The two child nodes of t are defined as:

$$\begin{aligned} t_0(j) &= \left\{ \mathbf{x} \in t \mid \hat{\beta}^{j\top} \left(\mathbf{x}^j - \frac{(\hat{\mu}_{1,t}^j - \hat{\mu}_{0,t}^j)}{2} \right) < 0 \right\} \\ &\text{and} \\ t_1(j) &= \left\{ \mathbf{x} \in t \mid \hat{\beta}^{j\top} \left(\mathbf{x}^j - \frac{(\hat{\mu}_{1,t}^j - \hat{\mu}_{0,t}^j)}{2} \right) \geq 0 \right\}. \end{aligned} \tag{2.4}$$

  In (2.3), if λ_j is equal to zero and if either the inputs in group $\mathbf{X}^j$ are mutually independent or the size d_j of the j-th group is 1, then the matrix $\widehat{\Sigma}_t^j$ is reduced to

the standard estimate of the within-class covariance matrix . In this case, the PLDA problem (2.3) is equivalent to the FDA problem.
Note that FDA cannot be used here since it is not adapted to the recursive splitting of nodes that become smaller and smaller (Shao et al., 2011; Friedman, 1989; Xu et al., 2009; Bouveyron et al., 2007). This point is discussed in Section 2.6.3.

- Step 2: choosing the splitting group.
  Selection of the splitting group is based on Gini impurity function, which is estimated on the training set by

$$\mathrm{Q}(t) = \pi_{1,t}(1 - \pi_{1,t}).$$

  The algorithm selects the splitting group $j_t^\star \in \{1, \ldots, J\}$, which maximizes the impurity decrease defined for each group $\mathbf{X}^j$, $j = 1, \ldots, J$, by

$$\Delta\mathrm{Q}(j,t) = \left[n_t \mathrm{Q}(t) - n_{t_0(j)}\mathrm{Q}(t_0(j)) - n_{t_1(j)}\mathrm{Q}(t_1(j))\right]. \tag{2.5}$$

  In practice, criterion (2.5) may not be satisfying since it tends to foster larger groups. Indeed, the largest groups have more possible splits than smallest groups. As a consequence, it is more likely that the largest groups will be optimal with respect to the decrease in node impurity (Strobl et al., 2007). Thus, to control this selection bias, we propose to penalize the criterion (2.5) by a decreasing function $\mathsf{pen}(d_j)$ of the group size d_j:

$$\Delta_p\mathrm{Q}(j,t) = \mathsf{pen}(d_j)\mathrm{Q}(j,t). \tag{2.6}$$

  Several penalty functions can be used. We propose:

$$\begin{aligned} \mathsf{pen}(d_j) &= 1/d_j, \\ \mathsf{pen}(d_j) &= 1/\sqrt{d_j}, \\ \mathsf{pen}(d_j) &= 1/\max(\log d_j, 1). \end{aligned} \tag{2.7}$$

  The use of the corrected impurity criterion (2.6) and the choice of the penalty function are discussed in Section 2.3.

Remark 2.2.1.

- In step 1, the value of the tuning parameter λ_j is chosen by K-fold cross-validation. The algorithm selects among L guided values the value of λ_j which maximizes the cross-validated estimate of the decrease in impurity (2.5).
- The impurity function Q measures the homogeneity of a node. Here, we use Gini impurity function. However, other impurity criteria, such as the information criterion, could be used.

- The time complexity of TPLDA at a node t of size n_t is in the worst case $\mathcal{O}\left(JLKn_t d_{\max}^2\right)$ with J referring to the number of groups, K being the number of folds in the cross-validation used to tune λ_j, L denoting the number of guided values for λ^j in the cross-validation and $d_{\max} = \max_j(d_j)$ with d_j being the size of $\mathbf{X}^j$. The computation is detailed in Section 2.6.1. As the inequality $K \leq n_t$ is always satisfied, TPLDA remains less time consuming than lots of multivariate classification tree algorithms such as HHCART (time complexity $= \mathcal{O}\left(n_t^2 d^3\right)$ with $d = \sum_{j=1}^{J} d_j$ is the total number of input) and OC1 (time complexity $= \mathcal{O}\left(n_t^2 \log(n_t) d\right)$), excepted in very small nodes (i.e. $Ld_{\max} > \log(n_t)$). A detailed calculation of the time complexity of HHCART and OC1 is provided by Wickramarachchi et al. (2016).

At the very beginning of the whole procedure, steps 1 and 2 are applied to partition the entire data space into two nodes. Then, these steps are repeated recursively on each node t until each one satisfies at least one of the following stopping criteria:

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

$$\pi_{1,t} < \epsilon \quad \text{or} \quad \pi_{1,t} > 1 - \epsilon,$$

  for a small given value ϵ,

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

$$\Delta_p \mathrm{Q}(j, t) = 0, \text{ for any } j = 1, \ldots, J.$$

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

2.2.3 Pruning strategy

Let T be a subtree of $T_{\max}$ and $\widetilde{T}$ be the set of $|\widetilde{T}|$ terminal nodes of T. We define the depth of node t, which is denoted by $D(t)$, as the number of conditions that an observation $\mathbf{x} \in \mathbb{R}^d$ has to satisfy from the root to the node t. The depth $D(T)$ of the tree T is then defined as:

$$D(T) = \max_{t \in \widetilde{T}} D(t).$$

Figure 2.1 illustrates the notions of nodes, terminal nodes and depth.

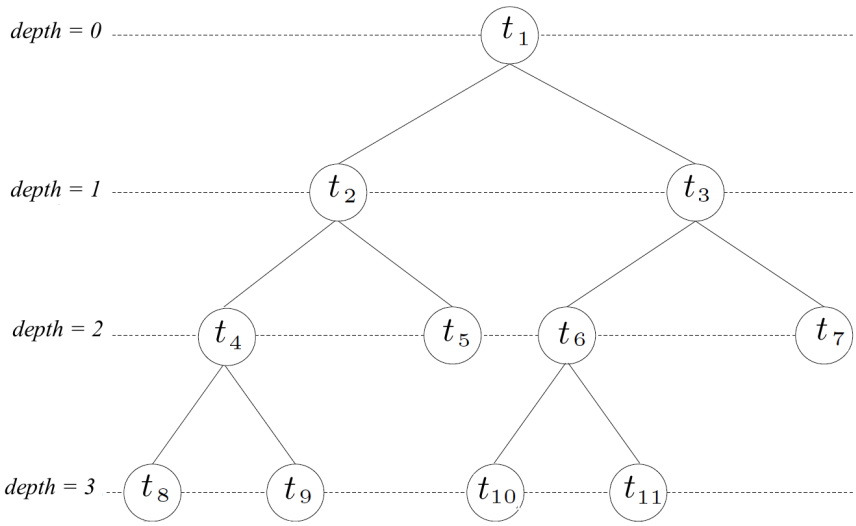


Figure 2.1: Example of a classification tree. Circles indicate the nodes. *depth* refers to the depth of the nodes. Here $D(T) = 3$. The terminal nodes are $\widetilde{T} = \{t_5, t_7, t_8, t_9, t_{10}, t_{11}\}$. The node t_1 denotes the tree root.

Define the sequence

$$t_1 = T_0 \subset T_1 \subset \ldots \subset T_{D(T_{\max})} = T_{\max} \tag{2.8}$$

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

$$\sum_{t \in \widetilde{T}} D(t) \quad \text{subject to} \quad D(t) \leq k.$$

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 2.1 gives the terminal nodes for the sequence of subtrees of the tree displayed in Figure 2.1.

Tree	**Terminal Nodes**
T_0	t_0
T_1	t_2, t_3
T_2	t_4, t_5, t_6, t_7
T_3	$t_8, t_9, t_5, t_{10}, t_{11}, t_7$

Table 2.1: Terminal nodes for the subtrees in Figure 2.1.

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

$$\hat{g}_k(\mathbf{x}) = \sum_{t \in \widetilde{T}_k} \hat{y}_t \mathbb{1}_t(\mathbf{x}), \quad \mathbf{x} \in \mathbb{R}^d, \tag{2.9}$$

where $\mathbb{1}_t(\mathbf{x})$ is the indicator function which equals 1 if $\mathbf{x}$ falls into the leaf t and 0 otherwise, and $\hat{y}_t = \mathbb{1}_{n_{1,t} \geq n_{0,t}}$ is the most represented class in the node t. Note that the classification rules $\hat{g}_k$, $k = 1, \ldots, D(T_{\max})$, depend only on the training set $(\mathbf{X}_1, Y_1), \ldots, (\mathbf{X}_n, Y_n)$. The proposed pruning strategy selects the rule $\hat{g}_{\widehat{k}}$ which minimizes the misclassification error $\mathbb{P}(\hat{g}_k(\mathbf{X}) \neq Y)$. In practice, this error is estimated on the validation set $\mathcal{T}_q$. Precisely, we choose

$$\widehat{k} \;=\; \underset{k=1,\ldots,D_{T_{\max}}}{\operatorname{argmin}} \frac{1}{m} \sum_{i:(\mathbf{X}_i,Y_i)\in\mathcal{T}_q} \mathbb{1}_{\hat{g}_k(\mathbf{X}_i)\neq Y_i}.$$

The final tree retained by our procedure is the subtree $T_{\widehat{k}}$.

The cardinality of the sequence $\{\hat{g}_1, \ldots, \hat{g}_{D(T_{\max})}\}$ of classifiers is finite and bounded by the size n of the training set $\mathcal{D}_n$. Therefore, using classical empirical minimization tools (see Devroye et al., 1996, chapter 26), we obtain that the selected rule $\hat{g}_{\hat{k}}$ satisfies the following inequality:

$$\mathbb{E}\left[\,|\mathbb{P}(\hat{g}_{\widehat{k}}(\mathbf{X}) \neq Y) \;-\; \inf_{k\in\{1,\ldots,D(T_{\max})\}} \mathbb{P}(\hat{g}_k(\mathbf{X}) \neq Y)|\,\right] \leq 2\sqrt{\frac{\log(2n)+1}{2m}}, \tag{2.10}$$

where m is the size of the validation set $\mathcal{T}_q$. Thus, since in most cases of interest $\log(m) \ll n$, inequality (2.10) means that the selected classification rule $\hat{g}_{\widehat{k}}$ classifies as well as the best classifier in the sequence $\{\hat{g}_1, \ldots, \hat{g}_{D(T_{\max})}\}$ (with respect to the misclassification error).

The following section illustrates the TPLDA algorithm.

2.2.4 A toy example

Consider the random vector $(\mathbf{X}, Y)$ with values in $\mathbb{R}^2 \times \{0, 1\}$. X_1 and X_2 are two independent random variables with distribution $\mathcal{N}(0, 1)$. The conditional distribution of Y is defined as

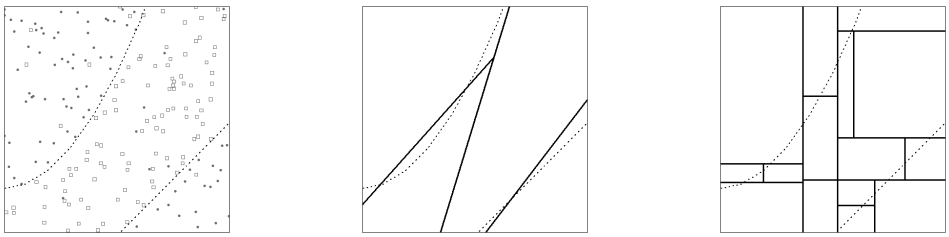
$$\mathcal{L}(Y \mid \mathbf{X} = \mathbf{x}) = \begin{cases} \mathcal{B}(0.9) & \text{if} \quad x_2 > 2x_1^2 + 0.20 \quad \text{or} \quad x_2 < 0.5 + x_1 \\ \mathcal{B}(0.1) & \text{otherwise.} \end{cases} \tag{2.11}$$

where $\mathcal{B}(\pi)$ denotes a Bernoulli distribution of parameter π. 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 $g^*(\mathbf{x})$ is defined by:

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

For TPLDA and CART, a maximal tree is first built on a training sample of 50 observations and is next pruned by using a validation set of 50 observations. TPLDA uses the proposed pruning strategy described above while CART uses the classical minimal cost-complexity pruning method (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, TPLDA allows to elaborate a less complex partition of the input space without lost of accuracy (Figure 2.2). The associated trees are displayed in Section 2.6.2.



(a) (b) (c)

Figure 2.2: Illustration of the TPLDA method - a simple binary classification problem in $\mathbb{R}^2$. (a) 200 observations defined by model (2.11), (b) a TPLDA partition (AUC=0.90), (c) a CART partition (AUC=0.89). On each graph, Bayes decision boundaries are represented by the two dotted lines.

2.2.5 Group importance measure

In supervised classification problems involving grouped inputs, groups are seldom equally relevant. Often only a few of them are important with respect to the prediction of the response variable. The quantification of the group importance is then useful for both interpretation and performing group variable selection. TPLDA provides a measure of importance of each group. This score, which is related to a TPLDA tree, is based on the penalized splitting criterion (2.6). Formally, the importance of the group $\mathbf{X}^j$, $j = 1, \ldots, J$, related to a TPLDA tree T, is the sum over all non-terminal nodes of T of the *corrected* decrease in node impurity from splitting on group j,

$$\mathcal{I}_{\mathsf{tplda}}(j, T) \;=\; \sum_{t \,\in\, T \setminus \widetilde{T}} \Delta_p \mathrm{Q}(j, t) p(j, j_t^\star), \tag{2.12}$$

where $\Delta_p \mathrm{Q}(j, t)$ is the penalized decrease in impurity (2.6) from splitting t on group j, $j_t^\star$ is the index of the group selected to split the node t (see Step 2 in Section 2.2.2) and $p(j_t^\star, j)$ is a *correction*. The parameter $p(j, j_t^\star)$ is the empirical probability of agreement between the split of the node t based on j and the one based on $j_t^\star$. It is defined by

$$p(j, j_t^\star) \;=\; \max\left\{ p_{00}(j, j_t^\star) + p_{11}(j, j_t^\star)\,,\, p_{01}(j, j_t^\star) + p_{10}(j, j_t^\star) \right\},$$

where $p_{kk'}(j, j_t^\star)$, with $(k, k') \in \{0, 1\}^2$, is the empirical probability that the split of node t based on group j and the one based on group $j_t^\star$ send an observation in node t both into $t_k(j)$ and $t_{k'}(j_t^\star)$. $p(j, j_t^\star)$ lies between 0 and 1 and takes the value 1 if the two splits send all observations in node t into the same child node. This quantity is used to prevent

overestimating the importance of groups which are weakly correlated with both the relevant groups and the response variable (see chap. 5, Breiman et al., 1984).

As only the relative magnitude of this score matters, the measure of group importance is normalized to a scale between 0 and 100,

$$\widetilde{\mathcal{I}}_{\mathsf{tplda}}(j,T) = 100 \times \frac{\mathcal{I}_{\mathsf{tplda}}(j,T)}{\max\limits_{j'=1,\ldots,J} \mathcal{I}_{\mathsf{tplda}}(j',T)}. \tag{2.13}$$

This score induces an order of importance. Groups with the highest values for $\widetilde{\mathcal{I}}_{\mathsf{tplda}}$ are considered as important. The group importance measure is assessed in the simulation studies introduced in Section 2.3.

2.3 Evaluation of the method by simulation studies

Several numerical experiments based on Hastie et al. (2009) are used to assess the performances of TPLDA. In this simulation study, the proposed method is compared to CART since the two methods are very similar when inputs are not grouped. TPLDA is also compared to GL, which is one of the reference methods to elaborate classification rules with groups of inputs. The general simulation design is described below.

2.3.1 Simulation design

The outcome variable Y is simulated from a Bernoulli distribution $Y \sim \mathcal{B}(0.5)$. The vector $\mathbf{X}$ of inputs is structured into J groups: $\mathbf{X} = (\mathbf{X}^1, \ldots, \mathbf{X}^J)$. Each group $\mathbf{X}^j$, $j = 1, \ldots, J$, includes d_j variables. When $Y = 0$, for any $j = 1, \ldots, J$ and any $\ell = 1, \ldots, d_j$ the component X_ℓ^j follows a standard Gaussian distribution:

$$\mathcal{L}(X_\ell^j \mid Y = 0) \;=\; \mathcal{N}(0,1).$$

When $Y = 1$, for any $j = 1, \ldots, J$ and any $\ell = 1, \ldots, d_j$ the component X_ℓ^j is defined conditionally to the value of the standard uniform random variable U:

$$\mathcal{L}(X_\ell^j \mid Y = 1, U = u) = \begin{cases} \mathcal{N}(-\mu_j, 1) & \text{if } u < u_1; \\ \mathcal{N}(\mu_j, 1) & \text{if } u_1 \leq u < u_2; \\ \mathcal{N}(0,1) & \text{otherwise.} \end{cases} \tag{2.14}$$

where u_1, u_2 are two fixed real numbers satisfying $0 \leq u_1 < u_2 \leq 1$. For any $j = 1, \ldots, J$, the component $\mu_j \geq 0$ 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. If $\mu_j = 0$, all inputs in group $\mathbf{X}^j$ are distributed according to a standard Gaussian distribution, whatever

the values of Y and U. In this case, the group $\mathbf{X}^j$ is not relevant to predict Y. We note $\mu = (\mu_1, \ldots, \mu_J)$.

The covariance between two inputs X_ℓ^j and $X_{\ell'}^{j'}$ ($j, j' = 1, \ldots, J$ and $\ell = 1, \ldots, d_j$ and $\ell' = 1, \ldots, p_{j'}$) is defined as:

$$\mathrm{Cov}\left(X_\ell^j, X_{\ell'}^{j'}\right) = \begin{cases} c_w^{|\ell-\ell'|} & \text{if } j = j', \\ 0 & \text{otherwise.} \end{cases}$$

where $0 \leq c_w < 1$ and $|\ell - \ell'|$ denotes the distance between two inputs belonging to the same group. Thus, in this simulation design, the group structure of the inputs comes from both the discriminatory power of the inputs defined by the vector μ and the block structure of the covariance matrix of $\mathbf{X}$. The covariance structure mimics the one of gene expression data: genes included in a same putative biological pathway are correlated and the correlation is a decreasing function of the "distance" between two genes.

Finally, $n + m + q$ observations are generated according to this simulation model and 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.

To assess the performances of TPLDA, five experiments are considered by varying the parameters n, m and d_j, $j = 1, \ldots, J$. In every experiment, the size of the test set is $q = 1000$ and $J = 10$ groups are simulated. 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. We choose $(u_1, u_2) = (0.25, 0.90)$ and $c_w = 0.85$.

The five considered scenarios are described below:

- Experiment 1: ungrouped data.
  Each group includes $d_j = 1$ variable and the training and the validation samples both include $n = m = 500$ observations.

- Experiment 2: groups of equal size.
  Each group includes $d_j = 10$ variable and the training and the validation samples both include $n = m = 500$ observations.

- Experiment 3: large groups of equal size.
  Each group includes $d_j = 50$ variables and the training and the validation samples both include $n = m = 100$ observations.

- Experiment 4: a large noisy group.
  This experiment is similar to experiment 2 with the addition of a large noisy group including realizations of 50 independent standard Gaussian variables.

- Experiment 5: a large noisy group and some noisy variables in the most relevant group. This experiment is similar to experiment 4 with the addition of 10 independent standard Gaussian variables in the first group of variables (i.e. in the most relevant group of variables).

The first three experiments are used to assess the performances of the TPLDA method in comparison with CART and GL and to evaluate the group importance measure. The last two experiments are used to study the use of the penalized Gini criterion (2.6) when choosing the splitting group. The three penalty functions defined in equation (2.7) are assessed. In the first three experiments, the Gini criterion is not penalized, that is $\mathsf{pen}(d_j) = 1$.

For TPLDA, the maximal tree is built on the training set and is next pruned by applying the pruning strategy described in Section 2.2.3 on the validation sample. In CART, the training set is used to elaborate the maximal tree that is next pruned by using the minimal cost-complexity pruning method and the validation set. For GL, the model is elaborated on the training set and the shrinkage parameter is selected on the validation set.

A variant of TPLDA is also applied on the five experiments. In this variant, PLDA is replaced by FDA. Results, that are given in Section 2.6.3, illustrate the fact that FDA is not adapted to recursively split nodes that become smaller and smaller.

Moreover, in order to assess the sensitivity to the pruning method, the pruning strategy proposed in Section 2.2.3 is also used to prune the CART maximal tree. Results are given in Section 2.6.3 and show no significant difference between methods.

CART and GL results in experiments 4 and 5 are displayed in Section 2.6.3. Results of each experiment are averaged over 200 independent samples.

2.3.2 Performances of TPLDA, CART and GL

In each experiment, the predictive performances of TPLDA, CART and GL are assessed and compared by the AUC on the test set. Furthermore, the complexity of the classification rule is also studied. For TPLDA and CART, this criterion is measured by using the tree depth: interpretation of a large tree is harder than the one of a small tree. For GL, the complexity of the classification rule is measured by the number of groups included in the model: the complexity increases with the number of groups included in the model.

Table 2.2 displays the simulation results for each assessed method. For each criterion, the median value is given following by the values of the first and the third quartiles in brackets. The model size gives the number of groups of variables included in the final GL model. Figures 2.3 and 2.5 display group selection frequencies for TPLDA. The selection frequency of a given group is defined as the number of times that a group is included at least once

in the final model. Distribution of the AUC for each method in the three experiments are displayed in Section 2.6.3. Globally, TPLDA performs well in the three scenarios. Compared to CART and GL, it elaborates more accurate and easily understandable classification rules.

	TPLDA	CART	GL
Experiment 1			
AUC	0.66 (0.65,0.68)	0.67 (0.65,0.68)	0.64 (0.63,0.66)
Tree depth	4 (3,5)	5 (3,7)	.
Model size	.	.	4 (3,7)
Experiment 2			
AUC	0.76 (0.74,0.77)	0.68 (0.66,0.7)	0.66 (0.65,0.68)
Tree depth	3 (3,4)	6 (4,8)	.
Model size	.	.	4 (3,5)
Experiment 3			
AUC	0.83 (0.7,0.85)	0.64 (0.62,0.66)	0.67 (0.64,0.69)
Tree depth	2 (2,3)	4 (2,5)	.
Model size	.	.	2 (1,4)

Table 2.2: Performances of the assessed methods.

In experiment 1, we highlight the similarity between TPLDA and CART when inputs are not grouped. Indeed, TPLDA selects the same input variables and has similar predictive performances as CART (Figure 2.3). Nonetheless, CART elaborates larger trees and tends to select less frequently the noisy groups. The two methods do not exactly give the same results since they do not use the same splitting process. To split a node, CART tries to find the splitting input and the value for this input that maximizes the decrease in impurity in the node (see Breiman et al., 1984). On the contrary, TPLDA first estimates a split for every group based on a maximization of the ratio of the between-class covariance matrix and the within-group covariance matrix and next selects the split that maximizes the decrease in impurity in the node (see Section 2.2.2).

In the second and the third experiments, input variables are grouped. In these scenarios, TPLDA outperforms the other methods. In particular, it has higher predictive performances. Besides, the final TPLDA trees are smaller than the final CART trees. This last point can be explained by the use of multivariate splits which are more informative. This leads to a quicker decreasing of the misclassification error in both the training set and the validation set and then to smaller final trees (Figures 2.4). Furthermore, since TPLDA splits are defined according to the group which makes more sense that inputs taken individually, TPLDA trees are more easily interpretable than CART trees. Also, TPLDA well identifies the most relevant groups and the selection frequency of a given group behaves as an increasing function of the discriminatory power of the group (Figures 2.3 and 2.5).

The predictive performances of TPLDA and GL seem to improve with the group size. We can explain such behavior as follows. Since in each group all inputs share the same discriminatory power, the discriminatory power of a predictive group increases when the group size increases.

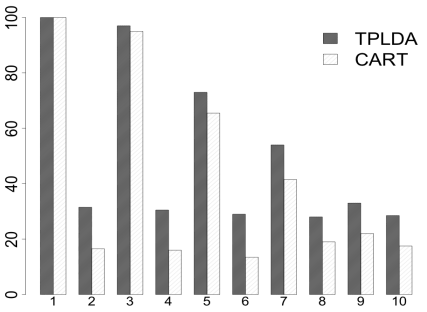


Figure 2.3: Group selection frequency in experiment 1 (in %).

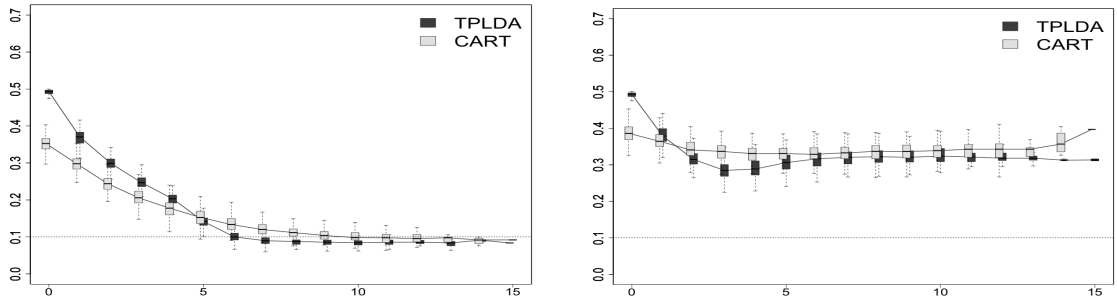


Figure 2.4: Misclassification error estimate according to the tree depth on the training set (left) and on the validation set (right) in experiment 2. The dotted lines denote the value of the Bayes error (Bayes error=10%).

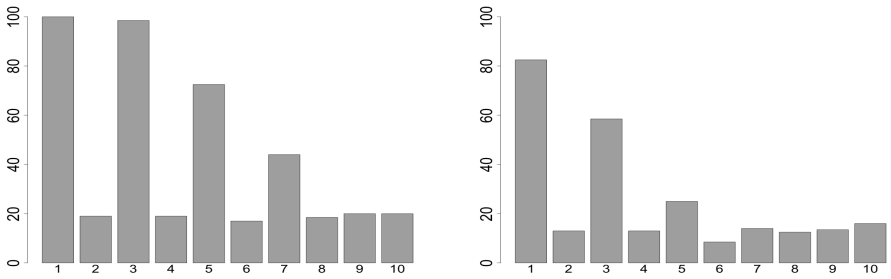


Figure 2.5: Group selection frequency (in %) for TPLDA in experiment 2 (left) and experiment 3 (right).

2.3.3 Assessment of the group importance measure

In this section, we study the performances of the group importance measure. Table 2.3 displays the percentage of time that the relevant groups are part of the 5 groups of inputs with the highest score of importance. The average selection frequency of the relevant groups with GL is also added, for comparison purpose.

In all experiments, the TPLDA group importance measure seems to well identify the three most relevant groups. The fourth and fifth most relevant groups are less frequently identified. This may be due to the relative low discriminatory power of these groups compared to the three other relevant groups. The distribution of the group importance measure for each group in each experiment is displayed in Section 2.6.3.

	TPLDA importance score	GL model
Experiment 1		
Selection rate of the 5 relevant groups	30	20.5
Selection rate of at least 3 relevant groups	100	78
Selection rate of the 3 most relevant groups	98.5	65.5
Experiment 2		
Selection rate of the 5 relevant groups	33.5	11.5
Selection rate of at least 3 relevant groups	100	79.5
Selection rate of the 3 most relevant groups	100	66.5
Experiment 3		
Selection rate of the 5 relevant groups	14	7.5
Selection rate of at least 3 relevant groups	90.5	35
Selection rate of the 3 most relevant groups	80.5	19

Table 2.3: Assessment of the group importance measure: top 5 groups with the highest score of importance.

2.3.4 Choice of the penalty function

This section investigates the use of a penalized Gini criterion for choosing the splitting group (2.5). Three penalty functions are evaluated (2.7). The performances of the TPLDA method when using these penalty functions are compared to the TPLDA method with no penalty (i.e. $\mathsf{pen}(d_j) = 1$).

Tables 2.4 and 2.5 summarize the simulation results. Boxplots of the group importance measure are displayed in 2.6.3. According to these two experiments, the use of a penalty function enables to control the sensitivity to the group size. Indeed, when no penalty function is used, the large noisy group is often chosen to build the tree whereas this group is significantly less

frequently selected when a penalty function is used (Figures 2.6 and 2.7). Consequently, in these scenarios, using a penalized Gini criterion allows to improve significantly the predictive performances of the classification rule and also the ability of the importance score to identify the true relevant group (Tables 2.4 and 2.5). Moreover, in these scenarios, the three penalty functions give similar results in terms of predictive performances and group selection frequency. None penalty function is preferable, they all seem adapted.

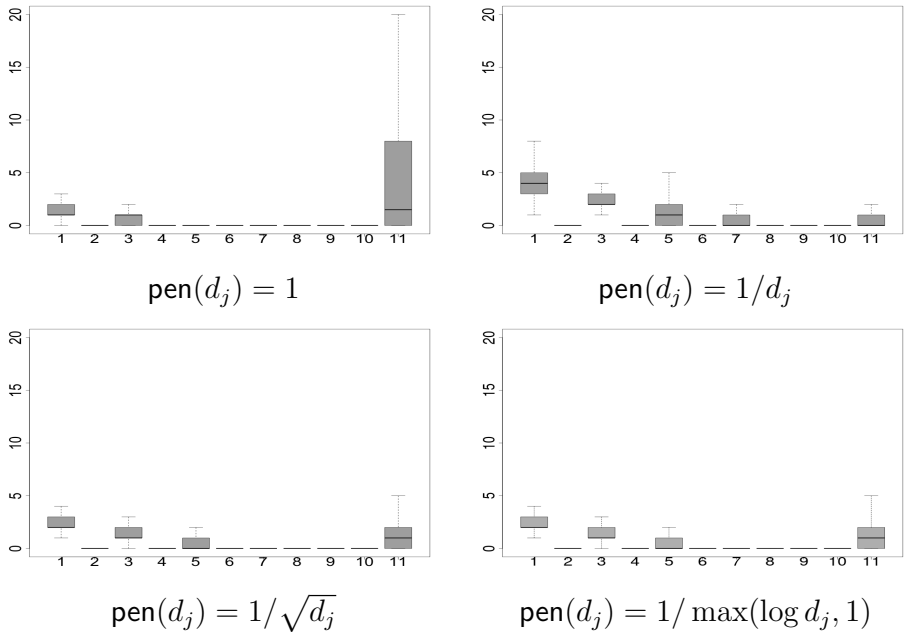
Generally, the choice of the penalty function is highly dependent on the data. Indeed, if it is expected that the noise is mostly included in the largest groups which are much larger than the supposed relevant groups, then the penalty $\mathsf{pen}(d_j) = 1/d_j$ would be preferable. Otherwise, this penalty function may appear too strong and other penalties such as $\mathsf{pen}(d_j) = 1/\sqrt{d_j}$ or $\mathsf{pen}(d_j) = 1/\max(\log d_j, 1)$ may perform better. Note that if all groups have the same size, such as in the first three scenarios, there is no need to use a penalty function.

Penalty function	1	$1/d_j$	$1/\sqrt{d_j}$	$1/\max(\log d_j, 1)$
Experiment 4				
AUC	0.65 (0.6,0.7)	0.75 (0.74,0.77)	0.73 (0.71,0.75)	0.72 (0.68,0.74)
Tree depth	2 (2,4)	3 (3,4)	3 (2,3)	2 (2,3)
Experiment 5				
AUC	0.66 (0.6,0.71)	0.73 (0.7,0.75)	0.72 (0.68,0.74)	0.71 (0.65,0.73)
Tree depth	2 (2,3)	3 (3,4)	3 (2,3)	2 (2,3)

Table 2.4: Sensitivity to the choice of the penalty function pen: performances of TPLDA according to the penalty function.

Penalty function	1	$1/d_j$	$1/\sqrt{d_j}$	$1/\max(\log d_j, 1)$
Experiment 4				
Selection rate of the 5 relevant groups	0	2	0.5	0
Selection rate of at least 3 relevant groups	99.5	100	100	100
Selection rate of the 3 most relevant groups	95.5	100	99.5	100
Median ranking of the 1st group	2 (1,2)	1 (1,1)	1 (1,2)	1 (1,2)
Median ranking of the 11th group	1 (1,3)	5 (4,5)	3 (3,4)	3 (2,4)
Experiment 5				
Selection rate of the 5 relevant groups	0	3.5	0	0
Selection rate of at least 3 relevant groups	99.5	100	100	100
Selection rate of the 3 most relevant groups	98	99.5	100	98.5
Median ranking of the 1st group	2 (1,2)	2 (1,2)	1 (1,2)	1 (1,2)
Median ranking of the 11th group	2 (1,3)	5 (4,5)	3 (3,4)	2 (3,4)

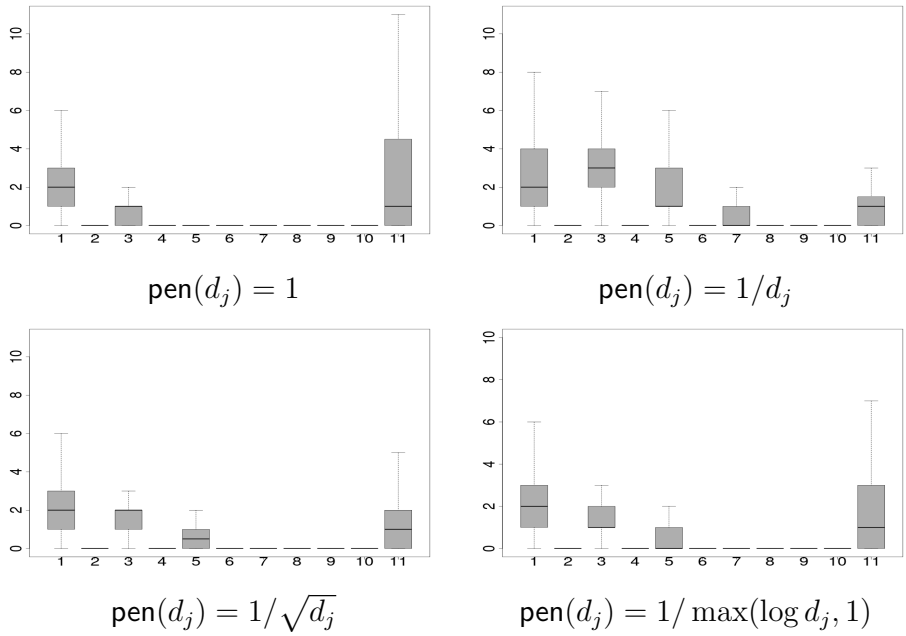
Table 2.5: Sensitivity to the choice of the penalty function pen: assessment of the score of group importance.



$\mathsf{pen}(d_j) = 1$ $\mathsf{pen}(d_j) = 1/d_j$

$\mathsf{pen}(d_j) = 1/\sqrt{d_j}$ $\mathsf{pen}(d_j) = 1/\max(\log d_j, 1)$

Figure 2.6: Group selection for TPLDA according to the penalty function $\mathsf{pen}(d_j)$ in experiment 4.



$\mathsf{pen}(d_j) = 1$ $\mathsf{pen}(d_j) = 1/d_j$

$\mathsf{pen}(d_j) = 1/\sqrt{d_j}$ $\mathsf{pen}(d_j) = 1/\max(\log d_j, 1)$

Figure 2.7: Group selection for TPLDA according to the penalty function $\mathsf{pen}(d_j)$ in experiment 5.

2.4 Application to tumor classification using gene expression data

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, that is a large number of variables relative to a much smaller number of observations, implies the use of features selection, clustering and/or regularized methods. Moreover, the resulted model must be easily understandable to enable identification of "marker" genes and characterization of the tumor subtypes.

In this paper, TPLDA, CART, GL are applied to datasets from three published cancer gene expression studies. For comparison purpose, the shrunken centroid regularized discriminant analysis (SCRDA) method (Guo et al., 2006), which is one of the standard methods used to classify tumors with gene expression data, is also applied to the three datasets. The objective is to elaborate a classification rule with a good prediction accuracy and which enables to highlight some relevant groups of genes. The three public microarray gene expression datasets used are briefly described below and in Table 2.6.

(1) The leukemia dataset (Golub et al., 1999) consists of an original training set, that gives the expression level of 7129 genes from 38 samples, and an original test set giving the expression level of 2185 genes from 34 patients. Based on pathological and histological criteria, in the training set, 27 tumor samples are classified as acute lymphobastic leukemias (called ALL) and the remaining 11 samples are classified as acute myeloid leukemias (called AML). In the test sample, there 20 ALL tumors and 14 AML tumors. The two datasets have been merged into a larger dataset that consists of the expression levels of 2135 genes from 72 samples (47 ALL and 25 AML).The data can be freely downloaded from http://www.broadinstitute.org/cgi-bin/cancer/publications/pub_paper.cgi?paper_id=43.

(2) The lymphoma dataset (Shipp et al., 2002) consists of 7129 gene expression levels from 77 lymphomas. The 77 samples are divided into 58 diffuse large B-cell lymphomas (DLBCL) and 19 follicular lymphomas (FL). The data can be found at https://github.com/ramhiser/datamicroarray/blob/master/data/shipp.RData.

(3) The colon dataset is from the microarray experiment of colon tissues samples of Alon et al. (1999). It contains the expression level of 2000 genes for 40 tumors and 22 normal colon tissues. The data can be freely downloaded from http://microarray.princeton.edu/oncology/affydata/index.html.

Dataset	Reference	Number of samples	Classes	Number of genes
Leukemia	Golub et al. (1999)	72	ALL(47), AML(25)	2185
Lymphoma	Shipp et al. (2002)	77	Cured(32), Disease(26)	7129
Colon	Alon et al. (1999)	62	Tumor(40), Normal(22)	2000

Table 2.6: The three datasets used in our application.

2.4.1 Data preprocessing and genes clustering

Following Dudoit et al. (2002), Shipp et al. (2002) and Sewak et al. (2009), a data preprocessing is applied to each dataset. First, a ceiling of 16000 units and a floor of 20 units are chosen to minimize noise effects. Next, in each dataset, only the first quartile of genes with the greatest variation across the sample is considered, the other genes are excluded. After that, we use independent component analysis (ICA) (Lee & Batzoglou, 2003) to cluster the remaining genes. In this approach, each independent component is considered as a putative biological pathway which can be characterized by the genes that contribute the most to the related independent component. Genes are then clustered into non-mutually exclusive groups based on their load on each ICA component. Finally, the expression of the selected genes are standardized so that the observations have zero mean and unit variance across genes.

2.4.2 Evaluation of the methods

The assessment of the methods is based on 500 repetitions of the following process.

First, each original dataset is randomly divided into a training sample, a validation sample and a test sample with the respective proportions $(0.8, 0.1, 0.1)$. In the training sample, classes are balanced by using the Synthetic Minority Over-sampling Technique proposed by Chawla et al. (2002).

For TPLDA and CART, a maximal tree is built on the training sample. To prune the maximal tree, TPLDA uses the pruning procedure described in Section 2.2.3 and the validation sample. CART uses cross-validation on the training sample and the minimal cost-complexity pruning method to select the final tree. For GL, the model is elaborated on the training set and the tuning parameter is selected by using 5-fold cross-validation. For SCRDA, the model is elaborated on the training set and the tuning parameter is selected by using the 10-fold cross-validation and the *Min-Min* rule proposed in the original paper (Guo et al., 2006). We use the function `rda` in the R package rda to compute SCRDA.

TPLDA and GL are applied on the groups of genes created during the clustering step. Since CART and SCRDA do not allow to take into account the groups of genes, these two methods are applied on the individual genes selected during the data-preprocessing phase.

Following previous studies with microarray data (Guo et al., 2006; Huang et al., 2009; Tai & Pan, 2007; Sewak et al., 2009; Dudoit et al., 2002), the predictive performances of all the assessed methods are measured by using the average error rate estimated on the test sample.

Finally, the measure of group importance provided by TPLDA is investigated using the leukemia data. TPLDA is applied on the original training set to elaborate a maximal tree that is next pruned by using the original test sample and the pruning method described previously. For the purpose of comparison, GL is also applied on the original training set to elaborate a model and the shrinkage parameter is selected by using the original test sample. The colon data and the lymphoma data are not used to study the interest of the measure of group importance since no test sample is available.

2.4.3 Results

Table 2.7 describes the datasets after performing data preprocessing and genes clustering. For each dataset, 15 non-mutually exclusive groups of genes are created. In Table 2.7, the number of selected genes refers to the number of distinct genes which belong to the groups. The elaboration of the groups of genes is explained in Section 2.6.4.

Dataset	**Size of the groups**	**Number of selected genes**
Leukemia	28	99
Lymphoma	18	100
Colon	26	106

Table 2.7: The datasets after applying data preprocessing and clustering genes into 15 groups.

Table 2.8 shows the results. For each method and each dataset, the average error rate over the 500 samples is displayed. For CART and TPLDA, the average tree depth is given. The table also shows the number of selected group (respectively genes) for GL (respectively SCRDA). Since the choice of both the number of groups and the group size may influence the results, several analyses have been performed by varying the values of these two parameters. Table 2.14 in Section 2.6.4 gives the results when genes are clustered into 50 groups. Results show no significant difference, that is consistent with previous sensitivity studies (Lee & Batzoglou, 2003).

	Classifier	Average error rate (in %)	Average tree depth	Number of groups/genes
Leukemia	TPLDA	**9 (11)**	1	
	CART	14 (14)	2	
	GL	**7 (10)**		4
	SCRDA	14 (14)		30
Lymphoma	TPLDA	**16 (15)**	2	
	CART	20 (17)	3	
	GL	**13 (14)**		5
	SCRDA	21 (18)		25
Colon	TPLDA	**20 (17)**	2	
	CART	26 (18)	2	
	GL	**16 (17)**		4
	SCRDA	31 (9)		6

Table 2.8: Average error rate averaged for 500 samples when genes are clustered into 15 groups. The standard error is given in brackets. The number of groups/genes is the average number of groups (respectively genes) included in the model for GL (respectively SCRDA).

The methods using the group structure (i.e. TPLDA and GL) outperform the others which emphasizes the interest of taking into account the group structure when data are grouped. Moreover, TPLDA performs consistently well for all datasets. GL tends to select more groups that TPLDA which may explain why GL performs a slightly better than TPLDA.

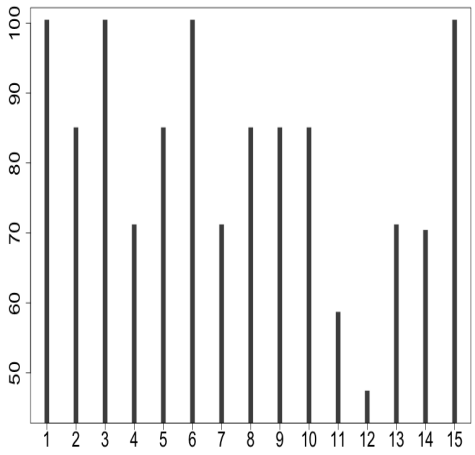


Figure 2.8: Importance of the 15 groups of genes in the leukemia study.

Nevertheless, contrary to GL, TPLDA provides automatically a measure of importance for

each group of genes, even for groups which are not included in the tree. This score of importance gives information about the relative importance of all the groups and also allows to perform selection of groups of genes. Figure 2.8 displays the measure of importance of each group, on the leukemia data. The measure of importance enables to highlight four groups: the first, the third, the sixth and the fifteenth groups. These groups build the same classification rule that explains why there have the same measure of importance. By comparison, GL includes only the fourth and the fifth groups in the model and provides no information about the prediction strength of the other groups. Therefore, GL does not able to identify relevant groups of genes that would not be included in the model because there are highly correlated with the two groups included in the model. Note that groups selected by GL are not considered as the most relevant groups with respect to the measure of importance computed with TPLDA. Nonetheless, these groups include some common genes (groups can be available at `https://github.com/apoterie/TPLDA`).

Since we have no expert knowledge for gene functions and pathways, we are not attempted to provide biological interpretation of these groups of genes. However, these results seem quite consistent with those presented in the original paper (Golub et al., 1999), since these groups include some genes that have been reported as informative genes in the original paper.

Thus, TPLDA can also be used to perform group variable selection. Compared to GL, TPLDA achieves a better trade-off between prediction accuracy and group variable selection.

Remark 2.4.1. The low predictive performances of SCRDA here may be explained by the exclusion of many genes during the data preprocessing and the clustering process. Indeed, previous studies showed that SCRDA gives good performances when the method is applied on datasets involving a large number of variables relative to the much small number of observations (see Huang et al., 2009; Guo et al., 2006, for instance).

2.5 Conclusion

In this work we have presented a new way to classify data with grouped inputs. Our approach consists in using recursive penalized linear discriminant analysis to build a classification tree based on the groups of variables. To our knowledge, it is the first classification trees algorithms dealing with grouped inputs.

The TPLDA method can be considered as a multivariate classification tree algorithm which uses linear combinations of inputs to build the partition, as already do several 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.), TPLDA is not computationally expensive. Moreover,

classification trees obtained by using TPLDA are more easily understandable since the classification rule is based on the group structure which makes sense.

Through applications on simulated datasets and real datasets, we have shown that TPLDA is well adapted to classify data with groups of inputs. Furthermore, the group importance measure computed within the TPLDA method allows to quantify the relevance of each group of variables, even if some groups are not included in the resulted final classification tree. Consequentially the TPLDA method also enables to answer the second most important issue in supervised classification: the identification of relevant groups of variables and/or the group variable selection. Thus, this algorithm shows promising results in terms of predictive performances, interpretation and variable selection.

2.6 Appendices

2.6.1 Time complexity of TPLDA

In the following section, the maximal time complexity at a node t of the TPLDA method is detailed. We assume that there are:

- n_t observations in the node t,
- J groups of variables denoted X^j, for $j = 1, \ldots, J$ and
- the group $\mathbf{X}^j$, with $j = 1, \ldots, J$, includes d_j variables such as $\mathbf{X}^j = (X_{j_1}, X_{j_2}, \ldots, X_{j_{d_j}})$.

To split a node t including n_t observations, TPLDA uses the following two steps:

- Step 1 within group PLDA.
  For any group $\mathbf{X}^j$ of variables, with $j = 1, \ldots, J$, a PLDA is applied on the node t and the shrinkage parameter λ_j is selected by cross-validation.
- Step 2 choosing the splitting group.
  For any group $\mathbf{X}^j$ of variables, with $j = 1, \ldots, J$, TPLDA computes the penalized decrease in node impurity resulting from splitting on group j and selects the group that maximizes it.

The time complexity of these steps are detailed below. Consider the group j, with $j = 1, \ldots, J$.

Complexity when performing PLDA on group j:
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 $\widehat{B}_t^j$ is $\mathcal{O}\left(n_t d_j^2\right)$.
- Complexity for constructing the diagonal positive estimate of the within covariance matrix $\widehat{\Sigma}_t^j$ is $\mathcal{O}\left(n_t d_j\right)$.
- Complexity of the eigen analysis of $(\widehat{\Sigma}_t^j)^{-1}\widehat{B}_t^j$ is $\mathcal{O}\left(d_j^2\right)$.
- Complexity of the eigen analysis of $(\widehat{B}_t^j)^{-1}\widehat{B}_t^j$ and the research for the dominant eigenvector is $\mathcal{O}\left(d_j^2\right)$.
- Complexity for estimating the penalized discriminant vector $\hat{\beta}^j$ by performing M iterations of the minimization-maximization algorithm (Lange et al., 2000) is $\mathcal{O}\left(M d_j^2\right)$.

$\Rightarrow$ So the maximal time complexity of performing PLDA on the group j in the node t is $\mathcal{O}\left(n_t d_j^2\right) + \mathcal{O}\left(n_t d_j\right) + \mathcal{O}\left(d_j^2\right) + \mathcal{O}\left(M d_j^2\right) = \mathcal{O}\left(n_t d_j^2\right)$ by supposing that $M < n_t$.

Complexity when selecting of the shrinkage parameter λ_j:
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. The maximal time complexity of this step is detailed below:

- Complexity for dividing the n_t observations in the node t into K disjoint samples $\{S_1, \ldots, S_K\}$ is $\mathcal{O}\left(n_t\right)$.
- For each fold k, $k = 1, \ldots, K$ and each value v_ℓ, $\ell = 1, \ldots, L$:
  - Complexity for performing a PLDA on $t \backslash S_k$ (i.e. all the disjoint sets $\{S_1, \ldots, S_K\}$ excepted S_k) with $\lambda_j = v_\ell$ is $\mathcal{O}\left(\frac{K-1}{K} n_t d_j^2\right)$.
  - Complexity for predicting the class of each observation in S_k using the resulted PLDA model computed on $t \setminus S_k$ is $\mathcal{O}\left(\frac{n_t}{K} d_j\right)$.
  - Complexity for computing the penalized decrease $\Delta_j(t, v_\ell)$ in node impurity is $\mathcal{O}\left(n_t\right)$.
- Complexity for choosing the value in the grid $\{v_1, \ldots, v_L\}$ which maximizes the penalized decrease in node impurity is $\mathcal{O}\left(1\right)$.

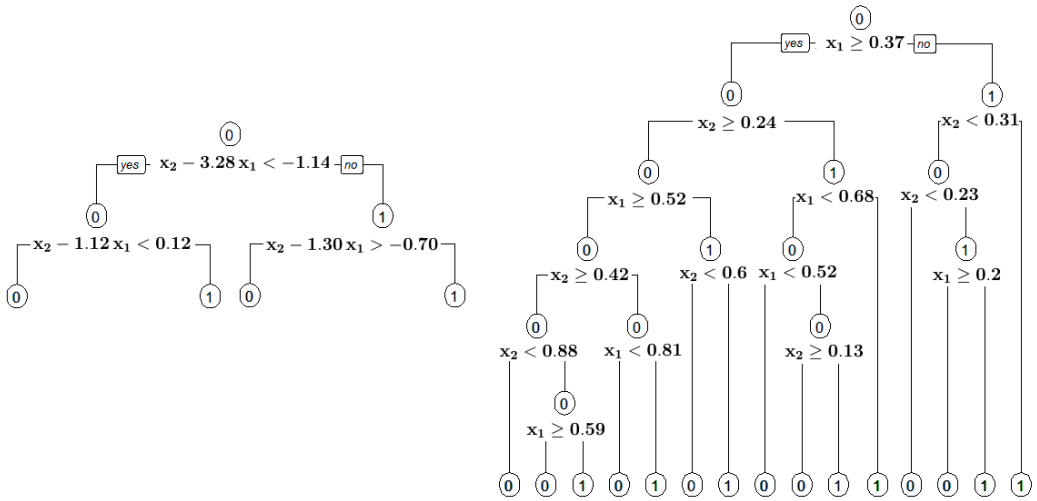
$\Rightarrow$ So the complexity for selecting the value of λ_j is $\mathcal{O}\left(n_t\right) + \mathcal{O}\left(L(K-1) n_t d_j^2\right) + \mathcal{O}\left(L n_t d_j\right) + \mathcal{O}\left(L n_t\right) + \mathcal{O}\left(1\right) = \mathcal{O}\left(L(K-1) n_t d_j^2\right)$.

Complexity when choosing the splitting group:
TPLDA 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 t is in the worst case $(Jn_t d^2_{\max}) + \mathcal{O}\left(JL(K-1)n_t d^2_{\max}\right) + \mathcal{O}\left(1\right) = \mathcal{O}\left(JLKn_t d^2_{\max}\right)$ with $d_{\max} = \max_j(d_j)$.

2.6.2 Additional figures about the illustration of the TPLDA method on a simple example

Figure 2.9 displays the two trees built by CART and TPLDA in the simple example used to illustrate TPLDA in Section 2.2.4. As mentioned previously, in this example, the TPLDA tree is much easier than the CART tree. Moreover, the simple TPLDA tree is as accurate as the complex CART tree (TPLDA AUC = 0.90, CART AUC = 0.89).



The TPLDA tree The CART tree

Figure 2.9: The two trees associated to the TPLDA and CART partitions displayed in Figure 2.2. Circles define the nodes and the figure in each node indicates the node label. The splitting rule is denoted below each node.

2.6.3 Additional information about the numerical experiments

This section provides additional results and figures about the simulation studies.

Justification for using PLDA instead of FDA in the splitting process

First of all, here we discuss the choice of the use of PLDA instead of FDA in the splitting process. In TPLDA, PLDA is replaced by FDA. This modified TPLDA is named TLDA and is applied on each sample of the first three experiments.

Table 2.9 displays the simulation results for TLDA in comparison with TPLDA. First, when data are not grouped, TPLDA and TLDA lead to almost the same results and can be then used interchangeably. Indeed, when the group size equals 1 and if the regularized parameter in the PLDA problem (2.3) is set to zero, the FDA problem and the PLDA problem are identical.

In the second and the third experiment, TLDA underperforms TPLDA. This can be explained by the fact that FDA performs badly in small nodes i.e. in the nodes where the number of observations is small relative to the size of some groups of variables (Shao et al., 2011; Friedman, 1989; Xu et al., 2009; Bouveyron et al., 2007). Yet, tree elaboration is based on a recursive splitting procedure which creates nodes that becomes smaller and smaller whereas the sizes of input groups remain unchanged.

Then FDA may not be appropriate for estimating recursively the hyperplane splits. This is well illustrated by the performances of TLDA in the third experiment where the groups of input variables are large compared to the number of observations in the training sample. Indeed, in the first split, the FDA used to split the entire data space overfits the training set. This can be seen in Figure 2.10: 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 at least 25 % of the simulations (Table 2.9).

Conversely, TPLDA does not seem to be affected by the high-dimension. Then, PLDA overcomes the weakness of FDA in high-dimensional situations.

	TPLDA	TLDA
Experiment 1		
AUC	0.66 (0.65,0.68)	0.66 (0.65,0.67)
Tree depth	4 (3,5)	4 (3,5)
Experiment 2		
AUC	0.76 (0.74,0.77)	0.67 (0.64,0.69)
Tree depth	3 (3,4)	3 (2,3)
Experiment 3		
AUC	0.83 (0.7,0.85)	0.5 (0.5,0.52)
Tree depth	2 (2,3)	1 (0,2)

Table 2.9: Performances of TPLDA and TLDA. For each criterion, the median value is given following by the values in brackets of the first and the third quartiles.

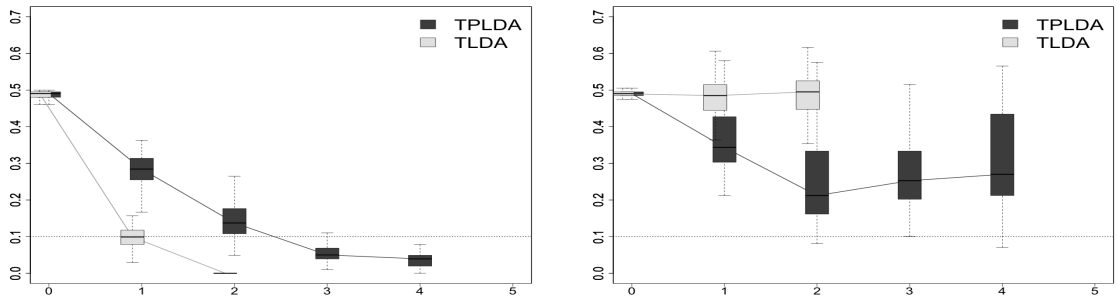
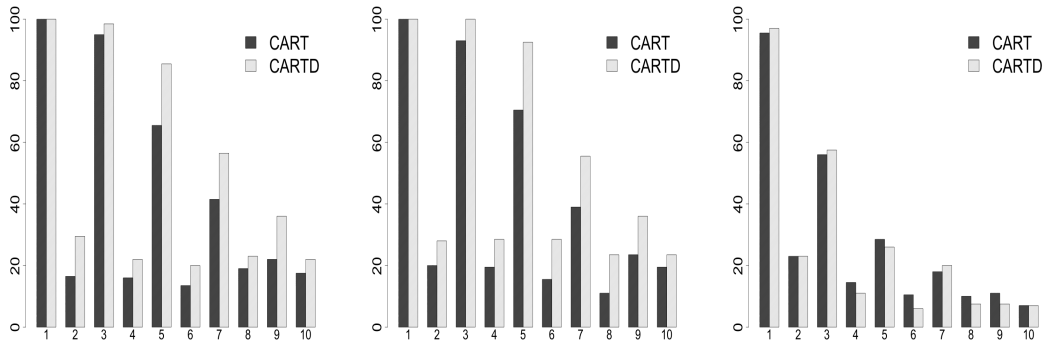


Figure 2.10: Misclassification error estimate according to the tree depth on the training set (left) and on the validation set (right) in experiment 3. The dotted lines denote the values of the Bayes error (Bayes error=10%).

Sensitivity to the pruning strategy for CART

Here, the performances of CART when using the cost-complexity pruning strategy are compared to those obtained by using the proposed pruning strategy based on the tree depth, in the first three experiments. The approach using the proposed pruning strategy is named CARTD (while the approach using the cost-complexity pruning is named CART). The results are given in Table 2.10. Figure 2.11 displays the group selection frequencies of CART and CARTD. 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. CARTD trees may be slightly smaller. Moreover, the group selection frequencies do not really differ: they are lightly higher when using the proposed pruning strategy based on the depth. Thus, CART performances do not seem to be sensitive to the choice of one of the two pruning methods.



Experiment 1 Experiment 2 Experiment 3

Figure 2.11: Group selection frequency (in %) for CART according to the pruning strategy in the first three experiments.

	CART	CARTD
Experiment 1		
AUC	0.67 (0.65,0.68)	0.67 (0.66,0.69)
Tree depth	5 (3,7)	5 (4,6)
Experiment 2		
AUC	0.68 (0.66,0.70)	0.68 (0.67,0.70)
Tree depth	6 (4,8)	5 (4,7)
Experiment 3		
AUC	0.64 (0.62,0.66)	0.65 (0.62,0.67)
Tree depth	4 (2,5)	3 (2,4)

Table 2.10: Performances of CART and CARTD. For each criterion, the median value is given following by the values in brackets of the first and the third quartiles.

Additional results

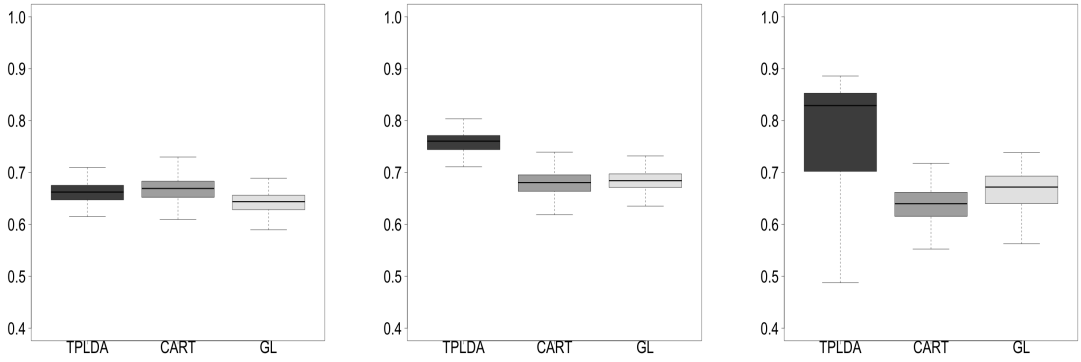
Table 2.11 displays the simulation results for TPLDA, TLDA, CART and GL for the fourth and fifth scenarios. As previously, TLDA underperforms TPLDA. GL and CART overperform slightly TPLDA when no penalty function is used.

	TPLDA	TLDA	CART	GL
Experiment 4				
AUC	0.65 (0.60,0.70)	0.59 (0.54,0.65)	0.68 (0.66,0.69)	0.66 (0.65,0.68)
Tree depth	2 (2,4)	2 (2,3)	5 (4,7)	.
Model size	.	.	.	4 (3,5)
Experiment 5				
AUC	0.66 (0.60,0.71)	0.58 (0.53,0.63)	0.68 (0.66,0.69)	0.66 (0.64,0.68)
Tree depth	2 (2,3)	2 (2,3)	5 (4,7)	.
Model size	.	.	.	5 (4,6)

Table 2.11: Performances of TPLDA, TLDA, CART and GL in Experiment 4 and Experiment 5. For each criterion, the median value is given following by the values in brackets of the first and the third quartiles. The model size gives the number of groups of variables included in the GL model.

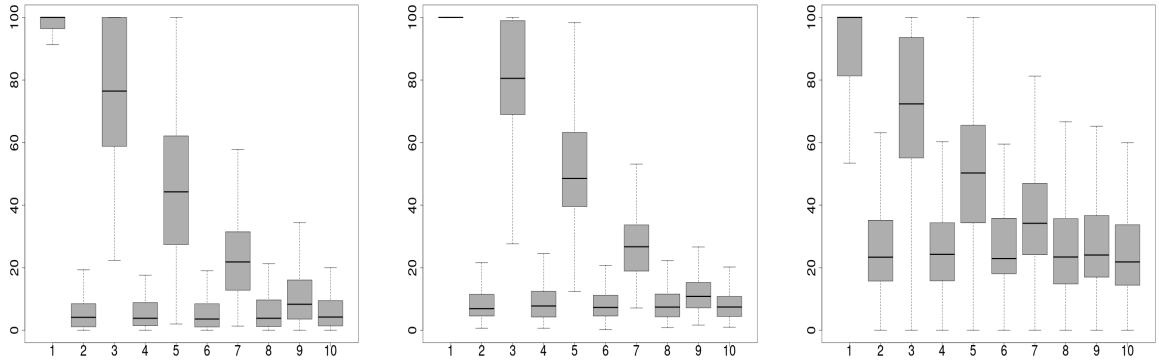
Additional figures

Additional figures about the simulation studies are displayed in this subsection. Figure 2.12 displays the predictive performances of TPLDA, CART and GL in the first three experiments. Figure 2.13 shows the distribution of the importance score for each group in the first three experiments. Figure 2.14 and Figure 2.15 display the distribution of the importance score for each group in the fourth and the fifth experiments according to the penalty function.



Experiment 1 Experiment 2 Experiment 3

Figure 2.12: Predictive performances of the assessed methods: boxplots of the AUC for the first three experiments.



Experiment 1 Experiment 2 Experiment 3

Figure 2.13: Distribution of the importance score for each group in the first three experiments.

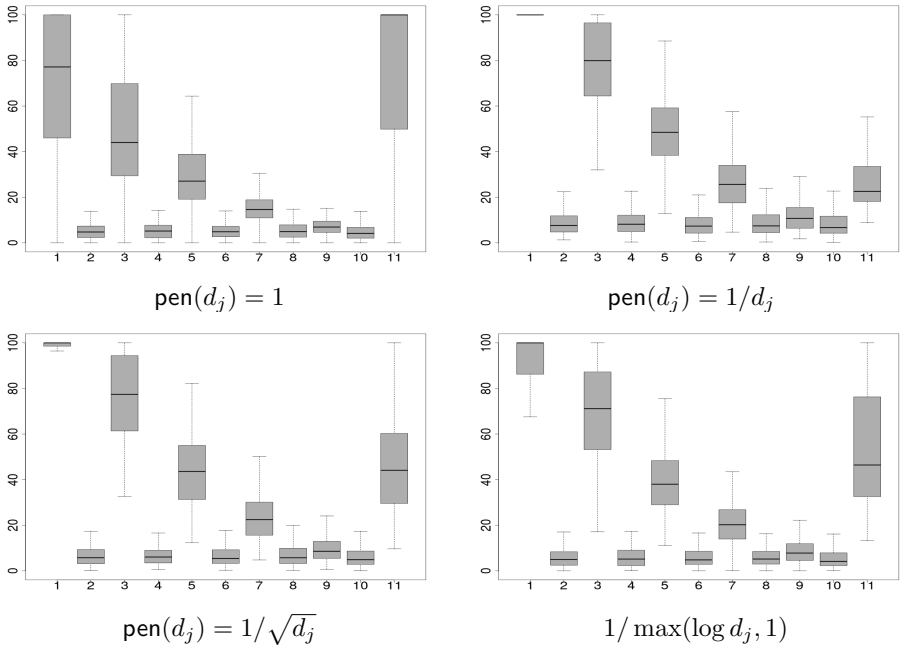


Figure 2.14: Distribution of the importance of each group according to the penalty function in experiment 4.

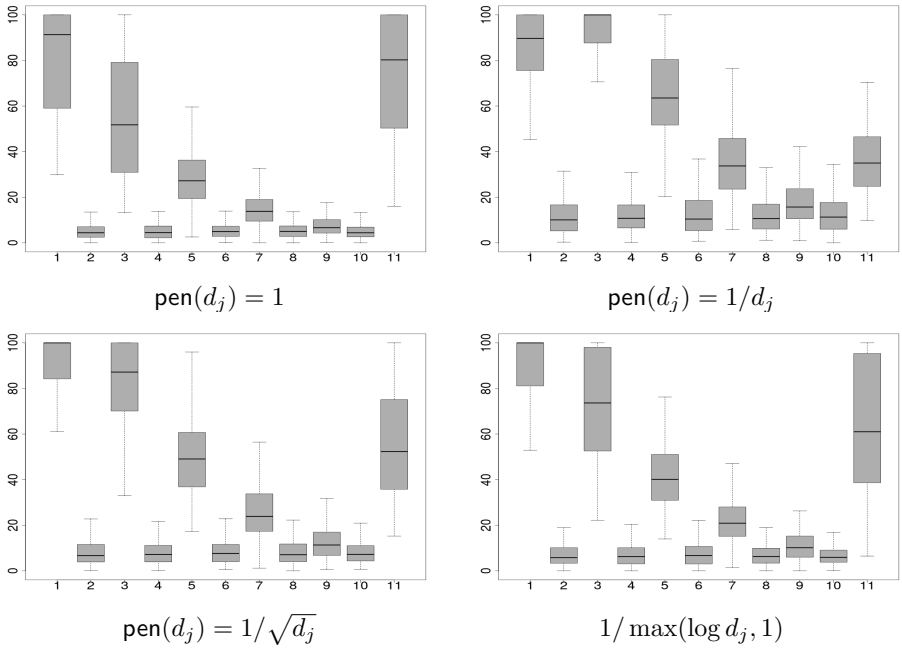


Figure 2.15: Distribution of the importance of each group according to the penalty function in experiment 5.

2.6.4 Additional information about the application to gene expression data

Data preprocessing and genes clustering

Following Lee & Batzoglou (2003), we assume that each independent component refers to a putative biological process and that a group of genes is then created for each independent component. For a given independent component, the most important genes are the genes with the largest loads in absolute terms. Then, the group of genes associated to the given independent component includes the $C\%$ of genes with the largest loads in absolute terms. The number of groups J (or equivalently the number of independent components) and the threshold parameter C are tuning parameters.

For each dataset, several values for the clustering parameters (J,C) are chosen in order to assess the sensitivity of the predictive performances of the methods TPLDA, CART, GL and SCRDA to the values of these parameters. For each dataset we show the results for two couples (J,C) (Table 2.12). In each dataset, genes are clustered into 15 or 50 non-mutually exclusive groups (Tables 2.7 and 2.13). Table 2.14 shows the results when genes are clustered into 50 groups of equal size. These results are not significantly different from those obtained when using 15 groups (Table 2.8).

Dataset	Total number of genes	Number of groups (J)	Threshold parameter (C)
Leukemia	2186	15	5
		50	2.5
Lymphoma	7129	15	1
		50	0.5
Colon	2000	15	5
		50	2.5

Table 2.12: Choice of the clustering parameters for each dataset.

Dataset	**Size of the groups**	**Number of selected genes**
Leukemia	14	101
Lymphoma	10	108
Colon	14	112

Table 2.13: The datasets after applying data preprocessing and clustering genes into 50 groups.

	Classifier	Average error rate (%)	Average tree depth	Average number of groups/genes
Leukemia	TPLDA	**9 (12)**	1	
	CART	17 (15)	2	
	GL	**7 (10)**		5
	SCRDA	13 (14)		36
Lymphoma	TPLDA	**16 (15)**	2	
	CART	22 (17)	2	
	GL	**12 (13)**		7
	SCRDA	17 (16)		28
Colon	TPLDA	**20 (15)**	1	
	CART	25 (17)	2	
	GL	**16 (17)**		5
	SCRDA	29 (11)		16

Table 2.14: Average error rate averaged for 500 samples when genes are clustered into 50 groups. The standard error is given in brackets. The number of groups/genes is the average number of groups (respectively genes) included in the model for GL (respectively SCRDA).

Chapter 3

Decision trees and random forests for grouped variables

Abstract. In the supervised learning, we consider the problem of estimating a prediction rule based on groups of inputs. As this problem has mainly been studied in parametric and semi-parametric settings, in this paper, we propose two original non-parametric methods that allow to construct prediction rules, both in regression and classification, based on predictor groups. The first proposed method is called the Classification And Regression Trees for Grouped Variables (CARTGV). It consists in building a large non-binary tree and next selecting an optimal tree by using a generalization of the efficient minimal cost-complexity pruning to non-binary trees. Two new measures of group importance are also proposed in the context of CARTGV trees. Next, we introduce a new random forests method adapted for grouped variables. We call this approach Random Forests for Grouped Variables (RFGV). This method, which can be considered as an extension of Breiman's random forests, combines a lot of random trees built with respect to a variant of CARTGV. Moreover, RFGV uses a measure of importance that allows to assess the relevance of each predictor group. The good performances of these two approaches are demonstrated in various comprehensive simulation studies.

Contents

3.1 Introduction 59
3.2 CARTGV: CART for grouped variables 61
3.2.1 The statistical framework 61
3.2.2 The tree growing procedure 62
3.2.3 The pruning procedure 65
3.2.4 Group importance measures 68
3.2.5 Illustrations 70
3.3 Random forests for grouped variables 81
3.3.1 Principles of the random forests for grouped variables 81

3.3.2 Details on RFGV parameters 84
3.3.3 Permutation-based group importance measure 85
3.4 Numerical experiments 86
3.4.1 Description of the datasets 86
3.4.2 Performances of the methods 90
3.4.3 Assessment of the measures of group importance 96
3.5 Conclusion 98
3.6 Appendices 98
3.6.1 Proof of the positivity of the decrease in impurity in the context of non-binary splits 98
3.6.2 Time complexity of the CARTGV algorithm 100
3.6.3 Generalization of the minimal cost-compexity pruning for CARTGV trees 100
3.6.4 CARTGV: additional information about the numerical experiments 114

3.1 Introduction

In this paper, we investigate prediction of a random variable Y, which can be a class label or a real, based on input variables having a group structure. As an example of input group structure, consider the use of functional variables, such as a spectrometry curve. In this case, one is often interested in identifying parts of the curve which are relevant for prediction. These parts constitute the input groups Tardivel et al. (2017). More generally, a group structure can be defined when there exists a natural association between some subsets of inputs, such as in gene expression data where some genes can share common biological functions, which results in strong correlations between gene expression levels (Jiang et al., 2004; Villa-Vialaneix et al., 2013). If such a group structure exists, a prediction rule that takes it into account will improve both interpretation and prediction accuracy (compared to prediction rule using only individual predictors), see Gregorutti et al. (2015). Parametric and semi-parametric prediction methods with input group structure have already been developed. See for example the linear and generalized regression models regularized by the Group Lasso penalty (Yuan & Lin, 2006; Meier et al., 2008). But as far as we know, this has not been studied for decision trees and random forest (Breiman et al., 1984; Breiman, 2001). This paper intends to fill this gap.

Classification And Regression Trees (CART, Breiman et al., 1984) constructs prediction rules for both regression and classification. The procedure is based on two steps, called the tree growing phase and the pruning phase. During the tree growing phase, the algorithm builds a maximal binary tree by means of a recursive partitioning of the input space. Starting with

all the data, the algorithm partitions the data space into two disjoint subregions, also called nodes, and repeats the splitting procedure on the resulting nodes. This recursive splitting strategy is then repeated on all nodes on until every node is homogeneous, that means every node contains only instances having the same value for the response variable. At each step, a node is divided into two child nodes based on the value of one input also called the splitting variable. The choice of the optimal split is based on the minimization of an impurity function, i.e. the algorithm seeks the split that yields the most homogeneous child nodes. Generally, the impurity function is the sum squared error in regression and the Gini index in classification. The nodes that are not split are called either the terminal nodes or the leaves. At the end of the splitting process, the leaves define a partition of the input space which can be represented as a maximal binary tree. Moreover, a predictive value for the response variable Y is assigned to each leaf (the average of Y in the leaf in regression or the majority vote in the leaf in classification). Generally, the maximal tree is not optimal with respect to any performance criterion (such as the misclassification error in classification and the sum squared error in regression). Indeed, an excessive large number of nodes is prone to overfitting. Thus, the tree is pruned. This pruning phase consists in extracting a finite collection of pruned subtrees of the maximal tree and then selecting a final tree among the sequence of subtrees thus constructed. To do this, CART uses a very efficient pruning algorithm called the minimal cost-complexity pruning (chap. 3, Breiman et al., 1984).

CART is a very attractive method for constructing prediction rules since the method elaborates a prediction rule easily interpretable with its tree-structure and is distribution-free. Nevertheless, this algorithm is known to be unstable. A small perturbation in the training sample can considerably change the prediction rule. For this reason, Breiman (2001) proposed the random forests (RF), which can be used for both regression and classification. RF consists in aggregating a large collection of trees-based estimators, as the bagging method (Breiman, 1996). Trees are built based on bootstrap samples and with respect to a variant of CART. First, instead of CART, at each node, a small number of inputs is randomly selected and the optimal split is then determined based only on this subset. Next, all trees are fully grown and are not pruned. A random forest has generally better predictive performances than a single decision tree. Indeed, a maximal tree has low bias but high variance, so aggregating a large number of maximal trees allows to achieve a bias-variance trade-off. RF has become very popular and has been successfully applied to many problems in various fields, including bioinformatics (Díaz-Uriarte & Alvarez de Andrés, 2006), ecology (Cutler et al., 2007) and object recognition (Shotton et al., 2011). What has also greatly contributed to the popularity of the method is the introduction of two measures of importance which allow to quantify the relevance of all inputs with respect to the prediction of the response variable.

CART and RF are very efficient techniques for estimating prediction rules. They offer lots of advantages such as wide applicability, fast and easy implementation and quantification of the relevance of each input variable through importance measures. However, these methods cannot accommodate groups of inputs. Thus, in this paper, we adapt them to this context.

We first introduce Classification And Regression Trees for Grouped Variables (CARTGV), a new tree-based method that constructs prediction rules based on group of inputs. As for CART, CARTGV builds a maximal tree by recursively partitioning the input space. At each step, a node is split into several child nodes, based on a group of inputs. At the end of this process, the maximal tree, which is not binary, is pruned using a generalization of the efficient minimal cost-complexity pruning strategy proposed by Breiman et al. (1984). This pruning strategy was originally introduced for binary decision trees. Here, we prove validity in the more general setting of non-binary decisions trees. We also introduce two new importance measures for groups of input variables. The first score is based on the "corrected" sum of the decrease in node impurity while the second one uses an extension of the notion of surrogate splits proposed by Breiman et al. (1984) to the context of groups of inputs. These scores are based on a CARTGV tree and allow to evaluate the relevance of each group of inputs. Next, we propose of a new random forests method adapted to groups of inputs. This method, called Random Forests for Grouped Variables (RFGV), is a modification of Breiman's random forests. It consists in aggregating a large number of decision trees built with respect to a variant of CARTGV. Moreover, RFGV provides a measure of importance of the groups based on the group importance measure proposed by Gregorutti et al. (2015).

The paper is organized as follows. In Section 3.2, we first introduce the framework and some useful notations. Section 3.2 next describes CARTGV and the associated measures of group importance. In this section, an illustration of CARTGV and its two scores of importance is also provided. In Section 3.3, we introduce RFGV and the group importance measure proposed by Gregorutti et al. (2015). In Section , performances of RFGV and CARTGV are assessed and compared to RF and CART in an extensive simulation study. A comprehensive explanation of the minimal cost-complexity pruning, as well as the proof of its validity in the more general framework of non-binary trees is given in Section 3.6. Computation of the time complexity of CARTGV and additional information about the numerical experiments are also given at the end of the chapter, in Section 3.6.
The proposed methods as well as the generalization of the minimal cost-complexity pruning algorithm have been implemented for the binary classification setting in `R` language.

3.2 CARTGV: CART for grouped variables

3.2.1 The statistical framework

Let $(\mathbf{X}, Y)$ be a random vector taking values in $\mathcal{X} \times \mathcal{Y}$, where $\mathbf{X} = (X_1, \ldots, X_d)$ is a vector of inputs, Y is the response variable and $\mathcal{X} = \mathbb{R}^d$. In the regression setting, $\mathcal{Y} = \mathbb{R}$ and in classification $\mathcal{Y} = \{1, \ldots, K\} \subset \mathbb{N}$, $K \geq 2$. Let $(\mathbf{X}_1, Y_1), \ldots, (\mathbf{X}_{n+m}, Y_{n+m})$ be independent copies of $(\mathbf{X}, Y)$ which are randomly split into a training set $\mathcal{D}_n = \{(\mathbf{X}_1, Y_1), \ldots, (\mathbf{X}_n, Y_n)\}$ and a validation set $\mathcal{V}_m = \{(\mathbf{X}_{n+1}, Y_{n+1}), \ldots, (\mathbf{X}_{n+m}, Y_{n+m})\}$. A prediction rule is a measurable function $\hat{f} : \mathbb{R}^d \times (\mathbb{R}^d \times \mathcal{Y})^{n+m} \to \mathcal{Y}$ that predicts the value of the response Y by

$\hat{f}(\mathbf{x}, \mathcal{D}_n)$, where $\mathbf{x} \in \mathbb{R}^d$ is a new observation. In what follows, we will write $\hat{f}(\mathbf{x})$ for convenience. In the regression setting, $\hat{f}$ is an estimate of $f^\star(\mathbf{x}) = \mathbb{E}[Y|\mathbf{X} = \mathbf{x}]$, for every $\mathbf{x} \in \mathcal{X}$. Otherwise, in classification, $\hat{f}$ is an estimate of the Bayes classifier defined, for every $\mathbf{x} \in \mathcal{X}$, by $f^\star(\mathbf{x}) = \text{argmax}_{k\in\{1,\dots,K\}} \mathbb{P}[Y = k|\mathbf{X} = \mathbf{x}]$.

Here, we consider the situation where $\mathbf{X}$ is structured into J known groups. For any $j = 1, \dots, J$, let $\mathbf{X}^j = (X_{j_1}, X_{j_2}, \dots, X_{j_{d_j}})$ denote the j-th group of size d_j. To simplify matter, the J groups are ordered such that $\mathbf{X} = (\mathbf{X}^1, \dots, \mathbf{X}^J)$. Some inputs can belong to several groups (I.e. groups are not necessarily disjoint). The objective is to use the training sample $\mathcal{D}_n$, the validation sample $\mathcal{V}_m$ and the known input group structure to build a prediction rule $\hat{f}$.

For this purpose, we first propose a new tree-based approach called CARTGV that constructs classification and regression trees by using groups of inputs. As for CART, CARTGV is based in two steps, that we name *the tree growing procedure* and *the pruning procedure*. First, the tree growing procedure enables to build a fully-grown tree which is next pruned, during the pruning procedure. These two steps are detailed in the two following subsections.

3.2.2 The tree growing procedure

CARTGV builds a maximal tree by recursively partitioning the input space into smaller and smaller disjoint subregions called nodes. At each step, a node is split into child nodes. Let t be a node. A split of t consists in selecting a splitting group and a split based on the inputs belonging to the splitting group. This choice is performed in two phases, that are now described in more details.

- Step 1: Choice of a *splitting tree* for each group.
  First, for any $j = 1, \dots, J$, the algorithm splits the node t by applying the CART growing procedure on each group $\mathbf{X}^j$. That is, the algorithm builds a CART tree based on the inputs $X_{j_1}, X_{j_2}, \dots, X_{j_{d_j}}$ that belong to group j. Formally, we first split the node t into two child nodes t_1 and t_2 defined by

  $$t_1 = \{\mathbf{X}_i, i \leq n : \mathbf{X}_i \in t \text{ and } X_{ij_k} \leq s\} \quad \text{and} \quad t_2 = \{\mathbf{X}_i, i \leq n : \mathbf{X}_i \in t \text{ and } X_{ij_k} > s\}.$$

  The splitting variable X_{j_k} and the splitting point s are selected by maximizing the decrease in impurity between t and its two child nodes defined by

  $$n_t \mathrm{Q}(t) - n_{t_1} \mathrm{Q}(t_1) - n_{t_2} \mathrm{Q}(t_2), \tag{3.1}$$

  where Q is an impurity function which measures the homogeneity of the observations in a node and n_t denotes the number of observations in $\mathcal{D}_n$ that fall into the node t.

This splitting process is repeated on each child node until the maximal depth of the tree (denoted D_j) is reached. Note that a homogeneous node, that we recall is a node which contains only observations having the same value for the response variable, are not split. This tree, which is called the *splitting tree* of t based on group j, has root t and a maximal depth of D_j. Let $\tilde{t}(j)$ denote the set of leaves of this tree and let $|\tilde{t}(j)|$ denote the cardinality of $\tilde{t}(j)$. Note that the value of the tuning parameter D_j need to be chosen. We recommend to take this parameter small ($D_j = 2$ or 3) in order to avoid overfitting. The calibration of this parameter is discussed in Section 3.2.5.

- Step 2: Choice of the splitting group.
  Let $\Delta\mathrm{Q}(t, j)$ denote the decrease in impurity between the node t and the leaves of the splitting tree of t based on group j:

$$\Delta\mathrm{Q}(t, j) = n_t \mathrm{Q}(t) - \sum_{t' \in \tilde{t}(j)} n_{t'} \mathrm{Q}(t') . \tag{3.2}$$

  In the context of grouped variables, the criterion (3.2) may not be appropriate to select the best splitting group since it tends to foster larger groups. Indeed, the largest groups have more possible splits than the smallest groups. Consequently, it is more likely that the largest groups will be optimal with respect to the decrease in node impurity (3.2) (Strobl et al., 2007). Thus, to control this selection bias, we propose to penalize the criterion (3.2) by a decreasing function $\mathsf{pen}(d_j)$ of the group size d_j, that is:

$$\Delta_p\mathrm{Q}(t, j) = \mathsf{pen}(d_j)\Delta\mathrm{Q}(t, j). \tag{3.3}$$

  The algorithm selects then the splitting group $j_t^\star \in \{1, \ldots, J\}$ that maximizes the penalized criterion (3.3).
  Various penalty functions can be used. Here, we suggest the three following functions:

$$\begin{aligned} \mathsf{pen}(d_j) &= (d_j)^{-1}, \\ \mathsf{pen}(d_j) &= (\sqrt{d_j})^{-1}, \\ \mathsf{pen}(d_j) &= (\max\{\log d_j, 1\})^{-1}. \end{aligned} \tag{3.4}$$

  The choice of pen is investigated in Section 3.2.5, through numerical experiments. Note that if all groups have the same size, there is no need to use a penalty function.

Remark 3.2.1.

- The impurity function Q measures the homogeneity of a node. It should be small when values of Y are closed to each other in the node and large otherwise. For regression problems, we propose to use the variance in the node

$$\mathrm{Q}(t) = \frac{1}{n_t} \sum_{i:\mathbf{X}_i \in t} (Y_i - \bar{Y}_t)^2$$

where $\bar{Y}_t$ stands for the mean of the Y_i such that $\mathbf{X}_i \in t$. For classification problems, we consider the Gini impurity function defined by

$$\mathrm{Q}(t) = \sum_{k=1}^{K} \pi_k(t)(1 - \pi_k(t)) \quad \text{where} \quad \pi_k(t) = \frac{1}{n_t} \sum_{i:\mathbf{X}_i \in t} \mathbb{1}_{Y_i = k}.$$

- The decrease in node impurity defined by (3.2) is always positive, regardless the split. In other words, the impurity is never increased when splitting. The proof is given in Section 3.6.1.

At the beginning of the tree-growing phase, steps 1 and 2 are applied on the whole input space, which is then partitioned into several child nodes. The two steps are next repeated recursively on all child nodes and all resulting nodes, until the nodes are homogeneous. At the end of the splitting process, the partitioning of the input space can be represented as a maximal tree $T_{\max}$. Moreover, the set of leaves of $T_{\max}$, which are denoted by $\widetilde{T}$, defines the prediction rule $\hat{f}_{T_{max}}$:

$$\hat{f}_{T_{\max}}(\mathbf{x}) = \sum_{t \in \widetilde{T}} \hat{y}_t \mathbb{1}_t(\mathbf{x}), \quad \text{for all } \mathbf{x} \in \mathbb{R}^d,$$

where $\mathbb{1}_t(\mathbf{x})$ is the indicator function that equals to 1 if $\mathbf{x} \in t$ and 0 otherwise. $\hat{y}_t$ takes values in $\mathcal{Y}$ and represents the prediction value in the node t. It is the empirical mean of Y in t in regression:

$$\hat{y}_t = n_t^{-1} \sum_{i:\mathbf{X}_i \in t} Y_i,$$

and in classification, the majority vote:

$$\hat{y}_t = \underset{k=1,\ldots,K}{\operatorname{argmax}}\, n_t^{-1} \sum_{i:\mathbf{X}_i \in t} \mathbb{1}_{Y_i = k}.$$

Figure 3.1 displays a CARTGV tree and a splitting tree.

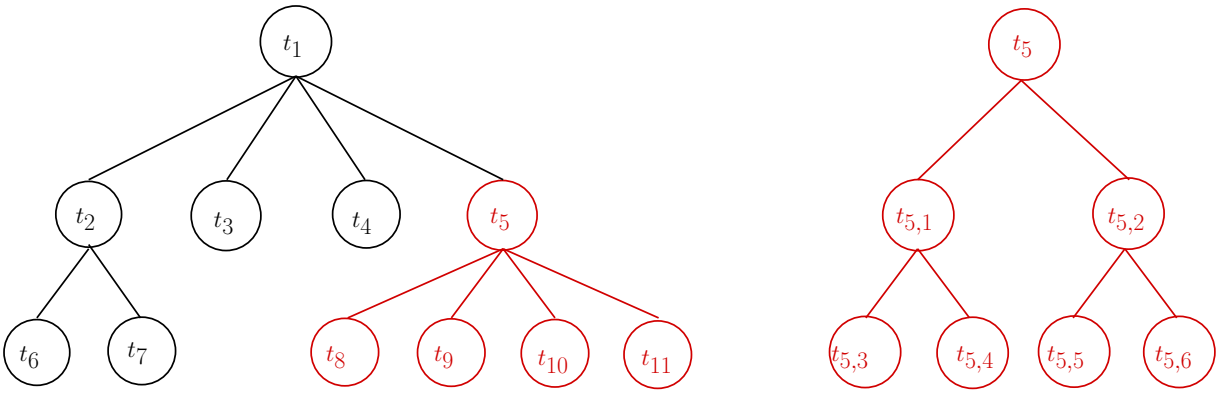


Figure 3.1: A CARTGV tree (left) and a splitting tree (right). The splitting tree divides the node t_5 into the child nodes t_8, t_9, t_{10} and t_{11} such that: $t_8 = t_{5,3}$, $t_9 = t_{5,4}$, $t_{10} = t_{5,5}$, $t_{11} = t_{5,6}$.

Remark 3.2.2.

- Splitting trees are binary trees while CARTGV trees are multivariate trees (Figure 3.1).
- If for any $j = 1, \ldots, J$, $d_j = 1$ (i.e. inputs are not grouped) and $D_j = 1$ (splitting trees are defined by a single split), then CARTGV is equivalent to CART.

3.2.3 The pruning procedure

We first introduce some notations. Consider a tree T.

- The cardinality of the set of its leaves $\widetilde{T}$ is denoted $|\widetilde{T}|$.
- If t is a non-terminal node of T, then T_t denotes the branch of T coming from t, that is the subtree of T with root t.
- Write $T' \preceq T$, if T' is a pruned subtree of T, that is T' is a subtree of T having the same root as T.

Let $\mathcal{R}(T, \mathcal{D})$ define the prediction error of a tree T on a given sample $\mathcal{D}$ of size $|\mathcal{D}|$. In regression, $\mathcal{R}(T, \mathcal{D})$ is the mean square error

$$\mathcal{R}(T, \mathcal{D}) = \frac{1}{|\mathcal{D}|} \sum_{i:(\mathbf{X}_i, Y_i) \in \mathcal{D}} (\hat{f}_T(\mathbf{X}_i) - Y_i)^2,$$

while in classification, $\mathcal{R}(T, \mathcal{D})$ is the empirical misclassification rate

$$\mathcal{R}(T, \mathcal{D}) = \frac{1}{|\mathcal{D}|} \sum_{i:(\mathbf{X}_i, Y_i) \in \mathcal{D}} \mathbb{1}_{\hat{f}_T(\mathbf{X}_i) \neq Y_i}.$$

In what follows, the prediction error $\mathcal{R}(T, \mathcal{D}_n)$ of a tree T on the training sample $\mathcal{D}_n$ will be written $\mathcal{R}(T)$ for the sake of convenience.

Once constructed, the fully-grown tree $T_{\max}$ is pruned to prevent from overfitting. One natural idea to prune it is to consider all pruned subtrees of $T_{\max}$ and to select the best pruned subtree according to a given performance criterion (such as the prediction error computed on the validation set $\mathcal{T}_q$). Nevertheless, since the number of subtrees of $T_{\max}$ may be very large, exhaustive search may rapidly be infeasible. Therefore, we propose to generalize the minimal cost-complexity pruning proposed by Breiman (1996) to our context. As originally this pruning method was introduced to prune CART binary trees, we have proved that it is valid in the context of non-binary decision trees and we have then naturally extend the method to this context. The proof and a comprehensive explanation of the minimal cost-complexity pruning are given at the end of the chapter, in Section 3.6.3. We now briefly describe the principle of the cost-minimal pruning method in the context of CARTGV trees, i.e. in the context of non-binary decision trees.

The main idea is to consider a small collection of pruned subtrees of $T_{\max}$, optimal with respect to the cost complexity measure defined as:

$$\mathcal{R}_\alpha(T) \;=\; \mathcal{R}(T) \;+\; \alpha|\widetilde{T}|, \quad \alpha \in \mathbb{R}^+, \tag{3.5}$$

for any tree T. According to this definition, the cost-complexity of a tree T depends on two components: the prediction error $\mathcal{R}(T)$ and a penalty term $\alpha|\widetilde{T}|$ for the complexity of T. The variable α is a tuning parameter which controls the complexity of the tree. The higher α is, the more penalized are trees with a lot of leaves.

The minimal cost-complexity pruning consists first in finding, for any α, the smallest pruned subtree of $T_{\max}$ which minimizes $\mathcal{R}_\alpha$. The following theorem shows that we only have to consider a finite sequence of values of α to obtain all the subtrees of $T_{\max}$ that minimizes $\mathcal{R}_\alpha$.

Theorem 3.2.1. *For any non-trivial and non-binary tree T with root t_1, there exist a unique increasing sequence*

$$0 = \alpha_1 < \ldots < \alpha_K, \quad K \in \mathbb{N}^\star,$$

and a unique sequence of nested subtrees of T

$$T = T_0 \succeq T_1 \succ \ldots \succ T_K = \{t_1\},$$

such that for every $1 \leq k < K$,

$$\forall\, \alpha \in [\alpha_k; \alpha_{k+1}[\quad T_k \;=\; \underset{T \preceq T_{\max}}{\operatorname{argmin}}\, \mathcal{R}_\alpha(T),$$

and

$$\forall\, \alpha \geq \alpha_K \quad T_K \;=\; \underset{T \preceq T_{\max}}{\operatorname{argmin}}\, \mathcal{R}_\alpha(T).$$

This result ensures that we only have to find the subtrees that minimises $R_{\alpha_k}(T)$ for $k = 1, \ldots, K$. In the sequence, trees are embedded. This means that the sequence is obtained by pruning iteratively some branches of the tree T. More precisely, for $k = 1, \ldots, K$, the k-th subtree T_k in the sequence is obtained by removing the branches of T_{k-1} that produce the smallest reduction in prediction error $\mathcal{R}$. These branches are called the weakest links and the value of the minimal reduction in prediction error induced by the weakest links corresponds to α_k, the k-th element of the increasing sequence of values of α. In this way, the sequence is extracted with a reasonable time-complexity.

Moreover, the sequence contains all the information in the sense that for each α, the smallest optimal subtree with respect to $\mathcal{R}_\alpha$ belongs to the created sequence.

Finally, the algorithm selects the final tree $T_{\hat{k}}$ by picking from the sequence $\{T_1, \ldots, T_K\}$ the subtree with the smallest prediction error $\mathcal{R}(T, \mathcal{V}_m)$ on the validation sample $\mathcal{V}_m$, that is

$$\hat{k} = \underset{k=1,\ldots,K}{\operatorname{argmin}} \mathcal{R}(T_k, \mathcal{V}_m).$$

Figure 3.2 provides an example of a sequence of optimal subtrees. The computation of this sequence is fully detailed in Section 3.6.3.

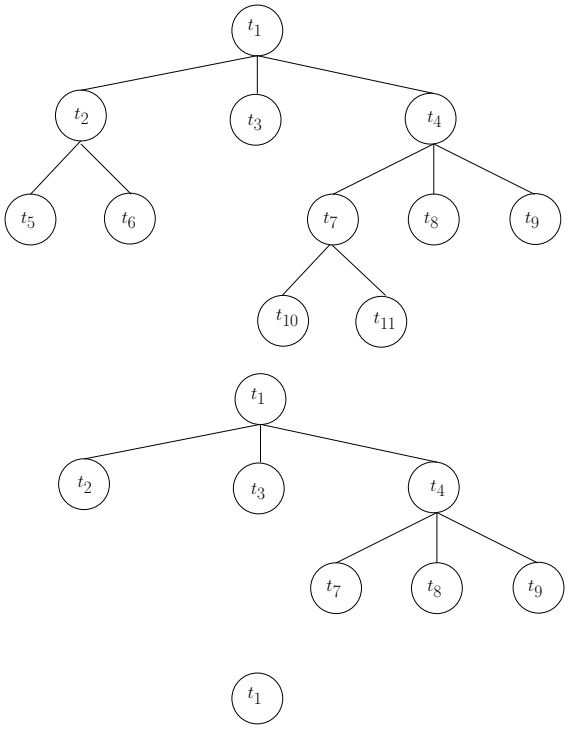


Figure 3.2: The sequence $\{T_1, T_2, T_3\}$ of optimal subtrees of a maximal CARTGV tree $T_{\max}$. From top to bottom: $T_1 = T_{\max}$, $T_2 = T_1 \setminus \{T_{1,t_2} \cup T_{1,t_7}\}$ and $T_3 = T_2 \setminus T_{2,t_1} = \{t_1\}$.

3.2.4 Group importance measures

In supervised classification problems involving grouped inputs, often only a few number of groups is relevant for predicting the response variable. Moreover, quantifying the importance of each group of inputs is useful to perform group variable selection and data interpretation (Gregorutti et al., 2015; Chakraborty & Pal, 2008). CARTGV can be used to rank the importance of groups of inputs via two new measures of importance. These two scores are based on a related CARTGV tree and the decrease in node impurity. The first score, called Group Rand Importance measure (GRI) and denoted by $\mathcal{I}^R_{\mathsf{cartgv}}$, is based on the total decrease in node impurity "corrected" by the Rand index (Rand, 1971). The second one, named Group Surrogate Importance (GSI) and denoted by $\mathcal{I}^S_{\mathsf{cartgv}}$, uses an extension of the notion of surrogate splits proposed by Breiman et al. (1984) to the context of groups of inputs (see also section 2.1 in Introduction for a description of the notion of surrogate split in CART).

Consider a CARTGV tree T and a group $\mathbf{X}^j$, $j = 1, \ldots, J$. The GRI of group j is the "corrected" sum, over all non-terminal nodes of T, of the penalized decrease in node impurity from splitting on group j,

$$\mathcal{I}^R_{\mathsf{cartgv}}(j, T) \;=\; \sum_{t \,\in\, T \setminus \widetilde{T}} \Delta_p \mathrm{Q}(j, t) p(j^\star_t, j), \tag{3.6}$$

where $\Delta_p \mathrm{Q}(j, t)$ is the penalized decrease in impurity (3.3) from splitting t on group j, $j^\star_t$ is the index of the group selected to split node t (see Step 2 in Section 3.2.2) and $p(j^\star_t, j)$ is a *correction*. The parameter $p(j^\star_t, j)$, which is the Rand index, quantifies the agreement between the split of t based on group j and the one based on $j^\star_t$ and it is defined by

$$p(j^\star_t, j) \;=\; \frac{A + B}{\binom{n_t}{2}},$$

where

- $\binom{n_t}{2}$ is the total number of pairs of observations in t,
- A is the number of pairs of observations in t that fall into the same child node with both the split based on group j and the split based on group $j^\star_t$,
- B is the number of pairs of observations in t that fall into different child nodes with both the split based on group j and the split based on group $j^\star_t$.

$p(j^\star_t, j)$ lies between 0 and 1. It takes the value 1 if the two splits are identical, in the sense that the two splits partition the node t into the same child nodes. This quantity is used to prevent overestimating the importance of groups which are weakly correlated with both the

relevant groups and the response variable (see chap. 5, Breiman et al., 1984).

The GSI relies on an extension of the notion of surrogate splits to the context of groups of inputs. Consider a node t and recall that $j_t^\star$ is the index of the group selected to split node t and $\tilde{t}(j_t^\star)$ is the partition of t induced by splitting tt on group $j_t^\star$. In the context of grouped inputs, we define the surrogate split of t based on group j as the split of t based on group j that is as close as possible to the split t based on group $j_t^\star$ (in the sense that the two splits send observations in node t into similar child nodes). It is denoted by $\tilde{t}^s(j)$ and we propose to estimate it as follows. First of all, a splitting tree of maximal depth D_j is built on node t based on group X^j and by using the response variable $\widetilde{Y}_{j_t^\star}$. The random variable $\widetilde{Y}_{j_t^\star}$ takes values in the set $\{1, \ldots, |\tilde{t}(j_t^\star)|\}$, where we recall that $|\tilde{t}(j_t^\star)|$ denotes the cardinality of $\tilde{t}(j_t^\star)$. $\widetilde{Y}_{j_t^\star}$ refers to the index of the child node in which an observation in t falls with the split based on $j_t^\star$. Then, we define the surrogate split $\tilde{t}^s(j)$ of t based j as the partition induced by the splitting tree grown on group j and the variable $\widetilde{Y}_{j_t^\star}$. Next, the penalized decrease in node impurity induced by the surrogate split $\tilde{t}^s(j)$ is computed:

$$\Delta_p \mathrm{Q}^S(t, j) = \mathsf{pen}(d_j) \Delta \mathrm{Q}^S(t, j),$$

with $\Delta \mathrm{Q}^S(t, j) = n_t \mathrm{Q}(t) - \sum_{t' \in \tilde{t}^s(j)} n_{t'} \mathrm{Q}(t')$ referring to the decrease in node impurity from using the surrogate split $\tilde{t}^s(j)$. Then we define the GSI of group j as the sum, over all non-terminal nodes of T, of the penalized decrease in node impurity from using the surrogate splits based on group j,

$$\mathcal{I}^S_{\mathsf{cartgv}}(j, T) \;=\; \sum_{t \,\in\, T \setminus \widetilde{T}} \Delta_p \mathrm{Q}^S(j, t). \tag{3.7}$$

The two group importance measures are normalized to a scale between 0 and 100:

$$\widetilde{\mathcal{I}}^R_{\mathsf{cartgv}}(j, T) = 100 \times \frac{\mathcal{I}^R_{\mathsf{cartgv}}(j, T)}{\max\limits_{j'=1,\ldots,J} \mathcal{I}^R_{\mathsf{cartgv}}(j', T)},$$

and

$$\widetilde{\mathcal{I}}^S_{\mathsf{cartgv}}(j, T) = 100 \times \frac{\mathcal{I}^S_{\mathsf{cartgv}}(j, T)}{\max\limits_{j'=1,\ldots,J} \mathcal{I}^S_{\mathsf{cartgv}}(j', T)}.$$

In this way, for any measure GRI or GSI, groups whose the importance is equal to 100 are considered as the most relevant. These two scores induce an order of importance. Indeed, groups with the highest values of importance are considered as the most relevant groups to predict the response variable Y. The group importance measures GRI and GSI are assessed in the next section.

3.2.5 Illustrations

This section is devoted to highlighting the good behavior of the two group importance measures GRI and GSI and to justifying the role of the CARTGV parameters. It also provides an illustration of CARTGV and offers a first insight of its performances. Performances of CARTGV will be more extensively assessed through other simulated models and in comparison with other methods in Section 3.4.
For this first illustration, CARTGV is compared to CART through several numerical experiments based on the synthetic model introduced by Weston et al. (2003) (see also Genuer et al., 2010)). We extend this model to the context of groups of inputs. CART is implemented with the `R` package `rpart`.

Description of the simulated data set

We consider the following equiprobable two-classes problem. Y is the response variable which takes value in $\{-1, 1\}$ and $\mathbf{X}$ is the vector of inputs which is structured into J groups, that is $\mathbf{X} = (\mathbf{X}^1, \ldots, \mathbf{X}^J)$. For any $j = 1, \ldots, J$, the group $\mathbf{X}^j = (X_1^j, \ldots, X_{d_j}^j)$ includes d_j variables identically distributed and defined as follows:

$$X_\ell^j \sim \mathcal{N}(z_j, 1). \text{ for any } \ell = 1, \ldots, d_j,$$

where z_j is the realization of the random variable Z_j defined conditionally to the value of the standard uniform random variable U:

$$\mathcal{L}(Z_j \mid Y = y, U = u) = \begin{cases} \mathcal{N}\left(y\frac{j}{3}, 1\right) & \text{if } u \leq 0.7 \text{ and } j = 1, 2, 3, \\ \mathcal{N}\left(y\frac{j-3}{3}, 1\right) & \text{if } u > 0.7 \text{ and } j = 4, 5, 6, \\ \mathcal{N}(0, 1) & \text{otherwise.} \end{cases}$$

Thus, there are six true predictive groups, all with different discriminative power: highly ($\mathbf{X}^3$ and $\mathbf{X}^6$), moderately ($\mathbf{X}^2$ and $\mathbf{X}^4$) and weakly ($\mathbf{X}^1$ and $\mathbf{X}^4$) correlated with the response Y. The other groups, namely groups $\mathbf{X}^7 - \mathbf{X}^J$, are noise.

Groups are independent. Within each group $\mathbf{X}^j$, $j = 1, \ldots, J$, the covariance between two inputs X_ℓ^j and $X_{\ell'}^j$ $(\ell, \ell' = 1, \ldots, d_j)$ is defined as:

$$\text{Cov}\left(X_\ell^j, X_{\ell'}^j\right) = 0.8^{|\ell - \ell'|}.$$

Remark 3.2.3. *In this model, the group structure comes mainly from both the block structure of the covariance matrix of the input vector X and the discriminative power of the inputs.*

We simulate $n = 600$ i.i.d copies of the random vector $(\mathbf{X}, Y)$ from the model described above. Inputs are next standardized to have zero mean and unit variance.

Based on this synthetic model, we conduct the following four experiments:

- Experiment 1: (easy case)

  - $J = 12$,
  - $d_j = 10$ for any $j = 1, \ldots, 12$.
- Experiment 2: (many noisy groups)
  - $J = 56$,
  - $d_j = 10$ for any $j = 1, \ldots, 56$.
- Experiment 3: (a large noisy group*)
  - $J = 13$,
  - $d_j = \begin{cases} 10 & \text{if } j = 1, \ldots, 12, \\ 100 & \text{if } j = 13. \end{cases}$.
- Experiment 4: (a large noisy group* and inclusion of 10 noisy variables in the most relevant group*)
  - $J = 13$,
  - $d_j = \begin{cases} 20 & \text{if } j = 3, \\ 100 & \text{if } j = 13, \\ 10 & \text{otherwise.} \end{cases}$

($\star$) *noisy variables are independent standard Gaussian variables.*

The first two experiments are specially used to highlight the good behavior of the two group importance measures GRI and GSI. It also offers a first insight of the performances of CARTGV. The two last experiments are used to justify the use of the penalty function in the impurity criterion (3.3).

For each data set, observations are randomly divided into a training sample $\mathcal{D}_n$, a validation sample $\mathcal{V}_m$ and a test sample $\mathcal{T}_q$ of equal size (i.e. 200 observations in each set). For each method, the maximal tree is first built on the training set $\mathcal{D}_n$. Then maximal trees are pruned by using the minimal cost-complexity pruning algorithm and the validation set $\mathcal{V}_m$ (a detailed description of this pruning strategy is introduced in Section 3.2.3). The test sample $\mathcal{T}_q$ is used to evaluated the predictive performance of each method.

In CARTGV, the validation set is also used to select the tuning parameters, namely the maximal depth of the splitting trees D_j and the penalty function pen. We consider two values for D_j (2 and 3) and the four following penalty functions:

$$\begin{aligned} \mathsf{pen}(d_j) &= 1 \qquad \text{(no penalty)}, \\ \mathsf{pen}(d_j) &= 1/d_j \qquad \text{(the "size" penalty)}, \\ \mathsf{pen}(d_j) &= 1/\sqrt{d_j} \quad \text{(the "root" penalty)}, \\ \mathsf{pen}(d_j) &= 1/\max(\log d_j, 1) \quad \text{(the "log" penalty)}. \end{aligned}$$

The tuning parameters are selected as follows. For each pair of values, a maximal tree is first built on the training set $\mathcal{D}_n$. Each maximal tree is next pruned by using the validation set $\mathcal{V}_m$ and the cost-complexity strategy described in Section 3.2.3. Thus denoting by $T^\star(\nu_{D_j}, \nu_{\mathsf{pen}})$ the pruned tree for $D_j = \nu_{D_j}$ and $\mathsf{pen} = \nu_{\mathsf{pen}}$, we select the couple $(\hat{\nu}_{D_j}, \hat{\nu}_{\mathsf{pen}})$ that minimizes the prediction error $\mathcal{R}(T^\star(\nu_{D_j}, \nu_{\mathsf{pen}}), \mathcal{V}_m)$ on the validation sample $\mathcal{V}_m$. Note that when all groups have the same size, as for instance in the first two scenarios, there is no need to use a penalty function and we thus set $\mathsf{pen}(d_j) = 1$.

All results of each experiment are averaged over 50 independent samples.

Illustration of CARTGV in comparison with CART

The performances of CARTGV and CART are assessed according to two criteria: the predictive accuracy of the classification rule and the ability to identify the true relevant groups. The first criterion is evaluated by the area under the curve (AUC) and the misclassification rate (Error) on the test set $\mathcal{T}_q$ whereas the ability to identify the relevant groups is measured by group selection frequencies. For CART, which ignores the group structure, we define the selection frequency of a group, as the number of times that at least one input in the group is included in the final tree. For CARTGV, the selection frequency of a group is defined as the number of times that the group is included at least once in the final tree.

Table 3.1 and Figure 3.6 display the results for the first two experiments. Table 3.1 provides the mean and standard deviation of the AUC, the misclassification rate (Error), the tree depth (Depth) and the number of leaves (Leaves) of final trees. In these two experiments, CARTGV and CART perform well and shown similar predictive performances. In the first experiment, we can note that CART slightly outperforms CARTGV. In the first two experiments, CARTGV trees are larger and less deep than CART trees, which can be explained by the use of non-binary splits in CARTGV. Nonetheless, note that this does not mean that CARTGV trees are less easily understandable since CARTGV takes into account the input group structure, which may make more sense than the inputs taken individually. The two methods do not seem to be very sensitive to the inclusion of many noisy groups. Indeed, the predictive performance of CARTGV and CART are not affected when the number of noisy groups strongly increases (see experiment 2).

Figure 3.4 displays group selection frequencies for CARTGV and CART in the first two experiments. For the sake of visibility and since selection frequencies of the noisy groups are similar, we only plot selection frequencies of the first twenty groups in experiment 2. In the two experiments, CARTGV and CART correctly identify the most relevant groups, even when there are a lot of noisy groups. Selection frequency of a group behaves as an increasing function of the relevance of the group. Moreover, although the definition of the selection frequency of a group for CARTGV is different from that of CART, it may seem that CART tends to select noisy variables more frequently than CARTGV.

	CARTGV	CART
Experiment 1		
AUC	0.646 (0.044)	0.660 (0.040)
Error	0.366 (0.039)	0.350 (0.040)
Depth	2 (0.857)	4 (1.626)
Leaves	7 (3.969)	7 (5.009)
Experiment 2		
AUC	0.657 (0.053)	0.660 (0.040)
Error	0.359 (0.04)	0.350 (0.040)
Depth	2 (0.728)	3 (1.232)
Leaves	7 (3.697)	4 (2.056)

Table 3.1: Performances of CARTGV and CART in the first two experiments. Depth and Leaves respectively denote the tree depth and the number of leaves of the pruned tree. For each criterion, the mean value is given following by the value of the standard deviation in brackets.

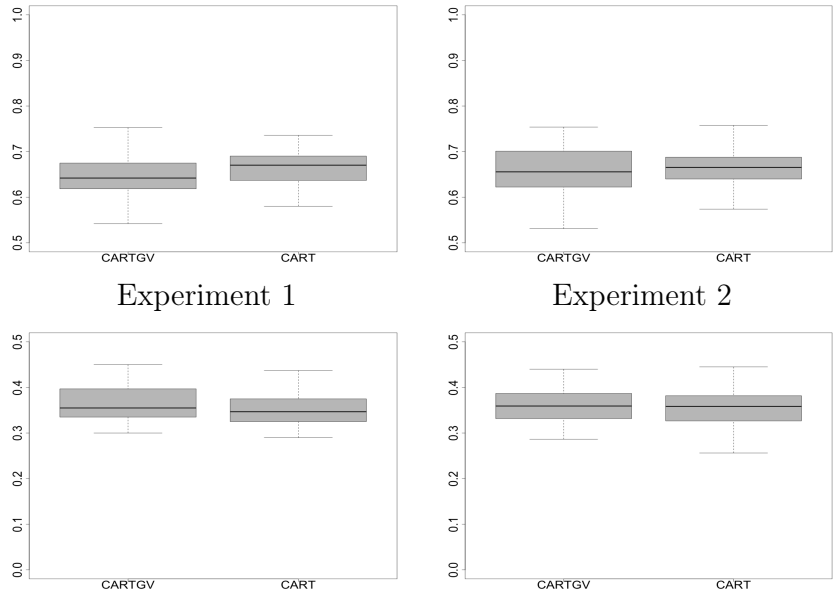


Figure 3.3: Performances of CARTGV and CART: distribution of AUC (top) and Error (bottom) in the first two scenarios.

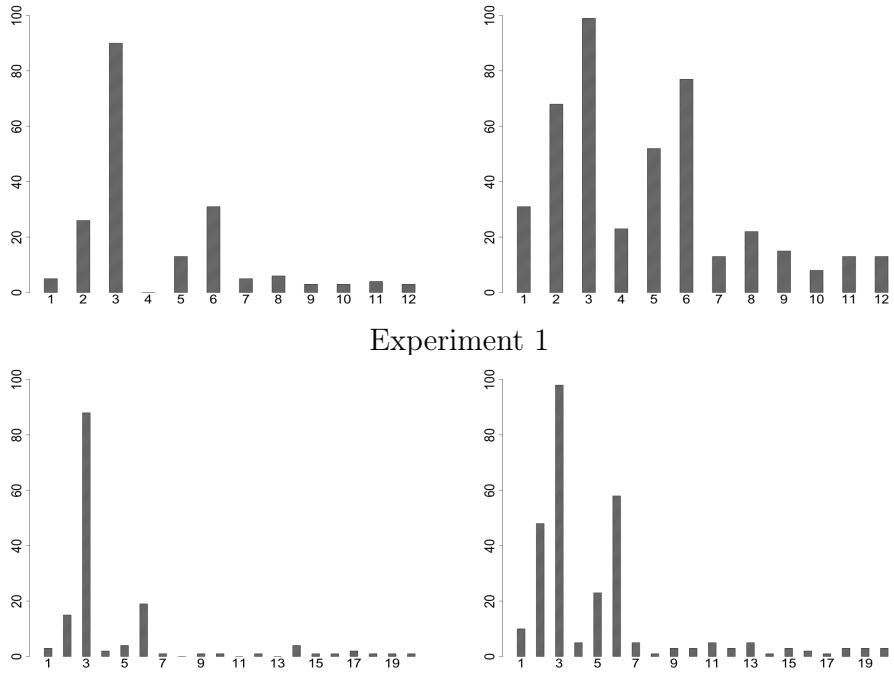


Figure 3.4: Performances of CARTGV and CART: group selection frequencies in the first two experiments for CARTGV (left) and CART (right). We recall that only the first six groups are relevant (in this order: 3, 6, 2, 4, 1, 3).

Remark 3.2.4. *The similar predictive performances of CARTGV and CART can be explained by the fact that in this synthetic model, the group structure is mainly used to "control" the correlation between the input variables. The relationship between the response variable and the inputs can be then well estimated by ignoring the group structure. In section 3.4, CARTGV and CART will be extensively assessed and compared through other simulated models in which the relationship between the response variable and the inputs is genuinely defined based on groups of inputs.*

Assessment of the group importance measure

The performances of the group importance measures GRI and GSI are also investigated. Table 3.2 displays, for the two measures, the percentage of times that the true relevant groups are part of the six groups with the highest importance. Figure 3.5 displays the distributions of the group importance in the first two experiments. As previously, in experiment 2, we plot the group importance for the first twenty groups only.

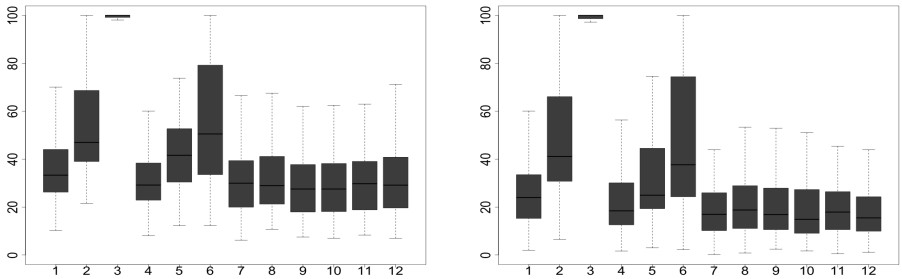
Overall, in the first two experiments, the two measures of group importance well identify and perfectly rank the true relevant groups. Moreover, the distributions of the group importance

are very similar for the two scores.

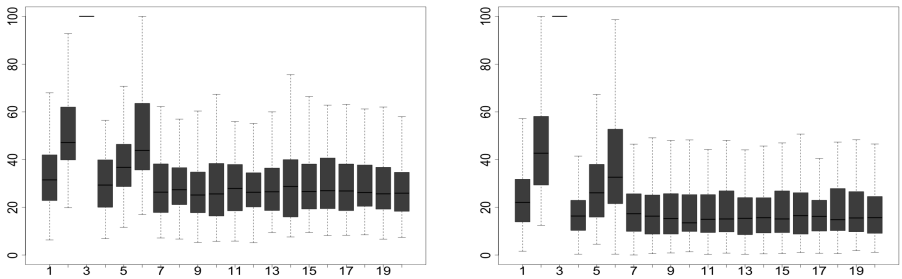
We can notice that the two scores identify rarely all the true predictive groups. That may be explained by the lower discriminatory power of the first and the fourth groups.

	GRI	GSI
Experiment 1		
Identification of the most relevant group	75	75
Selection of at least 3 relevant groups	100	100
Selection of at least 4 relevant groups	88	90
Selection of at least 5 relevant groups	47	49
Experiment 2		
Identification of the most relevant group	79	84
Selection of at least 3 relevant groups	75	72
Selection of at least 4 relevant groups	30	26
Selection of at least 5 relevant groups	4	3

Table 3.2: Top 6 groups with the highest importance for the group importance measure GRI and GSI. The table gives the percentage of times that the relevant groups are part of the 6 groups with the highest importance.



Experiment 1



Experiment 2

Figure 3.5: Distribution of the group importance for GRI (left) and GSI (right) in the first two experiments.

Choice of the tuning parameters

In this section, we discuss the choice of the tuning parameters $\mathsf{pen}(d_j)$ and D_j introduced in Section 3.2.2.

Choice of the penalty function pen

Experiment 3 and 4 are used to investigate the choice of the penalty function pen. For each experiment, CARTGV is performed using each of the following four functions:

$$\begin{aligned}
\mathsf{pen}(d_j) &= 1 \qquad \text{(no penalty)},\\
\mathsf{pen}(d_j) &= 1/d_j \qquad \text{(the "size" penalty)},\\
\mathsf{pen}(d_j) &= 1/\sqrt{d_j} \quad \text{(the "root" penalty)},\\
\mathsf{pen}(d_j) &= 1/\max(\log d_j, 1) \quad \text{(the "log" penalty)}.
\end{aligned}$$

Simulation results are summarized in Tables 3.3, 3.4 and 3.5. Figures 3.7 and 3.8 display group selection frequencies according to the penalty function in experiments 3 and 4. The figures showing the distribution of the group importance for GRI and GSI are provided in Section 3.6.4 (see Figures 3.17, 3.18, 3.19 and 3.20). Note that in experiment 4, since noisy variables are included in the third group, claiming that this group is still the most relevant group is questionable.

In these two experiments, the predictive performances of CARTGV do not seem to be very sensitive to the penalty function (excepted in experiment 4 with $\mathsf{pen}(d_j) = 1/d_j$). The choice of pen seems rather to affect group selection frequencies and the measure of the group importance. Indeed, in experiment 3 and 4, when no penalty is used (i.e. $\mathsf{pen}(d_1) = 1$), the large noisy group is often chosen to build the tree and its importance is high (for both GSI and GRI). On the contrary, when a penalty function is used, this group is significantly less frequently selected and its importance is low (for both GSI and GRI). Overall, it seems that using a penalty function allow to improve the ability of CARTGV to identify and select the true relevant groups of inputs. Obviously, note that if all groups have the same size, as for instance in the first three scenarios, there is no need to use a penalty function.

Generally, the choice of the penalty function is highly dependent on the data. Indeed, if it is expected that the noise is mostly included in the largest groups, as in experiment 3, then the penalty $\mathsf{pen}(d_j) = 1/p_j$ would be preferable. Otherwise, this penalty function may appear too strong and other penalties, such as $\mathsf{pen}(d_j) = 1/\sqrt{d_j}$ or $\mathsf{pen}(d_j) = 1/\max(\log d_j, 1)$, may perform better. For instance in experiment 4, the third group, which includes ten highly relevant variables and ten independent noisy variables, is rarely selected and its importance is much lower when using $\mathsf{pen}(d_j) = 1/d_j$. Then, in this experiment, using either $\mathsf{pen}(d_j) = 1/\sqrt{d_j}$ or $\mathsf{pen}(d_j) = 1/\max(\log d_j, 1)$ seems preferable.

Penalty function	1	$1/d_j$	$1/\sqrt{d_j}$	$1/\max\{\log d_j, 1\}$
Experiment 3				
AUC	0.654 (0.045)	0.662 (0.049)	0.657 (0.046)	0.654 (0.050)
Error	0.359 (0.041)	0.35 (0.041)	0.353 (0.038)	0.355 (0.040)
Depth	2 (0.776)	2 (0.918)	2 (0.958)	2 (1.018)
Leaves	8 (5.018)	9 (5.893)	10 (6.484)	10 (6.594)
Experimentt 4				
AUC	0.64 (0.050)	0.612 (0.057)	0.628 (0.049)	0.632 (0.052)
Error	0.372 (0.045)	0.398 (0.045)	0.379 (0.040)	0.376 (0.044)
Depth	2 (0.814)	2 (0.978)	2 (1.076)	2 (1.059)
Leaves	7 (4.756)	11 (9.234)	11 (6.779)	10 (6.655)

Table 3.3: Performances of CARTGV according to the penalty function pen. Depth and Leaves respectively denote the tree depth and the number of leaves of the pruned tree. For each criterion, the mean value is given following by the value of the standard deviation in brackets.

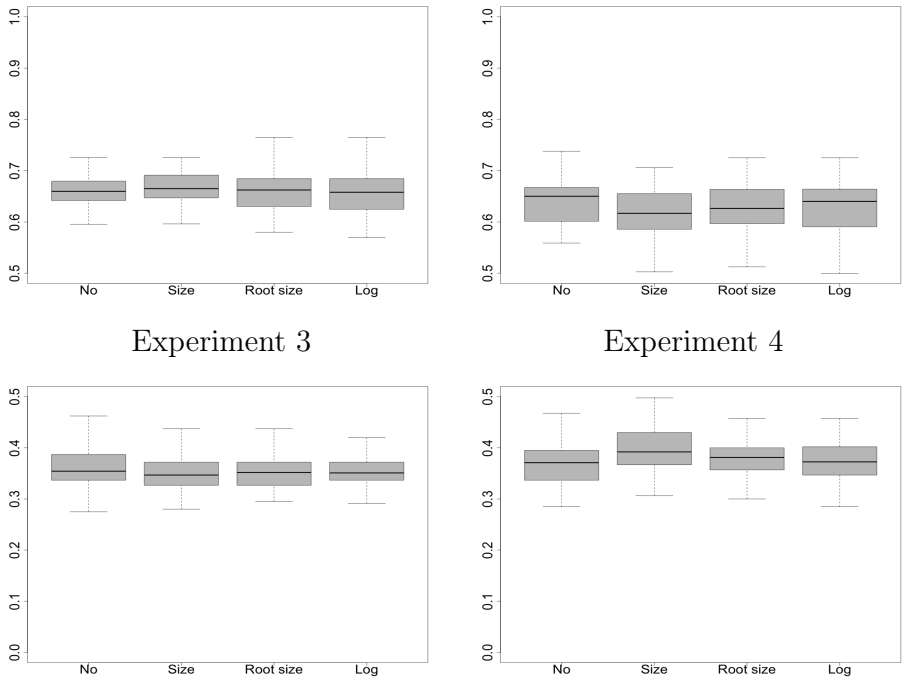


Figure 3.6: Performances of CARTGV according to the penalty function pen: distribution of AUC (top) and Error (bottom) in the last two scenarios.

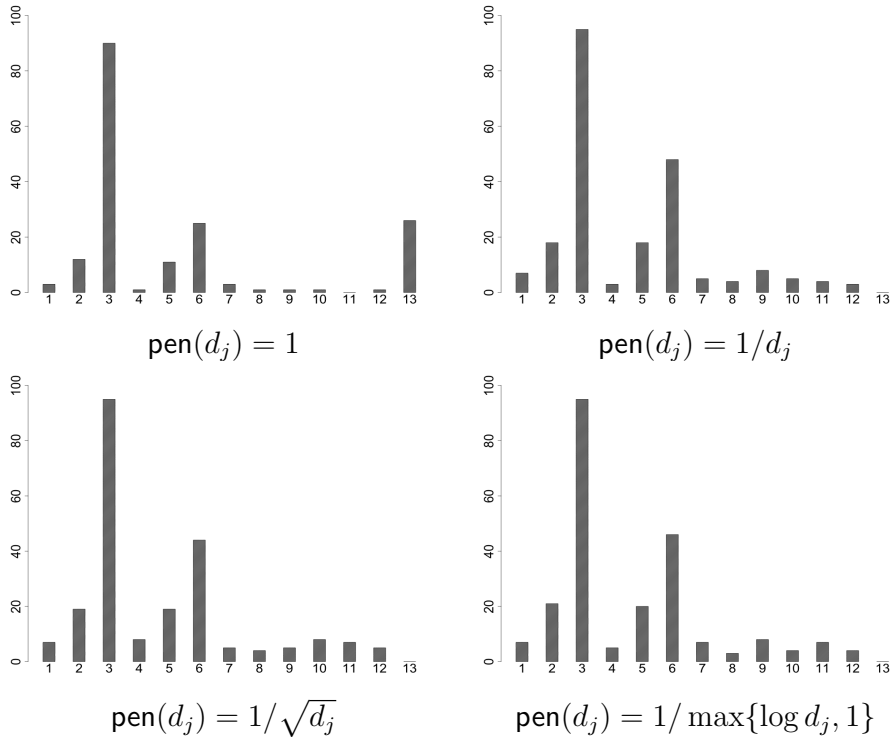


Figure 3.7: CARTGV group selection frequencies for each penalty function, in experiment 3. We recall that only the first six groups are relevant (in this order: 3, 6, 2, 4, 1, 3).

Penalty function	1	$1/d_j$	$1/\sqrt{d_j}$	$1/\max\{\log d_j, 1\}$
Experiment 3				
Identification of the most relevant group	74	76	75	75
Selection of at least 3 relevant groups	99	100	100	100
Selection of at least 4 relevant groups	87	94	91	91
Selection of at least 5 relevant groups	29	49	49	48
Selection of the 6 relevant groups	0	4	4	4
Experiment 4				
Identification of the most relevant group	91	4	47	57
Selection of at least 3 relevant groups	99	99	99	100
Selection of at least 4 relevant groups	81	87	93	91
Selection of at least 5 relevant groups	30	44	54	51
Selection of the 6 relevant groups	1	5	8	6

Table 3.4: Top 6 groups with the highest importance for the group importance measure GRI according to the penalty function. The table gives the percentage of times that the relevant groups are part of the 6 groups with the highest importance.

Penalty function	1	$1/d_j$	$1/\sqrt{d_j}$	$1/\max\{\log d_j, 1\}$
Experiment 3				
Identification of the most relevant group	81	82	83	82
Selection of at least 3 relevant groups	97	100	100	100
Selection of at least 4 relevant groups	76	89	85	87
Selection of at least 5 relevant groups	28	42	42	42
Selection of the 6 relevant groups	1	4	3	4
Experiment 4				
Identification of the most relevant group	89	3	43	56
Selection of at least 3 relevant groups	100	98	100	99
Selection of at least 4 relevant groups	81	79	87	87
Selection of at least 5 relevant groups	35	36	51	45
Selection of the 6 relevant groups	1	2	2	0

Table 3.5: Top 6 groups with the highest importance for the group importance measure GSI according to the penalty function. The table gives the percentage of times that the relevant groups are part of the 6 groups with the highest importance.

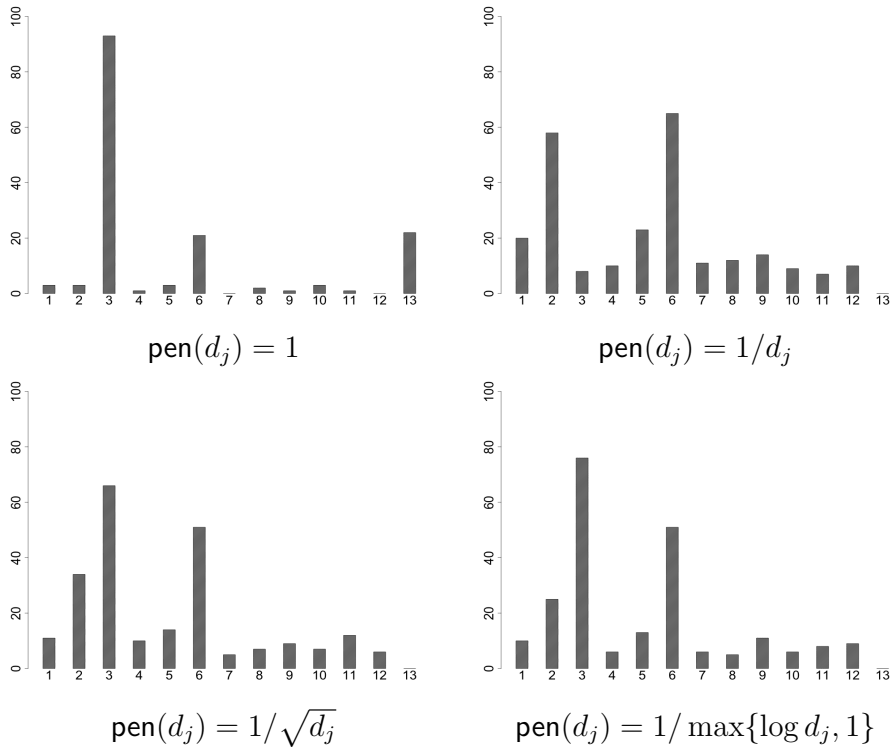


Figure 3.8: CARTGV group selection frequencies for each penalty function, in experiment 4. We recall that only the first six groups are relevant (in this order: 3, 6, 2, 4, 1, 3).

Choice of the maximal depth of the splitting trees D_j

We now consider the problem of selecting the tuning parameter D_j, which corresponds to the maximal depth of the splitting trees based on group j (see Section 3.2.2). Generally, the "optimal" value of D_j is unknown and need to be selected.

Here, several values for D_j are assessed through numerical simulations. Then, we provide some guidelines for the calibration of D_j. For simplicity, the discussion is based on experiment 1, where groups have the same size, and we set $D = D_j$, for any $J = 1, \ldots, J$. Table 3.6 shows the performances of CARTGV for several values of D. Results with $D = 1$ are not displayed in this table since CARTGV with $D = 1$ is equivalent to CART.

Results given in Table 3.6 supports the idea of choosing small values for D. First, taking a small value for D prevents from building too complex splits which can be prone to overfitting. This is illustrated in Table 3.6 for $D = 10$. In this case, at least 25% of the pruned trees are trivial. Furthermore, small values of D_j do not prevent from modeling complex relationships between group the $\mathbf{X}^j$ and the response variable, since the algorithm can consecutively choose the same group during the tree-growing phase. Thus, although the choice of D_j is strongly dependent on the data, we suggest taking $D_j = 2$ or 3, for any $j = 1, \ldots, J$. Furthermore, when group sizes are similar, setting $D_j = D$, for any $j = 1, \ldots, J$, can have two advantages. First, this reduces the number of parameters that need to be tuned. Secondly, using different values for all groups can foster groups with a large D_j.

D	2	3	5	10
AUC	0.653 (0.05)	0.621 (0.051)	0.579 (0.054)	0.536 (0.045)
Error	0.355 (0.038)	0.381 (0.04)	0.422 (0.047)	0.462 (0.045)
Depth	2 (1.104)	2 (0.843)	1 (0.567)	1 (0.431)
Leaves	9 (5.38)	14 (7.629)	24 (8.342)	33 (18.648)

Table 3.6: Performances of CARTGV according to D in Experiment 1. Depth and Leaves respectively denote the tree depth and the number of leaves of the pruned tree. For each criterion, the mean value is given following by the value of the standard deviation in brackets.

Overall, these initial experimental results offer a first insight of the method CARTGV and provide some guidelines about the calibration of the CARTGV parameters. The method does not require any assumption about the data and it elaborates easily understandable prediction rules when inputs are grouped. Moreover, CARTGV automatically provides two importance measures for each predictor group, which is useful to perform group variable selection. However, as for CART, CARTGV trees suffer from instability. Small perturbations

in the training sample can considerably change the prediction rule. For this reason, we propose a new random forest method adapted to grouped variables.

3.3 Random forests for grouped variables

3.3.1 Principles of the random forests for grouped variables

Random forests for grouped variables (RFGV) is a classification and regression method which consists in aggregating a large number of such random decision trees, in the same way that Breiman's RF (Breiman, 2001).

Consider the statistical framework introduced in Section 3.2.1. First, RFGV generates `ntree` bootstrap samples $\mathcal{D}_n^1, \ldots, \mathcal{D}_n^{\texttt{ntree}}$ from the training sample $\mathcal{D}_n$. Next, the method builds a decision tree on each bootstrap sample by applying a variant of CARTGV. Indeed, the elaboration of the `ntree` trees differs from CARTGV in two main aspects. This modified tree growing process is now described.

At each node t, the method chooses uniformly `mgrp` groups at random among the J ones. Let $\mathcal{J}$ denote the `mgrp`-tuple of indices of the selected groups. Then, the algorithm performs the following two steps.

- Step 1: Choice of a split for each selected group.
  For any $j \in \mathcal{J}$, the algorithm finds a split with respect to a variant of the splitting-tree growing process described in **Step 1** in Section 3.2.2. Indeed, the algorithm builds a splitting tree of maximal depth D_j by using variable resampling. Formally, at each step of the splitting-tree growing process, the algorithm selects only a small number ($\texttt{mvar}_j$) of variables in the group j and then selects the best split based on this subset of inputs. This process is recursively repeated until the maximal depth of the splitting tree D_j is reached.

- Step 2: Choice of the splitting group in the set $\mathcal{J}$ of selected groups.
  The algorithm selects the split that maximizes the penalized criterion, defined in equation (3.3).

This growing process is then repeated on each node until a fully grown tree is obtained. This maximal tree is not pruned.

Let $T^1, \ldots, T^{n_{tree}}$ denote the `ntree` resulting trees and let $\hat{f}(, \Theta^1), \ldots, \hat{f}(, \Theta^{\texttt{ntree}})$ denote the associated prediction rules. $\Theta^1, \ldots, \Theta^{\texttt{ntree}}$ are `ntree` independent copies of a random variable

Θ. The random variable Θ is independent of $\mathcal{D}_n$ and refers to the bootstrap sampling and the splitting subgroups.

The random forest, that we denote by $\{T^b\}_1^{\texttt{ntree}}$, is the aggregation of the `ntree` trees. It defines the prediction rule $\hat{f}_{\mathsf{rfgv}}$ which is based on the average of the prediction rules $\hat{f}(,\Theta^1),\ldots,\hat{f}(,\Theta^{ntree})$. Consider an observation $\mathbf{x}\in\mathbb{R}^d$. In regression, the prediction $\hat{f}_{\mathsf{rfgv}}(\mathbf{x})$ of $\mathbf{x}$ is defined by the average of the predictions, that is

$$\hat{f}_{\mathsf{rfgv}}(\mathbf{x}) = \frac{1}{\texttt{ntree}}\sum_{b=1}^{\texttt{ntree}}\hat{f}(\mathbf{x},\Theta^b),$$

while in classification, $\hat{f}_{\mathsf{rfgv}}(\mathbf{x})$ refers to the majority vote

$$\hat{f}_{\mathsf{rfgv}}(\mathbf{x}) = \underset{k=1,\ldots,K}{\operatorname{argmax}}\left(\sum_{b=1}^{\texttt{ntree}}\mathbb{1}_{\hat{f}(\mathbf{x},\Theta^b)=k}\right).$$

Algorithm 4 describes in detail how to build a RFGV forest.

Algorithm 4 Random Forest for Grouped Variables.

Input: Training set $\mathcal{D}_n$, $\texttt{ntree} \in \mathbb{N}^\star$, $a_n \in \{1, \ldots, n\}$, $\texttt{mgrp} \in \{1, \ldots, J\}$, $\texttt{nmin} \in \{1, \ldots, n\}$, $\texttt{mvar}_j \in \{1, \ldots, d_j\}$, $D_j \in \mathbb{N}^\star$, for all $j = 1, \ldots, J$.

For $b = 1, \ldots, \texttt{ntree}$ **do**

1. Construction of the bootstrap sample $\mathcal{D}_n^b$: select a_n observations, with (or without) replacement, uniformly in $\mathcal{D}_n$.

2. Construction of the tree $\mathcal{D}_n^b$: repeat recursively the following process on each terminal node of the tree until the minimum node size (denoted `nodesize`) is reached:
   Let's consider the terminal node t.

   (a) Select uniformly, without replacement, a subset $\mathcal{J} \subset \{1, \ldots, J\}$ of cardinality `mgrp`.

   (b) **For** $j \in \mathcal{J}$ **do**
   Choice of a split for group j: grow a *splitting tree* with root t based on the inputs $X_{j_1}, X_{j_2}, \ldots, X_{j_{d_j}}$ that belong to group j, by repeating recursively the following process on each terminal node of the splitting tree until the maximal tree-depth D_j is reached (*see* ***Step 1*** *in Section 3.2.2 for details)*:

      i. Select uniformly, without replacement, a subset $\mathcal{M}_j \subset \{X_{j_1}, \ldots, X_{j_{d_j}}\}$ of cardinality $\texttt{mvar}_j$.

      ii. Select the best input/splitting point by optimizing the CART-split criterion (3.1) along the subset $\mathcal{M}_j$ of inputs.

      iii. Split the node into two child nodes according to the best split.

      End.

   (c) Pick the best group/splitting tree by optimizing the penalized impurity criterion (3.3) along the coordinates in $\mathcal{J}$ (*see* ***Step 2*** *in Section 3.2.2 for details*).

   (d) Split the node according to the best split.

3. Definition of the prediction rule $\hat{f}(., \Theta^b)$: deduce from the maximal tree T^b the prediction rule $\hat{f}(., \Theta^b)$.

End.

Output: the resulting trees $T^1, \ldots, T^{\texttt{ntree}}$ and the associated prediction rules $\hat{f}(., \Theta^1), \ldots, \hat{f}(., \Theta^{\texttt{ntree}})$.

Prediction of the random forest at $\mathbf{x} \in \mathbb{R}^d$:

In regression: $\hat{f}_{\texttt{rfgv}}(\mathbf{x}) = \frac{1}{\texttt{ntree}} \sum_{b=1}^{\texttt{ntree}} \hat{f}(\mathbf{x}, \Theta^b)$.

In classification: $\hat{f}_{\texttt{rfgv}}(\mathbf{x}) = \underset{k=1,\ldots,K}{\operatorname{argmax}} \left(\sum_{b=1}^{\texttt{ntree}} \mathbb{1}_{\hat{f}(\mathbf{x}, \Theta^b) = k} \right)$.

3.3.2 Details on RFGV parameters

The algorithm has several important parameters. We now provide some guidelines for the choice of these parameters. Most of these suggestions are based on guidelines given for Breiman's RF (Breiman, 2001; Díaz-Uriarte & Alvarez de Andrés, 2006; Bernard et al., 2008; Genuer, 2010; Boulesteix et al., 2012; Biau & Scornet, 2016). The experimental studies introduced in the following section enable to provide further insights.

- The number of trees $\texttt{ntree} \in \mathbb{N}^\star$.
  It is recommended to pick a large `ntree` so that the prediction error of the forest is stable. The default value is $\texttt{ntree} = 500$. This parameter is not a real tuning parameter in the sense that choosing a large value is always preferable.

- The parameters `mgrp` and $\texttt{mvar}_j$.
  The first parameter, `mgrp` , denotes the number of groups randomly samples as candidates at each split for the trees in the RGFV forests. For any $j = 1, \ldots, J$, the parameter $\texttt{mvar}_j$ refers to the number of inputs randomly sampled as candidates at each step for the splitting trees based on group j in the trees of the RFGV forests. The parameters `mgrp` and $\texttt{mvar}_j$ are probably the most important parameters. These parameters can be viewed as the analogues of the parameter `mtry` in Breiman's RF which corresponds to the number of variables selected for splitting at each node of each tree. Their default values are $\texttt{mgrp} = \sqrt{J}$ and $\texttt{mvar}_j = \sqrt{d_j}$ in classification and $\texttt{mgrp} = J/3$ and $\texttt{mvar}_j = d_j/3$ in regression. Note, however, that the best values for these parameters depend heavily on the data. If the number of true relevant groups (respectively true relevant inputs in groups) is expected to be small, default values may appear too small. Moreover, observe that if all groups have approximately the same size, we recommend choosing the same value for any $\texttt{mvar}_j$, $j = 1, \ldots, J$.

- The maximal depth D_j of the splitting tree based on group j, $j = 1, \ldots, J$.
  As for CARTGV, it is preferable to choose a small value for D_j. We recommend taking $D_j = 2$ for any $j = 1, \ldots, J$.

- The penalty function `pen`.
  By default no penalty function is used, i.e. $\texttt{pen}(d_j) = 1$. Nonetheless, a penalty function may be used if the sizes of the groups vary greatly (see Section 3.2.5). Note that if D_j and $\texttt{mvar}_j$ is quite similar in all groups or if all groups have approximately the same size, there is no need to use a penalty function.

- The minimum node size `nodesize`.
  The default value is $\texttt{nodesize} = 1$ in classification and $\texttt{nodesize} = 5$ in regression.

- The number a_n of observations in each bootstrap sample.
  By default $a_n = n$ and observations are drawn with replacement. Nonetheless, other resampling strategies can be used such as subsampling with or without replacement. The effect of the resampling scheme is investigating by Strobl et al. (2007).

In practice, the best values for these parameters will depend on the data.

We recommend taking the same $D_j = D$ and $\texttt{mvar}_j = \texttt{mvar}$ for any $j = 1, \ldots, J$. It is generally a good choice and it allows to reduce the number of parameters that need to be tuned.

Overall, only the parameters $\texttt{mgrp}$, $\texttt{mvar}_j$ need to be tuned. The others can be set to their default value.

As noticed by Biau & Scornet (2016), in RFGV, parameters are easy to tune. No independent validation set is required. Since each tree is built using a bootstrap sample from the original data $\mathcal{D}_n$, the prediction error of each tree can be estimated internally by using the observations that do not belong to the bootstrap sample (these observations define a sample called the out-of-bag (OOB) sample, see the following section). In this way, parameters are adjusted by simply testing several values and selecting the ones which minimize the prediction error of the forest.

3.3.3 Permutation-based group importance measure

Unlike individual decision trees, interpretation of a RFGV random forest is not straightforward. Therefore, quantifying the importance of each predictor group is a solution to improve interpretation of RFGV random forests. To this end, RFGV uses the measure of group importance introduced by Gregorutti et al. (2015). This score is a natural extension of the permutation importance score proposed by (Breiman, 2001). Originally, Gregorutti et al. (2015) introduced it to evaluate the prediction accuracy of each group of variables based on Breman's RF. Here, we proposed to compute this measure of group importance based on RGFV forests. We now recall how to calculate the importance of a group of variables with this score.

For any $b = 1, \ldots, \texttt{ntree}$, consider the b-th bootstrap sample $\mathcal{D}_n^b$ and let $\bar{\mathcal{D}}_n^b$ denote the associated OOB sample which contains the observations in $\mathcal{D}_n$ not included in the b-th bootstrap sample $\mathcal{D}_n^b$, that is $\bar{\mathcal{D}}_n^b = \mathcal{D}_n \setminus \mathcal{D}_n^b$. For any $j = 1, \ldots, J$, let $\bar{\mathcal{D}}_n^{bj}$ be the permuted b-th OOB samples obtained by jointly permuted all the inputs in the j-th group, at random. The measure of importance of the group $\mathbf{X}^j$ is then given by

$$\mathcal{I}_{\text{rfgv}}(\mathbf{X}^j \, , \, \{T^b\}_1^{\texttt{ntree}}) \; = \; \frac{1}{\texttt{ntree}} \sum_{b=1}^{\texttt{ntree}} \mathcal{R}(T^b, \bar{\mathcal{D}}_n^b) - \mathcal{R}(T^b, \bar{\mathcal{D}}_n^{bj}), \tag{3.8}$$

where $\mathcal{R}(T^b, \bar{\mathcal{D}}_n^b)$ and $\mathcal{R}(T^b, \bar{\mathcal{D}}_n^{bj})$ are the prediction error of the tree T^b based respectively on $\bar{\mathcal{D}}_n^b$ and $\bar{\mathcal{D}}_n^{bj}$. (The definition of the prediction error of a tree is given in Section 3.2.3). The measure $\mathcal{I}_{\mathsf{rfgv}}(\mathbf{X}^j\,, \{T^b\}_1^{\mathtt{ntree}})$ is then equals to the increase of the prediction error after breaking the link between the group $\mathbf{X}^j$ and the response variable Y. If randomly permuting values of the j-th group results in a strong increase of the prediction error then $\mathcal{I}_{\mathsf{rfgv}}(\mathbf{X}^j\,, \{T^b\}_1^{\mathtt{ntree}})$ is high and group j is considered as relevant to predict the response variable Y. Conversely, if permuting values of the j-th group does not affect the prediction error then $\mathcal{I}_{\mathsf{rfgv}}(\mathbf{X}^j\,, \{T^b\}_1^{\mathtt{ntree}})$ is close to zero and group j is not considered as important.

As proposed by Gregorutti et al. (2015), this score can be rescaled:

$$\tilde{\mathcal{I}}_{\mathsf{rfgv}}(\mathbf{X}^j\,, \{T^b\}_1^{\mathtt{ntree}}) \;=\; \frac{1}{d_j}\,\mathcal{I}_{\mathsf{rfgv}}(\mathbf{X}^j\,, \{T^b\}_1^{\mathtt{ntree}}). \tag{3.9}$$

This rescaled version allows to take into account group sizes when comparing the importance of groups of different sizes. If two groups have equal importance, then the rescaled version will favor the smallest group. The normalized importance is useful when one wants to perform group variable selection and wants to obtain a sparser set of predictors. The rescaled score of importance is investigated through numerical experiments in the following section.

3.4 Numerical experiments

This section is devoted to the assessment of the performances of RFGV and CARTGV. The two proposed methods are exhaustively benchmarked and compared to CART and RF through several synthetic models in classification. CART and RF are respectively implemented with the `R` packages `rpart` and `randomForests`. Performances of the measures of group importance, namely the permuation-based group importance measure for RFGV and the scores GRI and GSI for CARTGV, are also investigated.

3.4.1 Description of the datasets

We use 4 models of classification problems involving different group structures. For each model, several experiments are considered. Model 1 is the model used to illustrate CARTGV (see Section 3.2.5). Models 2-4 are inspired from Biau et al. (2016) and Biau et al. (2018).

In Model 1, the group structure is mainly used to "control" the correlation between the input variables whereas in Models 2-4 the relationship between the response variable and the inputs is genuinely defined based on the group structure. For Models 2-4, we consider several shapes for the relationship between the variables belonging to a same group. Indeed, in Model 4, the relationship between inputs belonging to a same relevant group is additive, while Models 2 and 3, the relationship between inputs belonging to a same relevant group

includes some interactions. Moreover, note that each model is defined (i.e. choice of the values for the coefficients and the thresholds), so that the distribution of the response variable Y is balanced.

In each model, we consider the following design: Y is the response variable and $\mathbf{X} \in \mathbb{R}^d$ is the vector of inputs structured into J groups, that is $\mathbf{X} = (\mathbf{X}^1, \ldots, \mathbf{X}^J)$. For any $j = 1, \ldots, J$, the vector $\mathbf{X}^j = (X_1^j, \ldots, X_{d_j}^j)$ denotes the group j which includes d_j variables. We let $\mathcal{B}(\pi)$, $\mathcal{U}([a,b]^d)$ and $\mathcal{U}([a,b]^d)$ respectively refer to the standard notation of the Bernouilli distribution of paramteter π, the Uniform distribution over $[a,b]^d$ and the normal distribution with mean μ and variance σ^2.

Model 1. $n = 600$, $Y = 2\,\mathcal{B}(0.5) - 1$,

$$X_\ell^j \sim \mathcal{N}(z_j, 1) \quad \text{for any } j = 1, \ldots, J \text{ and any } \ell = 1, \ldots, d_j,$$

where z_j is the realization of the random variable Z_j defined conditionally to the value of $U \sim \mathcal{U}([0,1])$:

$$\mathcal{L}(Z_j \mid Y = y, U = u) = \begin{cases} \mathcal{N}\left(y\frac{j}{3}, 1\right) & \text{if } u \leq 0.7 \text{ and } j = 1, 2, 3, \\ \mathcal{N}\left(y\frac{j-3}{3}, 1\right) & \text{if } u > 0.7 \text{ and } j = 4, 5, 6, \\ \mathcal{N}(0, 1) & \text{otherwise.} \end{cases}$$

For any $j = 1, \ldots, J$ and any $\ell, \ell' = 1, \ldots, J$,

$$\text{Cov}\left(X_\ell^j, X_{\ell'}^{j'}\right) = \begin{cases} 0.8^{|\ell - \ell'|} & \text{if } j = j' \text{ (within-group correlation),} \\ 0 & \text{otherwise.} \end{cases}$$

The four experiments introduced in section 3.2.5 are used to assess the sensitivity to the number of noisy groups/variables.

- Experiment 1: (easy case)
  - $J = 12$,
  - $d_j = 10$ for any $j = 1, \ldots, 12$.
- Experiment 2: (many noisy groups)
  - $J = 56$,
  - $d_j = 10$ for any $j = 1, \ldots, 56$.
- Experiment 3: (a large noisy group*)
  - $J = 13$,

– $d_j = \begin{cases} 10 & \text{if } j = 1, \dots, 12, \\ 100 & \text{if } j = 13. \end{cases}$.

- Experiment 4: (a large noisy group* and inclusion of 10 noisy variables in the most relevant group*)
  – $J = 13$,
  – $d_j = \begin{cases} 20 & \text{if } j = 3, \\ 100 & \text{if } j = 13, \\ 10 & \text{otherwise.} \end{cases}$

  ($\star$) *noisy variables are independent standard Gaussian variables.*

Model 2. $n = 1000$, $Y = \mathbb{1}_{\{H_1(\mathbf{X}^1, \mathbf{X}^2) \geq 2.5\}}$ with

$$H_1(\mathbf{X}^1, \mathbf{X}^2) = 3\mathbb{1}_{X_1^1 X_2^1 > X_3^1 X_4^1} + 2\mathbb{1}_{X_1^2 X_2^2 > X_3^2 X_4^2},$$

$J = 10$ and $d_j = 5$ for any $j = 1, \dots, 10$.

Three experiments are considered by varying the correlation within and between groups.

- Experiment 1: (independent case)
  $\mathbf{X}$ follows the standard multivariate Gaussian distribution.
- Experiment 2: (within-group correlation)
  $\mathbf{X}$ is a Gaussian vector with zero mean and

  $$\mathrm{Cov}\left(X_\ell^j, X_{\ell'}^{j'}\right) = \begin{cases} 0.5^{|\ell - \ell'|} & \text{if } j = j', \\ 0 & \text{otherwise,} \end{cases}$$

  for any $j = 1, \dots, J$ and any $\ell, \ell' = 1, \dots, J$.
- Experiment 3: (within-group and between-group correlation)
  $\mathbf{X}$ is a Gaussian vector with zero mean and

  $$\mathrm{Cov}\left(X_\ell^j, X_{\ell'}^j\right) = 0.5^{|\ell - \ell'|},$$

  for any $\ell, \ell' = 1, \dots, d$.

Model 3. $n = 1000$, $Y = \mathbb{1}_{\{H_2(\mathbf{X}^1, \mathbf{X}^2) \geq 1.5\}} + \mathcal{N}(0, 0.5)$ with

$$H_2(\mathbf{X}^1, \mathbf{X}^2) = 2\mathbb{1}_{2\exp(-|X_1^1 X_2^1|)X_3^1 - \sin(X_4^1 X_5^1) > 0.55} + \mathbb{1}_{2\exp(-|X_2^2 X_2^2|)X_3^2 - \sin(X_4^2 X_5^2) > 0.55}$$

and $X \sim \mathcal{U}([0,1]^d)$.

Three experiments are considered by varying the size or the number of groups.

- Experiment 1: (large groups)
  - $J = 5$,
  - $d_j = 50$ for any $j = 1, \dots, 5$.
- Experiment 2: (many groups)
  - $J = 50$,
  - $d_j = 5$ for any $j = 1, \dots, 50$.
- Experiment 3: (large noisy groups)
  - $J = 5$,
  - $d_j = \begin{cases} 5 & \text{if } j = 1, 2, \\ 50 & \text{if } j = 3, \dots, 5. \end{cases}$

Model 4. $n = 1000$, $Y = \mathbb{1}_{\{H_3(\mathbf{X}^1, \mathbf{X}^2) \geq 2.5\}} + \mathcal{N}(0, 0.5)$, with

$$H_3(\mathbf{X}^1, \mathbf{X}^2) = 3\mathbb{1}_{X_1^1 + X_2^1 > X_3^1 + X_4^1 + X_5^1} + 2\mathbb{1}_{X_1^2 + X_2^2 > X_3^2 + X_4^2 + X_5^2}.$$

Two experiments are considered by varying both the size of the groups and the correlation.

- Experiment 1: (easy case)
  - $J = 10$,
  - $d_j = 10$ for any $j = 1, \dots, 10$,
  - $\mathbf{X}$ follows the standard multivariate Gaussian distribution (no correlation).
- Experiment 2: (large noisy groups and within-group correlation)
  - $J = 10$,
  - $d_j = \begin{cases} 5 & \text{if } j = 1, 2, \\ 50 & \text{if } j = 3, \dots, 10, \end{cases}$.
  - $\mathbf{X}$ is a Gaussian vector with zero mean and

$$\text{Cov}\left(X_\ell^j, X_{\ell'}^{j'}\right) = \begin{cases} 0.5^{|\ell - \ell'|} & \text{if } j = j', \\ 0 & \text{otherwise,} \end{cases}$$

    for any $j = 1, \dots, J$ and any $\ell, \ell' = 1, \dots, J$.

Each data set is randomly divided into a training sample $\mathcal{D}_n$, a validation sample $\mathcal{V}_m$ and a test sample $\mathcal{T}_q$ of equal size.

In CARTGV, the training sample $\mathcal{D}_n$ is used to build a maximal tree. Next, the validation sample $\mathcal{V}_m$ is used both to select the tuning parameters (i.e. the maximal depth of the splitting trees D_j and the penalty function `pen`) and to prune the maximal tree. The selection of the tuning parameters is described more precisely in section 3.2.5.

In CART, the maximal tree is first built on the training set $\mathcal{D}_n$ and then pruned by using the minimal cost-complexity pruning algorithm and the validation set $\mathcal{V}_m$ (a detailed description of this pruning strategy is introduced in Section 3.2.3).

In RFGV and RF, the training sample $\mathcal{D}_n$ is used to fit the model while the validation sample $\mathcal{V}_m$ is used to select the following tuning parameters of the algorithms:

- the number of variables randomly sampled as candidates at each split for the trees in the RF forests (parameter `mtry` in `randomForest`),
- the number of group randomly sampled as candidates at each split for the trees in the RFGV forests (parameter `mgrp`, see section 3.3.2),
- the number of inputs in group j randomly sampled as candidates at each step for the splitting trees based on group j in the trees of the RFGV forests (parameter $\texttt{mvar}_j$, see section 3.3.2).

For each method the tuning parameters are selected by minimizing the misclassification error computed on the validation set. For RF, we consider four values for the tuning parameter `mtry` $(1, \lfloor\sqrt{d}\rfloor, \lfloor d/3\rfloor, d)$. For RFGV, we consider four values for the tuning parameter `mgroup` $(1, \lfloor\sqrt{J}\rfloor, \lfloor J/3\rfloor, J)$ and four values for the tuning parameter $\texttt{mvar}_j$ $(1, \lfloor\sqrt{d_j}\rfloor, \lfloor d_j/3\rfloor, d_j)$. In RF and RFGV, the number of trees `ntree` is set to its defaults value `ntree` $= 500$. Moreover, in RFGV, the penalty function `pen` and the maximal depth of the splitting trees D_j are fixed: $D_j = 2$ and $\texttt{pen}(d_j) = 1$.

Finally, the test sample $\mathcal{T}_q$ is used to measure the predictive performances of each method. For reasons of computing time, all results are averaged over 50 independent replications.

3.4.2 Performances of the methods

The methods are assessed and compared in terms of prediction accuracy. This criterion is measured for each predictor by the area under the curve (AUC) and the misclassification rate (Error) on the test set $\mathcal{T}_q$. Table 3.7 and Figures 3.9-3.12 display AUC and misclassification rates for all experiments. Table 3.7 provides the mean and standard deviation for

each criterion.

First of all, in all experiments, RFGV outperforms CARTGV.

Furthermore, for experiments in Model 1 in which the group structure mainly lies in the correlation between the input variables, it seems that RF outperforms all the other methods. Specifically, in all experiments using Model 1, RF and CART respectively have slightly higher performances than RFGV and CARTGV respectively.

On the contrary, for Models 2-4 in which the relationship between the response variable and the inputs is defined through the group structure of the inputs, RFGV generally outperforms all the other methods, and CARTGV often shows better performances than CART.

In Model 2, three experiments are considered by varying the correlation between the input variables. In these three experiments, RFGV significantly outperforms all the other methods. Moreover, when there is no correlation (i.e. in experiment 1), CARTGV shows better performances than RF. Note that when inputs are correlated (see experiment 2 and 3 in Model 1), all methods perform better in terms of AUC and misclassification rate.

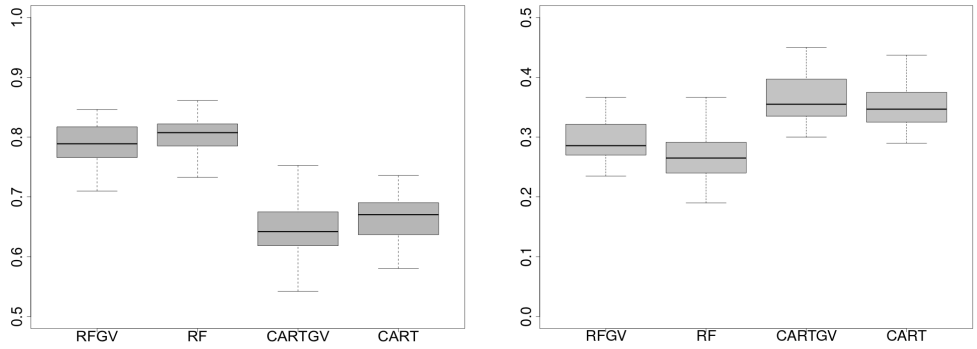
Next, Model 3 is used to highlight the performances of the methods when the number or the size of the groups vary. In the first experiment, there are few groups of inputs ($J = 5$) which contains lots of noisy variables ($d_j = 50$, for any $j = 1, \ldots, 5$). In this situation, RF and RFGV have similar performances, like CARTGV with CART. In the two other scenarios, when there are lots of small groups or large noisy groups, RFGV outperforms all the other methods and CARTGV has better performances than CART.

Finally, the two experiments in Model 4 support the results obtained with Model 3, that is RFGV outperforms all the other methods and CARTGV outperforms CART when the noisy groups are large.

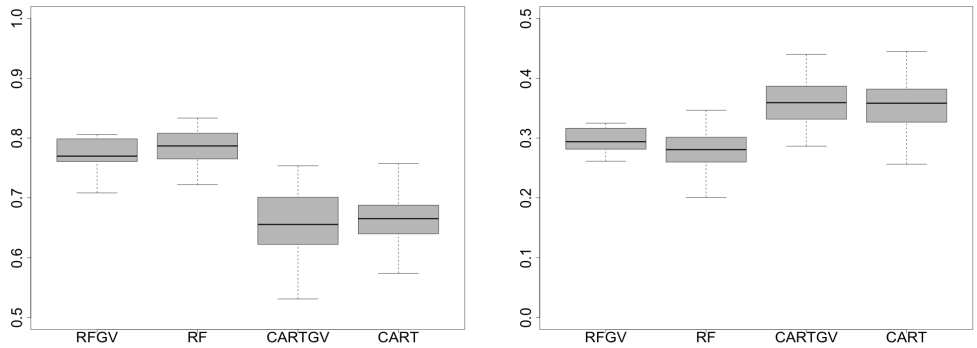
In conclusion, it seems that a RFGV forest has generally better predictive performances than a single CARTGV tree. Moreover, when information lies in the group structure (such as experiments in Model 2-4), RFGV and CARTGV seems more adapted to elaborate prediction rules than RF and CART. Otherwise, when the group structure is only used to control the correlation between the inputs (as for instance Model 1) RFGV and RF (respectively CARTGV and CART) have similar performances.

	RFGV	RF	CARTGV	CART
Model 1, Exp.1				
AUC	0.787 (0.041)	**0.801 (0.033)**	0.646 (0.044)	0.662 (0.043)
Error	0.295 (0.039)	**0.268 (0.035)**	0.366 (0.039)	0.355 (0.036)
Model 1, Exp.2				
AUC	0.770 (0.033)	**0.787 (0.033)**	0.657 (0.053)	0.66 (0.045)
Error	0.301 (0.032)	**0.282 (0.033)**	0.359 (0.04)	0.355 (0.041)
Model 1, Exp.3				
AUC	**0.809 (0.032)**	**0.809 (0.033)**	0.66 (0.044)	0.67 (0.039)
Error	**0.254 (0.029)**	0.268 (0.036)	0.353 (0.04)	0.347 (0.036)
Model 1, Exp.4				
AUC	**0.810 (0.033)**	**0.810 (0.03)**	0.636 (0.052)	0.666 (0.043)
Error	**0.262 (0.033)**	0.267 (0.03)	0.376 (0.042)	0.349 (0.034)
Model 2, Exp.1				
AUC	**0.896 (0.026)**	0.629 (0.077)	0.694 (0.09)	0.566 (0.091)
Error	**0.193 (0.024)**	0.404 (0.061)	0.316 (0.084)	0.444 (0.077)
Model 2, Exp.2				
AUC	**0.936 (0.022)**	0.851 (0.043)	0.781 (0.042)	0.727 (0.055)
Error	**0.15 (0.033)**	0.236 (0.041)	0.245 (0.044)	0.312 (0.06)
Model 2, Exp.3				
AUC	**0.926 (0.029)**	0.847 (0.046)	0.786 (0.048)	0.728 (0.058)
Error	**0.172 (0.043)**	0.241 (0.039)	0.245 (0.053)	0.306 (0.063)
Model 3, Exp.1				
AUC	0.867 (0.029)	**0.868 (0.024)**	0.816 (0.035)	0.815 (0.028)
Error	0.202 (0.02)	**0.195 (0.022)**	0.206 (0.024)	0.208 (0.027)
Model 3, Exp.2				
AUC	**0.884 (0.017)**	0.871 (0.02)	0.834 (0.031)	0.823 (0.028)
Error	**0.186 (0.015)**	0.19 (0.018)	0.2 (0.025)	0.206 (0.023)
Model 3, Exp.3				
AUC	**0.898 (0.019)**	0.868 (0.021)	0.828 (0.032)	0.808 (0.026)
Error	**0.183 (0.02)**	0.195 (0.021)	0.206 (0.026)	0.218 (0.03)
Model 4, Exp.1				
AUC	**0.857 (0.013)**	0.848 (0.023)	0.709 (0.034)	0.701 (0.028)
Error	**0.214 (0.02)**	0.227 (0.026)	0.309 (0.029)	0.316 (0.028)
Model 4, Exp.2				
AUC	**0.858 (0.027)**	0.821 (0.029)	0.728 (0.034)	0.708 (0.037)
Error	**0.212 (0.032)**	0.251 (0.03)	0.303 (0.031)	0.319 (0.036)

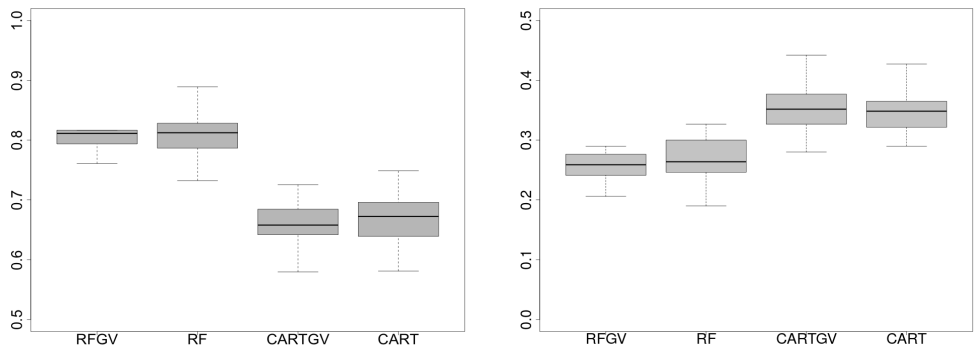
Table 3.7: Performances of RFGV, RF, CARTGV and CART. For each criterion, the mean value is given following by the value of the standard deviation in brackets.



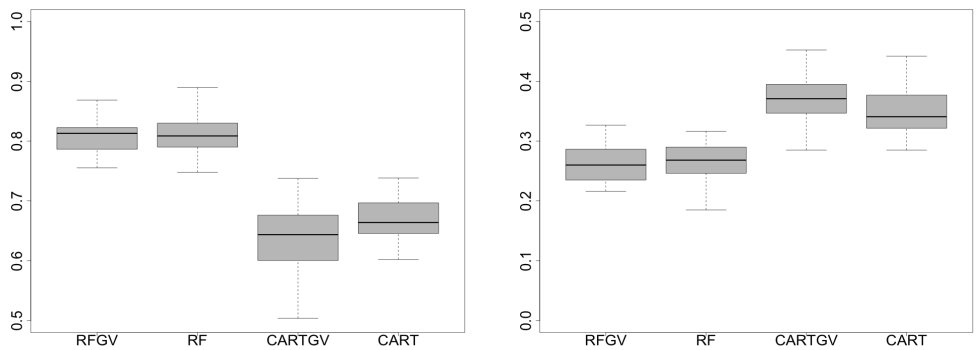
Experiment 1



Experiment 2

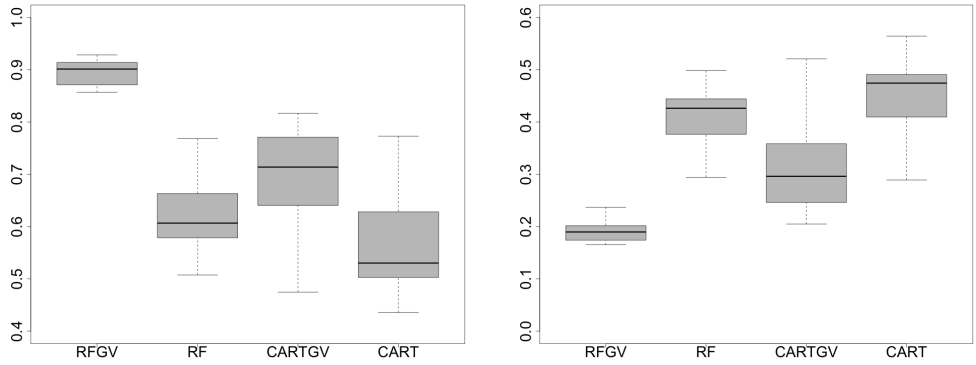


Experiment 3

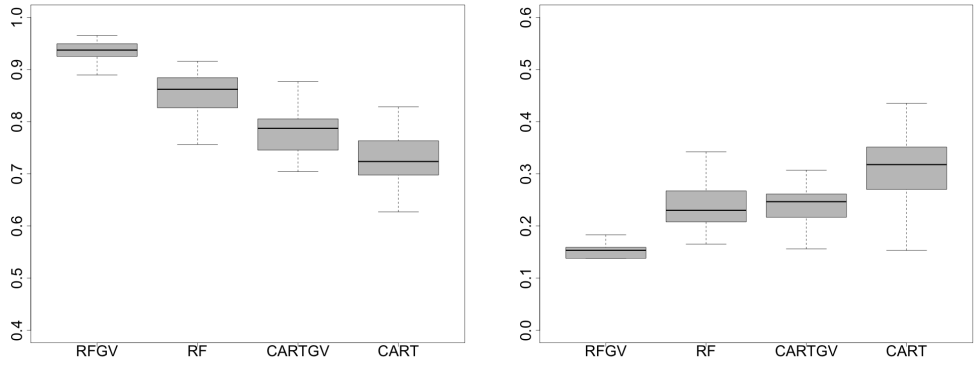


Experiment 4

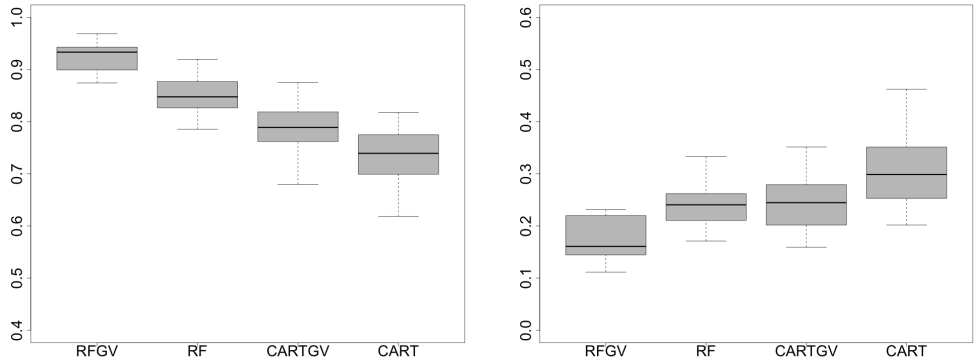
Figure 3.9: Performances of RFGV, RF, CARTGV and CART: distribution of AUC (left) and misclassification rate (right) for experiments in Model 1.



Experiment 1

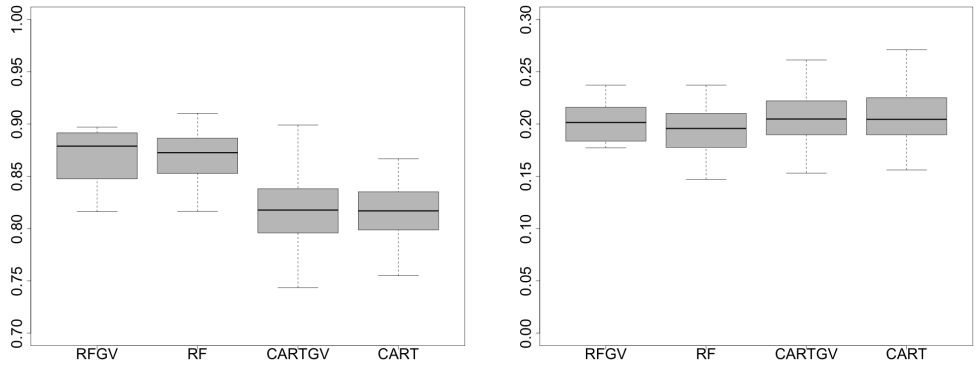


Experiment 2

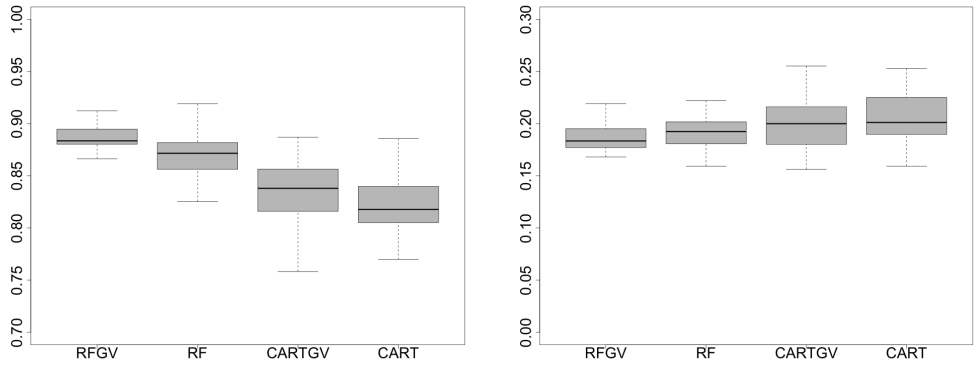


Experiment 3

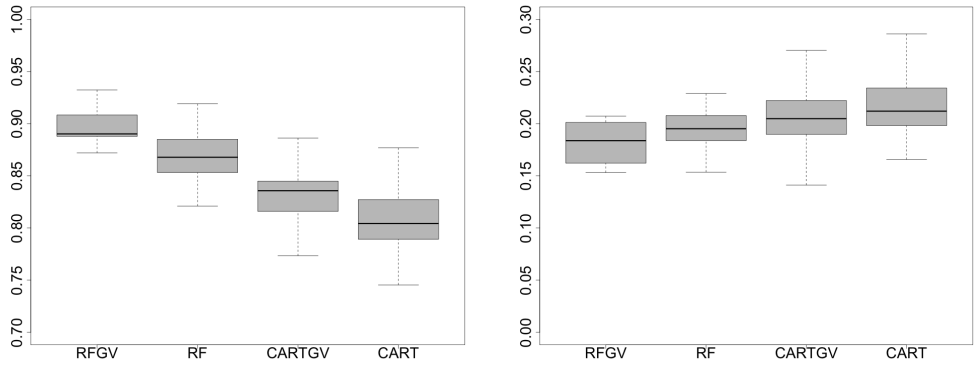
Figure 3.10: Performances of RFGV, RF, CARTGV and CART: distribution of AUC (left) and misclassification rate (right) for experiments in Model 2.



Experiment 1

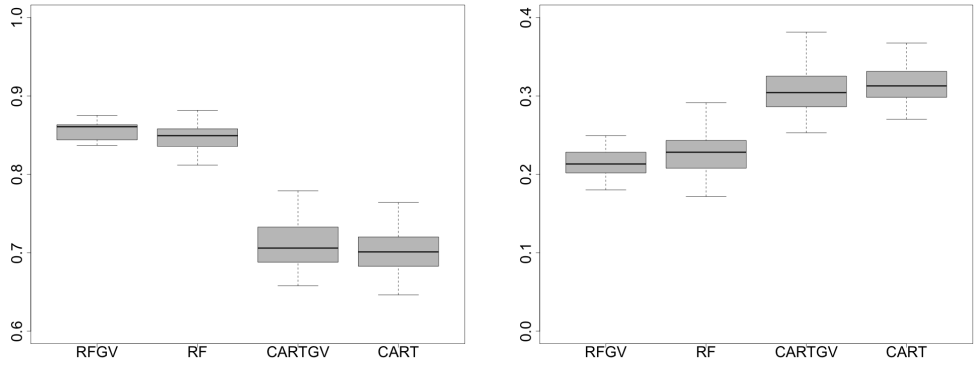


Experiment 2

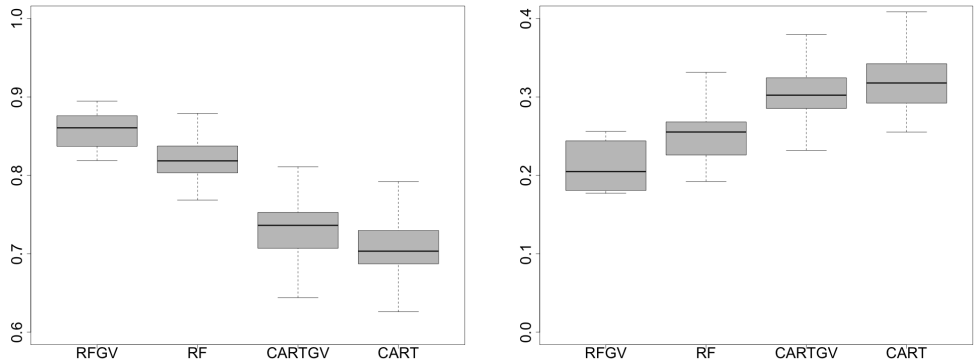


Experiment 3

Figure 3.11: Performances of RFGV, RF, CARTGV and CART: distribution of AUC (left) and misclassification rate (right) for experiments in Model 3.



Experiment 1



Experiment 2

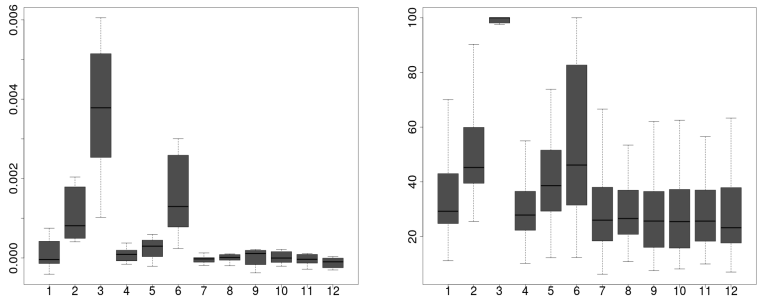
Figure 3.12: Performances of RFGV, RF, CARTGV and CART: distribution of AUC (left) and misclassification rate (right) for experiments in Model 4.

3.4.3 Assessment of the measures of group importance

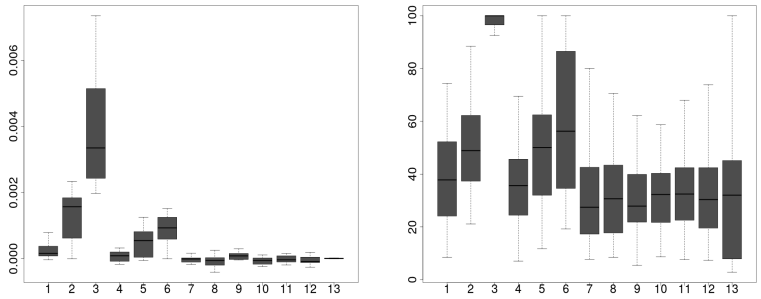
This subsection investigates the performances of the permutation-based group importance measure in RFGV and the scores GRI and GSI in CARTGV. For purpose of comparison, the group permutation importance measure is also calculated with RF as originally proposed by Gregorutti et al. (2015)).

Figure 3.13 displays the distribution of the group importance for each score in experiments 1 and 3 in Model 1 and experiment 1 in Model 2. We only show the permutation-based importance for RFGV since results are identical to the permutation-based importance computed with RF. Also, for the scores GRI and GSI, we only display the distribution of GSI since the two scores give the same results. Note that, as the importance scores are defined differently, they can only be compared in terms of variable ranking.

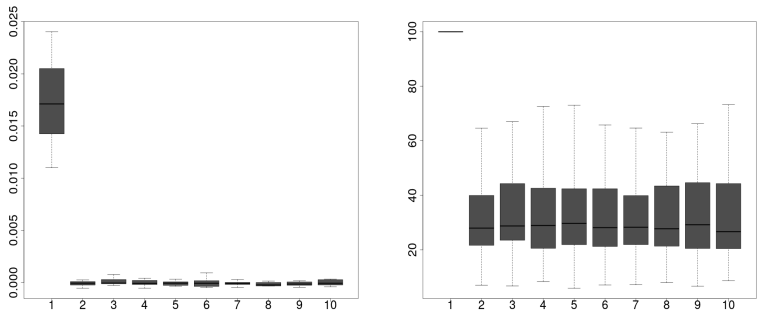
Overall, all importance measures well identify and correctly rank the true relevant groups.



Model 1, Experiment 1



Model 1, Experiment 3



Model 2, Experiment 1

Figure 3.13: Distribution of the group importance for the permutation-based importance in RFGV (left) and the GSI in CARTGV (right).

Remark 3.4.1. *In experiment 1 in model 2, we observe that all scores only identify the first relevant group and the importance of the second score is similar to the one of the noisy group. This is due to the model definition. Specifically, we use the value* 2.5 *for the threshold in the equation* $Y = \mathbb{1}_{\{H_1(\mathbf{X}^1,\mathbf{X}^2)\geq 2.5\}}$. *Consequently, according to the definition of the function* H_1, *that we recall is* $H_1(\mathbf{X}^1,\mathbf{X}^2) = 3\mathbb{1}_{X_1^1X_2^1>X_3^1X_4^1} + 2\mathbb{1}_{X_1^2X_2^2>X_3^2X_4^2}$, *the value of the response* Y *is merely defined based on the values of the variables belonging to the first group. The*

importance is not displayed for the other experiments since results are similar than those observed in experiment 1 in Model 2

3.5 Conclusion

In this paper, we have proposed two new methods for estimating prediction rules with grouped inputs.

The first method, which we called CARTGV, can be considered as an adaptation of CART to the context of predictor groups. The method builds a large decision tree by means of a recursive partitioning of the input space. Unlike CART, a split is not necessarily binary and is defined according to a binary tree based on a group of inputs. Once fully grown, the tree is pruned by using a generalization of the minimal cost-complexity pruning. The validity of this pruning technique in the context of non-binary trees has been then demonstrated and a detailed explanation of the pruning method has been provided.

Next, we have introduced RFGV, an original random forests method for groups of inputs. The method consists in aggregating a large number of random trees. Each tree is built with respect to a variant of CARTGV that performs resampling of both groups and inputs.

Thanks to their non-parametric nature, these two approaches offer a much wider applicability than the popular regularized methods used in the context of predictor groups (Yuan & Lin, 2006; Meier et al., 2008). Moreover, in parallel CARTGV and RFGV provide automatically some measures of importance for each predictor group and can be then used to identify relevant groups.

Through extensive simulation studies, we have shown that, in the context of grouped variables, CARTGV and RFGV are very attractive techniques both to estimate prediction rules and to perform group variable selection.

3.6 Appendices

3.6.1 Proof of the positivity of the decrease in impurity in the context of non-binary splits

First, let us introduce some notations and definitions. Consider a node t. If Y is a class label such that $\mathcal{Y} = \{1, \ldots, K\}$ and $K \geq 2$, then for any $k = 1, \ldots, K$,

$$\hat{p}_{k,t} = n_t^{-1} \sum_{x_i \in t} \mathbb{1}_{y_i = k}$$

denotes the empirical probability that an observation in t belongs to the class k. We define the impurity $\mathrm{Q}(t)$ of t by

$$\mathrm{Q}(t) = \phi(p_{1,t}, \ldots, p_{K,t}) \tag{3.10}$$

where ϕ is a strictly concave function defined on $(p_{1,t}, \ldots, p_{K,t})$ and that satisfies the properties:

(i) ϕ becomes minimum at points $(1, 0, \ldots, 0), \ldots, (0, \ldots, 0, 1)$,

(ii) ϕ becomes maximum at points $(1/K, \ldots, 1/k)$,

(iiI) ϕ is symmetric with regard to its arguments $p_1, \ldots, p_K$.

Proposition 3.6.1. *For every node t and every split of t into $L \geq 2$ child nodes, which are denoted by $t_1, \ldots, t_L$, the decrease in impurity defined as*

$$\Delta\mathrm{Q}(t) \;=\; n_t \mathrm{Q}(t) - \sum_{l=1}^{L} n_{t_l} \mathrm{Q}\left(t_l\right)$$

is positive, i.e.

$$\Delta\mathrm{Q}(t) \geq 0.$$

Proof . A detailed proof of this result is given for the classification setting. As it is similar in the regression setting, we only give the sketch of the proof.
Consider the classification problem with $K \geq 2$ classes. The decrease in impurity can be written with respect to ϕ (3.10)

$$\begin{aligned}
\frac{1}{n_t}\Delta\mathrm{Q}(t) \;&=\; \mathrm{Q}(t) - \sum_{l=1}^{L} \frac{n_{t_l}}{n_t} \mathrm{Q}\left(t_l\right) \\
&=\; \phi(p_{1,t}, \ldots, p_{K,t}) - \sum_{l=1}^{L} \frac{n_{t_l}}{n_t} \phi(p_{1,t_l}, \ldots, p_{K,t_l}).
\end{aligned}$$

Since ϕ is strictly concavity of ϕ and $\frac{n_{t_i}}{n_t} \in]0; 1[$, for all $i = 1, \ldots, L$, we have:

$$\phi\left(\sum_{l=1}^{L} \frac{n_{t_l}}{n_t} p_{1,t_l}, \ldots, \sum_{l=1}^{L} \frac{n_{t_l}}{n_t} p_{K,t_l}\right) \;\geq\; \sum_{l=1}^{L} \frac{n_{t_l}}{n_t} \phi(p_{1,t_l}, \ldots, p_{K,t_l}),$$

with $\sum_{i=1}^{L} \frac{n_{t_l}}{n_t} p_{k,t_l} = p_{k,t}$, $k = 1, \ldots, K$, (by using the total law Theorem).
So, we have:

$$\begin{aligned}
\phi(p_{1,t}, \ldots, p_{K,t}) \;&\geq\; \sum_{l=1}^{L} \frac{n_{t_l}}{n_t} \phi(p_{1,t_l}, \ldots, p_{K,t_l}) \\
\mathrm{Q}(t) \;&\geq\; \sum_{l=1}^{L} \frac{n_{t_l}}{n_t} \mathrm{Q}\left(t_l\right).
\end{aligned}$$

So

$$\Delta \mathrm{Q}(t) \ \geq \ 0,$$

with equality holding if, and only if, $p_{k,t} = p_{k,t_l}$, for all $k = 1, \ldots, K$ and all $l = 1, \ldots, L$. Thus, the result is true in classification setting.
In regression setting, the impurity $\mathrm{Q}(t)$ of a node t is generally defined by using the mean squared error

$$\mathrm{Q}(t) = \frac{1}{n_t} \sum_{i:\mathbf{X}_i \in t} (Y_i - \bar{Y}_t)^2.$$

The result is directly obtained by using the Jensen Inequality. ■

3.6.2 Time complexity of the CARTGV algorithm

Let consider a node t that include n_t observations. The input vector $\mathbf{X}$ is structured into J known groups. Let X^j the j-th group which includes variables.

The split of the node t is based on this two step:

(1) **Choice of a split for each group:** the complexity of building a CART tree based on the variables belonging $\mathbf{X}^j$ is $\mathcal{O}(d_j n_t D_j)$ with D_j denoting the maximal depth of the CART tree. See Witten et al. (2016, chapter 6), for a detailed calculation of CART time complexity. As a tree is built for each group, the complexity becomes $\mathcal{O}(dn_t D)$, with $D = \max_{j \in 1,\ldots,J} D_j$.

(2) **Choice of the splitting group:** the complexity for computing the decrease in impurity resulting from the split based on $\mathbf{X}^j$ is $\mathcal{O}(n_t)$. It is repeated on each group and the algorithm then selects the group maximizing the decrease in impurity. So, the complexity of this step is then $\mathcal{O}(Jn_t) + \mathcal{O}(1)$

Consequently, the maximal time complexity of the CARTGV algorithm at a node t is in the worst case $\mathcal{O}(dn_t D) + \mathcal{O}(Jn_t) + \mathcal{O}(1) = \mathcal{O}(dn_t D)$ with $D = \max_{j \in 1,\ldots,J} D_j$.

3.6.3 Generalization of the minimal cost-compexity pruning for CARTGV trees

Some notations and properties:

First, let us recall some notations. Consider a non trivial tree T with root t_1.

- $\widetilde{T}$ is the set of leaves of T and $|\widetilde{T}|$ refers to the cardinality of $\widetilde{T}$.
- If t is a non-terminal node of T, then $\tilde{t}$ refers to the set of child nodes of t with cardinality $|\tilde{t}|$.

- T_t denotes the branch of T coming from t, that is the subtree of T with root t.
- A tree T' is a pruned subtree of T, if T' is a subtree of T having the same root as T. We write $T' \preceq T$
- Take t a non-terminal node of a tree T. Then, the pruned tree of T obtained by pruning T in t is the pruned tree of T where t becomes a leaf and all descendants of t are removed. It is denoted by $T \setminus T_t$. Figure 3.14 illustrates this notion.

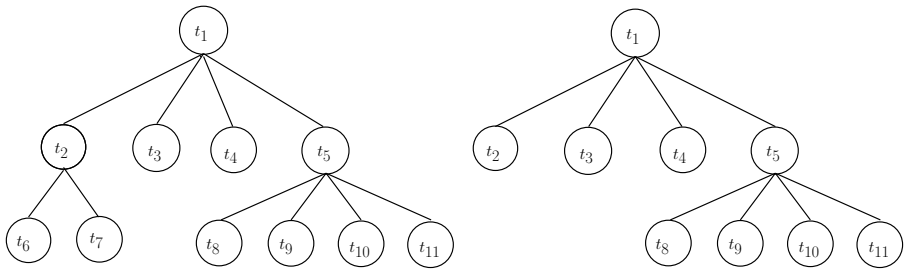


Figure 3.14: The tree T (left) displayed in Figure 3.1 and the tree $T \setminus T_{t_2}$ (right) obtained by pruning the tree T in the node t_2.

Let $R(t, \mathcal{D}_n)$ define the prediction error of the node t on the training samples $\mathcal{D}_n$. For the sake of convenience, we write $R(t)$ instead of $R(t, \mathcal{D}_n)$. In regression, $\mathcal{R}(t)$ is the mean square error

$$\mathcal{R}(t) = \frac{1}{n} \sum_{i:(\mathbf{X}_i, Y_i) \in t} (Y_i - \hat{y}_t)^2,$$

where we recall that n is the size of $\mathcal{D}_n$ and $\hat{y}_t$ is the prediction value in the node t. In classification, $\mathcal{R}(t)$ is the empirical misclassification rate

$$\mathcal{R}(t) = \frac{1}{n} \sum_{i:(\mathbf{X}_i, Y_i) \in t} \mathbb{1}_{\hat{y}_t \neq Y_i}.$$

The prediction error verifies the following property.

Lemma 3.6.1. *For any split of t into the set $\tilde{t}$ of leaves,*

$$R(t) \geq \sum_{t' \in \tilde{t}} \mathcal{R}(t').$$

The proof is given by Breiman et al. (1984, see chapter 4).

Let consider the prediction error $\mathcal{R}(T)$ of a tree T, that we define in Section 3.2.3. $\mathcal{R}(T)$ can be also written as the sum, over all terminal node of T, of the prediction error:

$$\mathcal{R}(T) \ = \ \sum_{t \in \tilde{T}} \mathcal{R}(t).$$

Similarly for $\mathcal{R}_\alpha(t) = \mathcal{R}(t) + \alpha$, it follows

$$\mathcal{R}_\alpha(T) = \sum_{t \in \widetilde{T}} \mathcal{R}_\alpha(t).$$

If T is a non trivial tree and t is a node of T, then the cost complexity measure $\mathcal{R}_\alpha(T_t)$ of the branch T_t is

$$\begin{aligned} \mathcal{R}_\alpha(T_t) &= \mathcal{R}(T_t) + \alpha|\widetilde{T}_t| \\ &= \sum_{t \in \widetilde{T}_t} \mathcal{R}_\alpha(t), \end{aligned}$$

where $\widetilde{T}_t$ is the set of terminal node of the branch T_t.

Let consider $\tilde{t}_0$ the set of child nodes of the root t_0 of T. The branch T_t, with $t \in \tilde{t}_0$, is called a primary branch of T and we have

$$\mathcal{R}_\alpha(T) = \sum_{t \in \tilde{t}_0} \mathcal{R}_\alpha(T_t). \tag{3.11}$$

Explanation of the generalization of the cost-minimal complexity pruning and proof of its validity:

Consider a large CARTGV tree T that we want to prune by using the minimal cost-complexity pruning. As explained in Section 3.2.3, the minimal cost-complexity pruning consists in constructing a sequence of pruned subtrees of T that contains, for any α, the subtree $T(\alpha)$ minimizing $\mathcal{R}_\alpha$ (3.5). Since the exhaustive search of $T(\alpha)$, for any value of α, through the whole set of pruned subtree of T is computationally expensive. Breiman et al. (1984) proposed using a very effective and computationally inexpensive algorithm to solve the pruning problem. Originally, this method was proposed to pruned binary trees. We are now going to explain and to prove the validity of the pruning algorithm in the more general framework of non-binary decisions trees. We call this extension the generalized minimal cost-complexity pruning.

First of all, let us formally define $T(\alpha)$.

Definition 3.6.1. *Let T be a tree (not necessarily binary) and consider the cost complexity measure $\mathcal{R}_\alpha$ (3.5). For any $\alpha \in \mathbb{R}^+$, the smallest α-optimally subtree of T denoted by $T(\alpha)$ is the pruned subtree of T that fulfills the following conditions:*

(i) $\mathcal{R}_\alpha(T(\alpha)) = \min_{T' \preceq T} \mathcal{R}_\alpha(T')$,

(ii) *if* $\mathcal{R}_\alpha(T(\alpha)) = \mathcal{R}_\alpha(T')$*, for* $T' \preceq T$*, then* $T(\alpha) \preceq T'$.

The first condition means that, at the value α, there is no subtree of T with smaller complexity-cost than $T(\alpha)$. The second condition implies that if several trees reach this minimum, the smallest tree is preferred.

As the set of pruned subtrees of $T_{\max}$ is finite, the first condition is always satisfied by at least one pruned subtree of $T_{\max}$. Nonetheless, the second condition implies the uniqueness of the smallest α-optimally subtree $T(\alpha)$ and calls into question the existence of $T(\alpha)$ (i.e. suppose that there are two subtrees T and T' that satisfied the first condition but neither are nested in the other). Thus, existence of $T(\alpha)$ for every value of α can be questionable and is investigated below.

Now, the proof of the validity of the generalized minimal cost-complexity algorithm is organized as follows.

1. We first prove the existence of $T(\alpha)$ for any $\alpha \in \mathbb{R}^+$ and any tree non-binary T.
2. The definition $T(\alpha)$ is refined.
3. We demonstrate some properties of $T(\alpha)$ that enable to circumvent the exhaustive search for $T(\alpha)$ in the whole set of pruned subtrees of T
4. The elaboration of the sequence of optimal subtrees is described step by step.

First, the following lemma shows existence of the smallest α-optimally subtree $T(\alpha)$ for any non-binary tree T and any $\alpha \in \mathbb{R}^+$.

Lemma 3.6.2. *For any* $\alpha \in \mathbb{R}^+$ *and any tree* T *with root* t_1 *and that is non-necessarily binary, the smallest* α*-optimally subtree* $T(\alpha)$ *of* T *exists. Moreover, it satisfies*

$$\mathcal{R}_\alpha(T(\alpha)) \;=\; \min \left[\mathcal{R}_\alpha(t_1), \sum_{t \in \tilde{t}_1} \mathcal{R}_\alpha(T_t(\alpha)) \right]. \tag{3.12}$$

Proof . This result is proved by induction. When T is trivial that is $T = \{t_1\}$ (or equivalently the depth of T equals 1), $T(\alpha) = \{t_1\}$ and $\mathcal{R}_\alpha(T(\alpha)) = \mathcal{R}_\alpha(t_1)$. Then, the result is true.
Now, suppose that the result is true for all trees of depth less than $p \geq 2$. Let T be a tree of depth p and with root t_1. T can be written as:

$$T = \{t_1\} \bigcup_{t \in \tilde{t}_1} T_t.$$

Then, we can write

$$\begin{aligned}\min_{T'\preceq T}\mathcal{R}_\alpha(T') &= \min\left[\mathcal{R}_\alpha(t_1),\ \min_{\substack{T'\preceq T\\ T'\neq\{t_1\}}}\ \mathcal{R}_\alpha(T')\right]\\ &= \min\left[\mathcal{R}_\alpha(t_1),\ \min_{\substack{T'\preceq T\\ T'\neq\{t_1\}}}\ \sum_{t\in\tilde{t}_1}\mathcal{R}_\alpha(T'_t)\right] \quad \text{according to (3.11)}\\ &= \min\left[\mathcal{R}_\alpha(t_1), \sum_{t\in\tilde{t}_1}\ \min_{T''\preceq T_t}\mathcal{R}_\alpha(T'')\right].\end{aligned}$$

Now, consider the primary branches of T which are denoted T_t for all $t \in \tilde{t}_0$. Observe that any primary branch T_t is a tree of depth less than p. Then, according to the induction hypothesis, the smallest α-optimally subtree $T_t(\alpha)$ of T_t exists for any α, and by definition it satisfies

$$\mathcal{R}_\alpha\left(T_t(\alpha)\right) = \min_{T''\preceq T_t}\mathcal{R}_\alpha(T'').$$

Consequently, we can write

$$\min_{T'\preceq T}\mathcal{R}_\alpha(T') = \min\left[\mathcal{R}_\alpha(t_1), \sum_{t\in\tilde{t}_1}\ \mathcal{R}_\alpha\left(T_t(\alpha)\right)\right].$$

It follows that.

- If $\min_{T'\preceq T}\mathcal{R}_\alpha(T') = \mathcal{R}_\alpha(t_1)$, then $T(\alpha)$ exists: $T(\alpha) = \{t_0\}$.
- Otherwise, we have $\min_{T'\preceq T}\mathcal{R}_\alpha(T') = \sum_{t\in\tilde{t}_1}\ \mathcal{R}_\alpha(T_t(\alpha))$ and $T(\alpha)$ also exists: $T(\alpha) = \{t_0\} \underset{t\in\tilde{t}_0}{\cup} T_t(\alpha)$.

Consequently, the result is valid by mathematical induction.

■

Lemma 3.6.2 ensures that, for any α, it cannot occur that two trees achieve the minimum for $\mathcal{R}_\alpha$ but neither is a subtree of the other. The following result refines the definition of $T(\alpha)$.

Lemma 3.6.3. *For each $\alpha \in \mathbb{R}^+$ and any tree T with root t_1 and that is non-necessarily binary, the two conditions are satisfied:*

(i) *if $\mathcal{R}_\alpha(T_t) \leq \mathcal{R}_\alpha(t)$, for all $t \in T \setminus \widetilde{T}$, then $\mathcal{R}_\alpha(T) = \mathcal{R}_\alpha(T(\alpha))$,*

(ii) *if* $\mathcal{R}_\alpha(T_t) < \mathcal{R}_\alpha(t)$, *for all* $t \in T \setminus \widetilde{T}$, *then* $T = T(\alpha)$.

Proof . We first demonstrate statement (i) by mathematical induction. When T is trivial that is $T = \{t_1\}$, it is obvious: for each α, $T(\alpha) = \{t_1\}$ and $\mathcal{R}_\alpha(T(\alpha)) = \mathcal{R}_\alpha(t_1)$.
Now, suppose that the result is true for all trees of depth less than $p \geq 2$. Let T be a tree of depth p and with root t_1

$$T = \{t_1\} \bigcup_{t \in \tilde{t}_1} T_t.$$

and we assume that

$$\mathcal{R}_\alpha(T_t) \leq \mathcal{R}_\alpha(t), \quad \text{for all } t \in T \setminus \widetilde{T}. \tag{3.13}$$

We can easily notice that the condition defined by the equation (3.13) is also satisfied by all the primary branches T_t, with $t \in \tilde{t}_0$. Indeed, we have

$$\mathcal{R}_\alpha(T_{t,t'}) \leq \mathcal{R}_\alpha(t'), \quad \text{for all } t' \in T_t \setminus \widetilde{T}_t,$$

where $T_{t,t'}$ is the branch of T_t coming from t' and $\widetilde{T}_t$ is the set of leaves of the primary branch T_t. Then, by using the induction hypothesis, for all branches T_t, we have

$$\mathcal{R}_\alpha(T_t) = \mathcal{R}_\alpha(T_t(\alpha)), \tag{3.14}$$

where we recall that $T_t(\alpha)$ denotes the smallest α-optimally subtree of T_t. Therefore, following equation (3.11), we can write

$$\mathcal{R}_\alpha(T) = \sum_{t \in \tilde{t}_1} \mathcal{R}_\alpha(T_t),$$

and by equation (3.14),

$$\mathcal{R}_\alpha(T) = \sum_{t \in \tilde{t}_1} \mathcal{R}_\alpha(T_t(\alpha)). \tag{3.15}$$

Now, by using Lemma 3.6.2, for any α we have

$$T(\alpha) = \{t_0\}, \quad \text{or} \quad T(\alpha) = \{t_0\} \bigcup_{t \in \tilde{t}_0} T_t(\alpha),$$

and the condition defined by equation (3.13) implies that

$$\mathcal{R}_\alpha(t_1) \geq \sum_{t \in \tilde{t}_1} \mathcal{R}_\alpha(T_t(\alpha)).$$

It follows:

- if $\mathcal{R}_\alpha(t_1) = \sum_{t \in \tilde{t}_1} \mathcal{R}_\alpha(T_t(\alpha))$, then $T(\alpha) = \{t_0\}$ and $\mathcal{R}_\alpha(T) = \mathcal{R}_\alpha(T(\alpha))$ (by using equation (3.15)),

- otherwise $\mathcal{R}_\alpha(t_1) > \sum_{t\in\tilde{t}_1} \mathcal{R}_\alpha(T_t(\alpha))$, so $T(\alpha) = \{t_0\} \underset{t\in\tilde{t}_0}{\cup} T_t(\alpha)$ and $\mathcal{R}_\alpha(T) = \mathcal{R}_\alpha(T(\alpha))$ (by using equation (3.15)).

So condition (i) is proved.

(ii)
For the second statement of the lemma, the result is also obvious when T is trivial. Suppose that the result is true for all trees of depth less than $p \geq 2$. Let T be a tree of depth p and with root t_1

$$T = \{t_1\} \underset{t\in\tilde{t}_1}{\cup} T_t.$$

and we assume that

$$\mathcal{R}_\alpha(T_t) < \mathcal{R}_\alpha(t), \quad \text{for all } t \in T \setminus \widetilde{T}. \tag{3.16}$$

The condition defined by equation (3.16) is also satisfied by all the primary branches T_t, with $t \in \tilde{t}_0$. Indeed, we have

$$\mathcal{R}_\alpha(T_{t,t'}) < \mathcal{R}_\alpha(t'), \quad \text{for all } t' \in T_t \setminus \widetilde{T}_t.$$

Then, by using the induction hypothesis, for all branches T_t it follows that

$$T_t(\alpha) = T_t. \tag{3.17}$$

Now, according to Lemma 3.6.2 we have

$$T(\alpha) = \{t_0\}, \quad \text{or} \quad T(\alpha) = \{t_0\} \underset{t\in\tilde{t}_0}{\cup} T_t(\alpha).$$

But the condition defined by equation (3.16) implies

$$T(\alpha) = \{t_0\} \underset{t\in\tilde{t}_0}{\cup} T_t(\alpha).$$

Therefore, it follows that:

$$\begin{aligned} T(\alpha) &= \{t_0\} \underset{t\in\tilde{t}_0}{\cup} T_t(\alpha) \\ &= \{t_1\} \underset{t\in\tilde{t}_1}{\cup} T_t \quad \text{using (3.17)} \\ &= T. \end{aligned}$$

Thus the second point is true. ■

Lemma 3.6.2 and Lemma 3.6.3 ensure that for any $\alpha \in \mathbb{R}^+$ and any maximal tree T (not necessarily binary) $T(\alpha)$ exists. Moreover by using Lemma 3.6.3 $T(\alpha)$ can be defined as:

$$T(\alpha) = \left\{ t \in T : \ \mathcal{R}_\alpha(T_s) < \mathcal{R}_\alpha(s), \text{ for all non-terminal node } s \in T \setminus \widetilde{T} \text{ such as } t \in T_s \right\}.$$

Furthermore, let notice that, even if α can take any value in $\mathbb{R}^+$, the sequence of optimal subtrees $T_{\max}(\alpha)$ is finite (that is because the set of pruned subtrees of $T_{\max_s}$ is finite). Consider a value of α and the optimal subtree $T(\alpha)$. When α increases, $T(\alpha)$ continues to be the smallest optimally subtree of T until reaching a certain value denoted by α'. At this jump value, another subtree becomes the smallest optimal subtree: $T(\alpha') \neq T(\alpha)$. Similarly, when α increases with $\alpha > \alpha'$, $T_{\max}(\alpha')$ continues to be the smallest optimally of $T_{\max}$ until reaching another value $\alpha'' > \alpha'$.

So the idea of the minimal cost-pruning algorithm is to make α increase and record each jump value α and each associated optimal subtree $T(\alpha)$.
Nevertheless, for any value of α, seeking for $T(\alpha)$ through the whole set of pruned subtrees of T is computationally expensive. The following result enables to solve this problem.

Lemma 3.6.4. *Let consider a decision tree T with root t_1 that is non-necessarily binary and α and α' two positive real numbers.*

(i) If $\alpha \leq \alpha'$, then $T(\alpha) \succeq T(\alpha')$;

(ii) If $T(\alpha) \preceq T' \preceq T$, then $T(\alpha) = T'(\alpha)$.

Proof . We first demonstrate statement (i). If T is trivial $T = \{t_1\}$, then $\{t_1\} = T(\alpha) = T(\alpha')$. So the result is obvious.
Suppose that the result is true for all trees of depth less than $p \geq 2$. Let T be a tree of depth p and with root t_1:

$$T = \{t_1\} \bigcup_{t \in \tilde{t}_1} T_t,$$

and consider two reals α and α' such as $\alpha \leq \alpha'$. Following Lemma 3.6.2 we have either

$$T(\alpha) = \{t_0\} \quad \text{or} \quad T(\alpha) = \{t_1\} \bigcup_{t \in \tilde{t}_0} T_t(\alpha).$$

First, consider the case where $T(\alpha) = \{t_0\}$, i.e.

$$R_\alpha(t_0) \leq \sum_{t \in \tilde{t}_0} \mathcal{R}_\alpha(T_t(\alpha)).$$

According to the induction hypothesis, for any primary branch T_t with $t \in \tilde{t}_0$, we have

$$T_t(\alpha) \succeq T_t(\alpha')$$

and we can write,

$$R(T_t(\alpha)) + \alpha|\widetilde{T}_t(\alpha)| \leq R(T_t(\alpha')) + \alpha|\widetilde{T}_t(\alpha')| \leq R(T_t(\alpha')) + \alpha'|\widetilde{T}_t(\alpha')|$$

$$R_\alpha(T_t(\alpha)) \leq R_\alpha(T_t(\alpha')) \leq R_{\alpha'}(T_t(\alpha')).$$

Consequently,

$$R_{\alpha'}(t_1) \leq \sum_{t\in\tilde{t}_0} \mathcal{R}_\alpha(T_t(\alpha)) \leq \sum_{t\in\tilde{t}_0} \mathcal{R}_{\alpha'}(T_t(\alpha')),$$

So by Lemma 3.6.2, it follows that

$$T(\alpha') = \{t_0\}.$$

Therefore, since by using Lemma 3.6.2 we have

$$T(\alpha') = \{t_0\}, \quad \text{or} \quad T(\alpha') = \{t_0\} \bigcup_{t\in\tilde{t}_0} T_t(\alpha'),$$

and it follows that

$$T(\alpha) \succeq T(\alpha').$$

Now, consider the case where $T(\alpha) = \{t_1\} \bigcup_{t\in\tilde{t}_0} T_t(\alpha)$. According to Lemma 3.6.2, we have

$$T(\alpha') = \{t_0\}, \quad \text{or} \quad T(\alpha') = \{t_0\} \bigcup_{t\in\tilde{t}_0} T_t(\alpha').$$

Moreover, according to the induction hypothesis on any primary branch T_t with $t \in \tilde{t}_0$, we have

$$T_t(\alpha) \succeq T_t(\alpha'),$$

it follows

$$T(\alpha) = \{t_1\} \bigcup_{t\in\tilde{t}_1} T_t(\alpha) \succeq \{t_0\} \bigcup_{t\in\tilde{t}_0} T_t(\alpha') \succ \{t_0\}.$$

So,

$$T(\alpha) \succeq T(\alpha').$$

Now, consider statement (ii). Since $T(\alpha) \preceq T' \preceq T$, we have:

$$\{T'' : T'' \preceq T'\} \subseteq \{T''' : T''' \preceq T\}.$$

Then,

$$\begin{aligned} \min_{T''\preceq T'} R_\alpha(T'') &\geq \min_{T'''\preceq T} \mathcal{R}_\alpha(T''') \\ R_\alpha(T'(\alpha)) &\geq R_\alpha(T(\alpha)), \end{aligned}$$

where $T'(\alpha)$ is the α-optimally subtree T'. Now, since $T(\alpha) \preceq T'$, we have:

$$\begin{aligned} R_\alpha(T(\alpha)) &\geq \min_{T''\preceq T'} \mathcal{R}_\alpha(T'') \\ R_\alpha(T(\alpha)) &\geq R_\alpha(T'(\alpha). \end{aligned}$$

Consequently,

$$R_\alpha(T'(\alpha)) \;=\; R_\alpha(T(\alpha)).$$

Now, by using the definition of $T(\alpha)$, we have that $T(\alpha)$ is the smallest subtree of T that minimizes $\mathcal{R}_\alpha$. Then, as $T'(\alpha)$ is also a subtree of T, $T'(\alpha) \preceq T$, we can write

$$T(\alpha) \;\preceq\; T'(\alpha).$$

Similarly, $T'(\alpha)$ is, by definition, the smallest subtree of T' that minimizes $\mathcal{R}_\alpha$ and $T(\alpha) \preceq T'$, so

$$T'(\alpha) \;\preceq\; T(\alpha).$$

Thus, it follows that

$$T'(\alpha) \;=\; T(\alpha).$$

■

Lemma 3.6.4 highlights a key element of the minimal cost-complexity algorithm: the sequence of smallest optimal subtrees of T is embedded. This property enables avoiding the exhaustive search of $T(\alpha)$ among the whole set of pruned subtrees of T.

We are now going to explain how the minimal cost-complexity algorithm works. First of all, let α take the value $\alpha_1 = 0$, set $T_{\max} = T_0$ and search through the set of pruned subtrees of T_0 the smallest α_1-optimally subtree $T_0(\alpha_1)$. It verifies:

$$\mathcal{R}(T_0(\alpha_1)) \;=\; \mathcal{R}(T_0).$$

Since according to Lemma 3.6.1, any split of t satisfies the condition

$$R(t) \;\geq\; \sum_{t' \in \tilde{t}} \mathcal{R}(t'),$$

$T_0(\alpha_1)$ is obtained by pruning off all the branches $T_{0,t}$ of T_0 such that

$$R(t) \;=\; R(T_{0,t}),$$

where t is a node of T_0. This means removing all the branches $T_{0,t}$ that do not reduce the prediction error. $T_0(\alpha_1)$ is denoted by T_1 and

$$T_1 = \left\{ t \in T_0 :\; \mathcal{R}(T_{0,s}) < \mathcal{R}(s),\; \text{for all } s \in T_0 \setminus \widetilde{T}_0 \text{ such as } t \in T_{0,s} \right\}.$$

T_1 is the first subtree of the sequence of optimal subtrees of $T_{\max}$.

If $T_1 = \{t_1\}$ (i.e. T_1 is trivial), then the extraction of the sequence of optimal subtrees is completed and the sequence is simply

$$T_{\max} \;=\; T_0 \succ T_1 \;=\; \{t_1\}.$$

Otherwise, $T_1 \neq \{t_1\}$ and let us find the second optimal subtree of the sequence. To this end, for any non terminal node t of T_1, compute

$$g_1(t) = \frac{\mathcal{R}(t) - \mathcal{R}(T_{1,t})}{|\widetilde{T}_{1,t}| \ - \ 1}, \tag{3.18}$$

which corresponds to the value of α at which $T_1 \setminus T_{1,t}$ becomes preferable than T_1, that means

$$\mathcal{R}_\alpha(T_{1,t}) \ = \ \mathcal{R}_\alpha(t).$$

Select the nodes that minimize $g_1(.)$ and denote by α_2 the corresponding minimum value. Observe that $\alpha_2 > \alpha_1 = 0$. The nodes which minimize $g_1(.)$ are called the weakest links. They are the nodes that produce the smallest reduction in prediction error $\mathcal{R}$. Then, $T_1(\alpha_2)$ is obtained by pruning T_1 in the weakest links. $T_1(\alpha_2)$ corresponds to the second tree of the sequence of optimal subtrees and is denoted T_2:

$$T_2 = \left\{ t \in T_1 : \ \mathcal{R}_{\alpha_2}(T_{1,s}) < \mathcal{R}_{\alpha_2}(s), \text{ for all } s \in T_1 \setminus \widetilde{T}_1 \text{ such as } t \in T_{1,s} \right\}.$$

If $T_2 \neq \{t_1\}$, the same process is repeated on T_2 and so on until the trivial tree is obtained. More generally, for any $k \geq 0$, let denote T_k the k-th subtree of the sequence. To find the $(k+1)$-th subtree T_{k+1} of the sequence, compute,

$$g_k(t) = \frac{\mathcal{R}(t) - \mathcal{R}(T_{kt})}{|\widetilde{T}_{kt}| \ - \ 1}, \tag{3.19}$$

for any non terminal node t of T_k. Then select:

$$\alpha_{k+1} \ = \ \min_{t \in T_k \setminus \tilde{T}_k} g_k(t).$$

We have $\alpha_{k+1} > \alpha_k$ and T_{k+1} is obtained by pruning T_k in the weakest links, that means removing all the branches $T_{k,t}$ of T_k that satisfies:

$$\mathcal{R}_{\alpha_{k+1}}(T_{k,t}) \ = \ \mathcal{R}_{\alpha_{k+1}}(t).$$

T_{k+1} is then defined by

$$T_{k+1} = \left\{ t \in T_k : \ \mathcal{R}_{\alpha_{k+1}}(T_{k,s}) < \mathcal{R}_{\alpha_{k+1}}(s), \text{ for all } s \in T_k \setminus \widetilde{T}_k \text{ such as } t \in T_{k,s} \right\}.$$

At the end of this process, we have a decreasing sequence sequence of optimal subtrees

$$T_1 \prec T_2 \prec \ldots \prec T_K = \{t_1\}$$

and a increasing sequence of values for the parameter α

$$0 = \alpha_1 < \ldots < \alpha_K.$$

Furthermore, since $T_1(\alpha_2) \preceq T_1 = T_0(\alpha_1) \preceq T_0 = T_{\max}$, it follows by statement (i) in Lemma 3.6.4 that

$$T_{\max}(\alpha_2) \preceq T_1 = T_0(\alpha_1) \preceq T_0.$$

So by using statement (ii) in Lemma 3.6.4,

$$T_0(\alpha_2) = T_1(\alpha_2) = T_2.$$

Then, by repeating for any k such as $1 \leq k \leq K$, we have,

$$T_0(\alpha_k) = T_{k-1}(\alpha_k) = T_k.$$

Consequently, for any k with $1 \leq k \leq K$,

$$T_k = T_0(\alpha_k) = T_{\max}(\alpha_k),$$

where $T_{\max}(\alpha_k)$ is the smallest α_k-optimally subtree of $T_{\max}$.

Finally, for any k with $1 \leq k \leq K$, let consider $\alpha \geq \alpha_k$. As α increases, α_{k+1} is by definition the first jump value at which T_k is not optimal, i.e. $T_k \neq T_{k+1}$. As a consequence, if $\alpha \in [\alpha_k, \alpha_{k+1}[$,

$$T_{\max}(\alpha) = T_k.$$

This last result ensures that the sequence of optimal subtrees contains all the information in the sense that for each α, the smallest optimal subtree with respect to $\mathcal{R}_\alpha$ belongs to the sequence.

The resolution of the pruning problem is then based on the following theorem.

Theorem 3.6.1. *For any nontrivial tree T non necessarily binary and with root t_1, there exist a unique increasing sequence*

$$0 = \alpha_1 < \ldots < \alpha_K, \quad K \in \mathbb{N}^\star,$$

and a unique sequence of nested subtrees of T

$$T \succeq T_1 \succ \ldots \succ T_K = \{t_1\},$$

such that for $1 \leq k \leq K$,

$$\alpha_{k+1} = \min_{t \in T_k \setminus \widetilde{T}_k} \frac{\mathcal{R}(t) - \mathcal{R}(T_{kt})}{|\widetilde{T}_{kt}| - 1},$$

and

$$T(\alpha) = \begin{cases} T_k, & \textit{if} \quad \alpha \in [\alpha_k, \alpha_{k+1}[, \ 1 \leq k < K \\ T_K, & \textit{if} \quad \alpha \geq \alpha_K. \end{cases}$$

Then this theorem describes the principle of the minimal cost-pruning algorithm which consists in extracting from the whole set of pruned subtrees of $T_{\max}$ a sequence of optimal subtrees $(T_k)_{1 \leq k \leq K}$. Moreover as mentioned above, this sequence contains all the subtrees of $T_{\max}$ optimal with respect to the penalized criterion $\mathcal{R}_\alpha$.

Finally, the algorithm selects through the sequence $(T_k)_{1 \leq k \leq K}$ the subtree that minimizes the prediction error $\mathcal{R}(T, \mathcal{T}_q)$ on the validation sample $\mathcal{T}_q$, that is

$$T_{\widehat{k}} \ = \ \underset{T_k, 1 \leq k \leq K}{\operatorname{argmin}} \mathcal{R}(T_k, \mathcal{T}_q).$$

The final tree $T_{\widehat{k}}$ is then the subtree that gives the best trade-off between prediction and complexity.

The following section provides an illustration of the construction of the nested sequence of optimal pruned subtrees in a simple example.

Illustration of the computation of the generalized minimal cost-complexity algorithm:

We consider a simple binary classification problem. Figure 3.15 displays the maximal CARTGV tree $T_{\max}$ obtained after applying CARTGV on a balanced training sample of 200 observations. Here we describe the elaboration of the sequence of optimal pruned subtrees and the associated sequence of values for α.

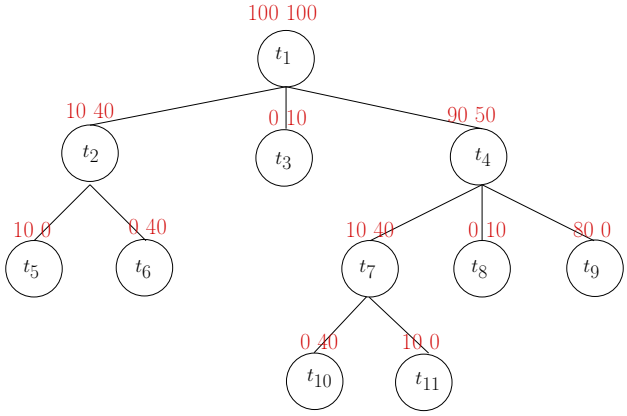


Figure 3.15: A maximal CARTGV tree $T_{\max}$. The two numbers indicated above each node are the number of observations in each class.

First of all, take $\alpha_1 = 0$. Since all leaves are homogeneous, $T_1 = T_{\max}$ (indeed merging nodes would make the prediction error increase). So, the first optimal subtree of the sequence is $T_{\max}$. Next, Table 3.8 gives the values of $\mathcal{R}(t)$, $\mathcal{R}(T_{1,t})$ and $g_1(t)$, for any non-terminal node of T_1. The weakest links are the nodes t_2 and t_7 (i.e. they minimize g_1). Thus, $\alpha_2 = 1/20$ and T_2 is obtained by removing t_2 and t_7 from T_1, that is

$$T_2 = T_1 \setminus \{T_{1,t_2} \cup T_{1,t_7}\}.$$

T_2 is displayed in Figure 3.16.

$t \in T_1 \setminus \widetilde{T}_1$	$\mathcal{R}(t)$	$\mathcal{R}(T_{1,t})$	$g_1(t)$
t_1	1/2	0	1/2
t_2	1/20	0	**1/20**
t_4	1/4	0	1/4
t_7	1/20	0	**1/20**

Table 3.8: Research of the weakest link in T_1.

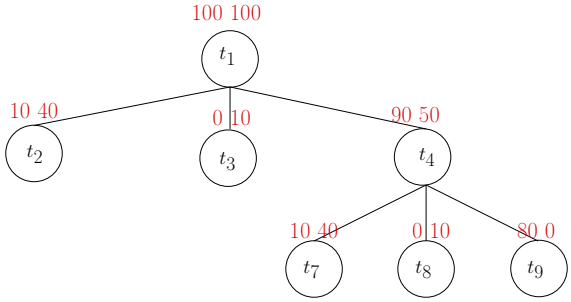


Figure 3.16: Tree T_2 obtained by pruning T_1 in t_2 and t_7: $T_2 = T_1 \setminus \{T_{1,t_2} \cup T_{1,t_7}\}$.

Next, we repeat the same process with T_2. Table 3.9 gives the values of $\mathcal{R}(t)$, $\mathcal{R}(T_{2,t})$ and $g_2(t)$, for any non-terminal node of T_2. The weakest link is t_1. So, $\alpha_3 = 1/10$ and $T_3 = T_2 \setminus T_{2,t_1} = \{t_1\}$.

$t \in T_2 \setminus \widetilde{T}_2$	$\mathcal{R}(t)$	$\mathcal{R}(T_{2,t})$	$g_2(t)$
t_1	1/2	1/10	2/25
t_4	1/4	1/20	**1/10**

Table 3.9: Research of the weakest link in T_2

Since $T_3 = \{t_1\}$, the process is completed. The whole sequence of optimal subtree is

$$T_{\max} = T_1 \prec T_2 \prec T_3 = \{t_1\}.$$

and the corresponding increasing sequence for α is

$$\alpha_1 = 0 < \alpha_2 = 1/20 < \alpha_3 = 2/25.$$

The final tree is next selected in the sequence $\{T_{\max}, T_2, t_1\}$.

3.6.4 CARTGV: additional information about the numerical experiments

This section provides additional results and figures about the simulation studies used to assess performance of CARTGV.

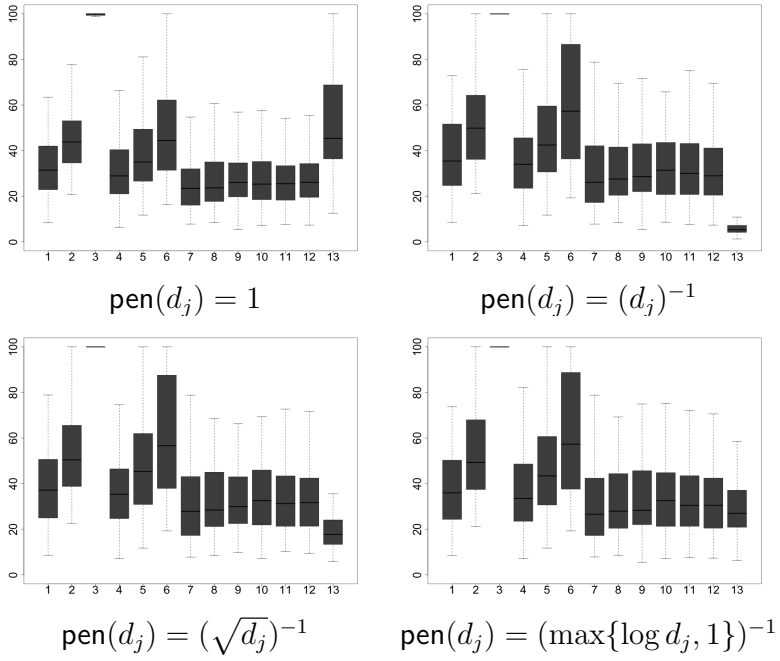


Figure 3.17: Distribution of the GRI score according to the penalty function in experiment 3.

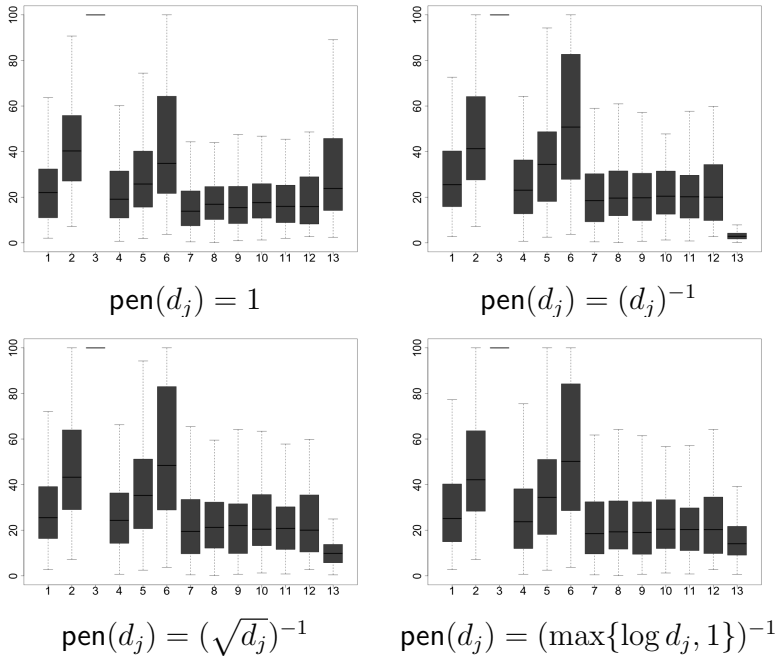


Figure 3.18: Distribution of the GSI score according to the penalty function in experiment 3.

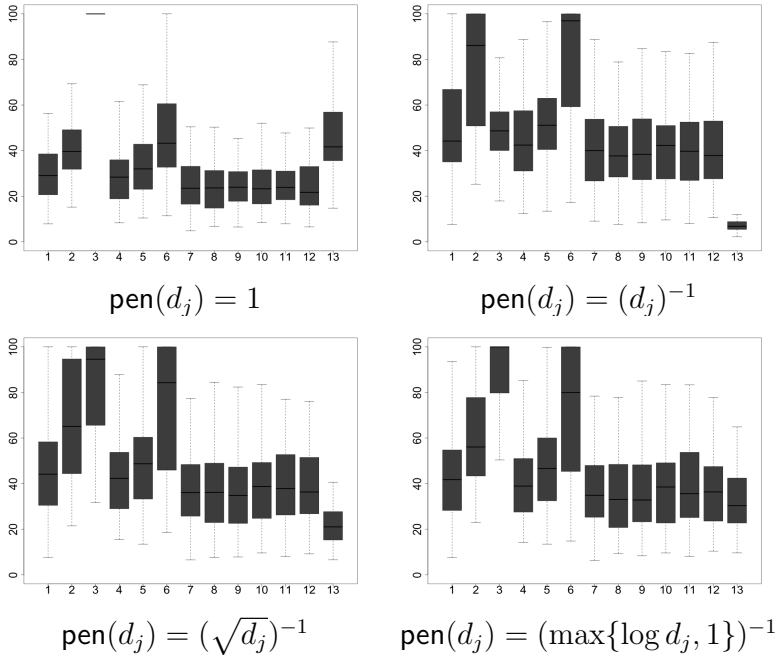


Figure 3.19: Distribution of the GRI score according to the penalty in experiment 4.

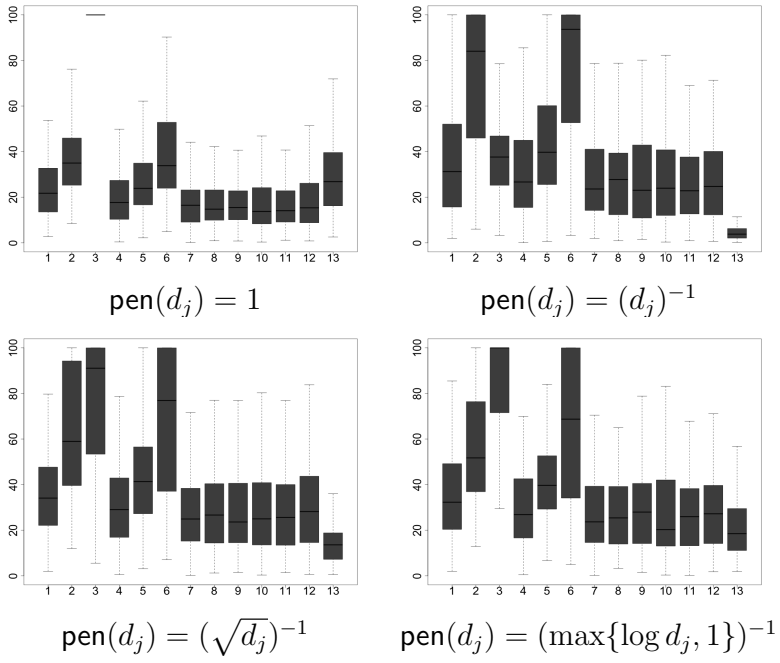


Figure 3.20: Distribution of the GSI score according to the penalty in experiment 4.

	Experiment 3	Experiment 4
AUC	0.66 (0.04)	0.67 (0.04)
Err	0.35 (0.04)	0.35 (0.04)
Depth	3 (1.27)	3 (1.48)
Leaves	6 (3.12)	6 (3.83)

Table 3.10: CART results in Experiments 3 and 4.

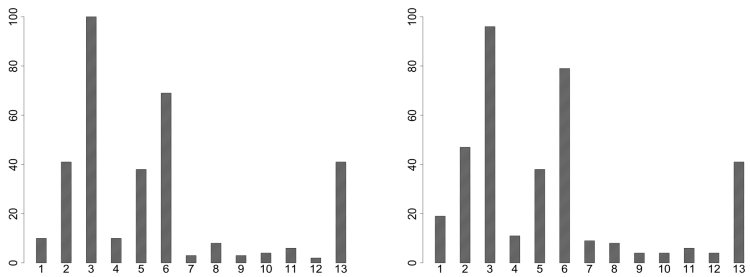


Figure 3.21: CART group selection frequencies in experiment 3 (left) and experiment 4 (right). We recall that only the first six groups are relevant (in this order: 3, 6, 2, 4, 1, 3).

Chapter 4

Statistical Analysis of a robust hierarchical clustering algorithm

Abstract. In the clustering context, we consider a classical model in which clusters are defined as the connected components of the level sets of the density of the observations. Moreover, we assume that outliers, which are observations that do not belong to any cluster, can be included. In this context, the goal is to identify accurately the clusters. To that end, we propose a clustering algorithm which is based on the single linkage hierarchical clustering algorithm. We prove that, under classical assumptions on the model, the proposed procedure identifies correctly each cluster with large probability. A comparison of our method with classical clustering algorithms on simulated data is also presented.

This chapter introduces a work conducted in collaboration with Nicolas Klutchnikoff (Senior lecturer at the University of Rennes 2) and Laurent Rouvière.

Contents

4.1 Introduction . . . 118
4.2 Mathematical framework . . . 119
4.2.1 Generative model . . . 119
4.2.2 Basic assumptions . . . 119
4.2.3 Sparsity assumption . . . 121
4.2.4 Examples . . . 121
4.2.5 Clustering risk . . . 123
4.3 Agglomerative clustering . . . 124
4.4 Simulation study . . . 129
4.4.1 Description of the models . . . 129
4.4.2 Results . . . 130
4.5 Proofs . . . 137

4.5.1 Technical lemmas 137
4.5.2 Proof of proposition 4.3.1 142
4.5.3 Proof of Theorem 4.3.1 143
4.6 A brief review of the Hausdorff measure 143

4.1 Introduction

In unsupervised learning, clustering refers to a very broad set of tools which aim to find a partition of the data into dissimilar groups so that the observations within each group are quite similar to each other. Considered as one of the most important questions in unsupervised learning, there is a vast literature on this paradigm (Hartigan, 1975; Jain & Dubes, 1988; Duda et al., 2012). Moreover, a lot of various clustering methods have been developed, such as the k-means algorithm (MacQueen, 1967), the hierarchical clustering methods (Johnson, 1967), the spectral clustering algorithms (Von Luxburg, 2007) or the model-based clustering approaches (McLachlan & Basford, 1988). Clustering plays an important role in explanatory data analysis and has been used in many fields including pattern recognition (Satish & Sekhar, 2006), image analysis (Filipovych et al., 2011), document retrieval, bioinformatics (Yamanishi et al., 2004; Zeng et al., 2012), data compression. Overall, clustering tools are often used to help users understand the data structure. Furthermore, with the massive increase in the amount of collected and stored data, clustering methods can also be used as dimensionality reduction techniques (Yengo et al., 2014).

Here, we consider a model close to the classical framework of Hartigan (1975) who defines clusters as the connected components of the level sets of the density of the observations. Moreover, we assume that outliers, which are observations that do not belong to any cluster, can be included. In this context, many authors have studied theoretical properties of various clustering algorithms (Arias-Castro et al., 2011; Maier et al., 2009). For instance, Arias-Castro (2011) proves that under assumptions about the cluster separability the hierarchical clustering algorithm named single linkage is efficient when there is no outlier. To deal with outliers, he also proposes to modify the single linkage algorithm and shown the efficiency of this new method under quite restrictive assumptions about the distance between clusters and outliers.

In this paper, we propose a modified version of the single linkage algorithm that can deal with outliers under weaker assumptions than those assumed by Arias-Castro (2011). We called this new method the robust single linkage algorithm. Then, under some classical regularity and sparsity assumptions we obtain the rate of convergence for the clustering risk of the proposed algorithm. Next, we assess the performances of this new approach through simulation studies where the algorithm is compared to some classical clustering methods

(single linkage, k-means, spectral clustering).

The paper is organized as follows. In Section 4.2, we first introduce the mathematical framework, the model assumptions and we define the clustering risk for the proposed algorithm. Section 4.3 describes the algorithms single linkage clustering and the robust single linkage. In this section, a simple example is used to prove that with outliers the single linkage algorithm often fails to recover the true clusters. Next, the consistency of the robust single linkage algorithm is demonstrated under regularity and sparsity assumptions. Finally, Section 4.4 is devoted to highlighting the performances of the proposed algorithms in comparison with the single linkage algorithm, the k-means method and the spectral clustering through several synthetic data sets. The various proofs are gathered at the end of the paper, in Section 4.5.

4.2 Mathematical framework

4.2.1 Generative model

In this section, our purpose is to construct a model that allows to consider a large range of data. We mainly follow two goals: the data can locally lie in low-dimensional structure and outliers can be included in the data. More precisely, we assume that we are given n independent $\mathbb{R}^d$-valued random variables $X_1, \dots, X_n$ randomly drawn from a distribution $\mathbb{P}$ with support $S \subseteq \mathbb{R}^d$. We also assume that $\mathbb{P}$ can be written as a mixture of $M+1$ distribution $\mathbb{P}_0, \dots, \mathbb{P}_M$ in such a way:

$$\mathbb{P} = \varepsilon \mathbb{P}_0 + (1-\varepsilon) \sum_{i=1}^{M} \gamma_i \mathbb{P}_i,$$

where $0 < \varepsilon < 1$, $\gamma_i > 0$ for any $i = 1, \dots, M$ and $\sum_{i=1}^{M} \gamma_i = 1$. In this decomposition, $\mathbb{P}_0$ denotes the distribution of the outliers whereas $\mathbb{P}_i$ denotes the distribution of the data that belong to the actual i-th cluster. In the following, we propose assumptions that allow us to clearly distinguish between actual data and outliers on the one hand and between the different clusters on the other hand. The main idea that guided our choices of assumptions is the following: clusters consist of groups of data that are dense in some sense whereas outliers are sparse.

4.2.2 Basic assumptions

We are given M subsets of S, namely $S_1, \dots, S_M$ and we denote by S_0 the complement of $\bigcup_{i=1,\dots,M} S_i$ into S. For $i \in [\![M]\!] = [\![1, M]\!] = [1, M] \cap \mathbb{Z}$, the set S_i is called the *actual i-th cluster*. We denote by δ the *minimal distance* between actual clusters, that is: $\delta = \min\{\|x - y\| : x \in S_i, y \in S_j, 1 \leq i < j \leq M\}$ where $\|\cdot\|$ denotes the Euclidean norm in $\mathbb{R}^d$. We assume that:

(P1) For any $i \in [\![0, M]\!]$, we have $\mathbb{P}_i(S_i) = 1$.

(G1) Each set S_i is compact and connected. Moreover $\delta > 0$.

Note that **(P1)** ensures that the distribution $\mathbb{P}_i$ has its support included into S_i while **(G1)** guarantees that the actual clusters are disjoint and well-separated. This implies that the model is identifiable since the decompostion of $\mathbb{P}$ is then unique. Moreover, the compactness of each S_i guarantees that its diameter $\Delta_i = \max\{\|x-y\| : x \in S_i, y \in S_i]$ is finite. Here and later, $\mathcal{H}^s$ denotes the s-dimensional Hausdorff outer measure. For the convenience of the reader, basic properties of Hausdorff measures are given in Section 4.6. For a more detailed review of this topic, we refer the reader to Evans & Gariepy (2015). For $i \in [\![M]\!]$, we denote by $s_i = \dim_H(S_i)$ the Hausdorff-dimension of S_i. We assume that:

(P2) There exits $\kappa_0 \geq 1/\mathcal{H}^d(S_0)$ such that, for any $A \subseteq S_0$, we have $\mathbb{P}_0(A) \leq \kappa_0 \mathcal{H}^d(A)$.

(P3) There exists $\kappa_i \geq \mathcal{H}^{s_i}(S_i)$ such that, for any $i \in [\![M]\!]$ and $A \subseteq S_i$, we have $\mathbb{P}_i(A) \geq \kappa_i^{-1} \mathcal{H}^{s_i}(A)$.

Since $\delta > 0$, there exists a ball with diameter less than δ included in S_0. This implies in particular that $\mathcal{H}^d(S_0)$ is positive which also entails that the Hausdorff dimension of S_0 is d. Keeping this in mind, we can reformulate **(P2)** as follows: the distribution of the outliers, namely $\mathbb{P}_0$, is assumed to be absolutely continuous with respect to the Lebesgue measure $\mathcal{H}^d$ with bounded Radon-Nikodym derivative. This implies that in some sense the distribution of the outliers is not dense. On the opposite, **(P3)** ensures that $\mathbb{P}_i$ is quite dense on each actual cluster S_i. Note in particular that $\mathbb{P}_i$ can be singular with respect to the Lebesgue measure $\mathcal{H}^d$. In our mind these properties highlight the main difference between outliers and actual data (that is, data that lie into actual clusters): actual data are distributed in a dense way into their support whereas outliers are diffuse. The following assumption is natural regarding (4.6.1) in Section 4.6:

(G2) There exists $\kappa_c \geq 1$ such that, for any $i \in [\![M]\!]$, $x \in S_i$ and $0 < r \leq \Delta_i$,

$$\kappa_c^{-1} \leq \frac{\mathcal{H}^{s_i}(S_i \cap B(x, r))}{\eta(s_i) r^{s_i}} \leq \kappa_c,$$

where $B(x, r)$ stands for the euclidean ball centered at x with radius r and for any $s > 0$ we define $\eta(s) = \pi^{s/2}\Gamma(1 + s/2)$. Here Γ stands for the usual gamma function and $\eta(s)$ generalizes, to non-integer parameters $s > 0$, the volume of the unit ball in dimension s.

Assumption **(G2)** can be viewed as a regularity assumption that roughly implies that the sets S_i, with $i \in [\![M]\!]$, are not too narrow. A similar assumption is made by Arias-Castro (2011) in a more restrictive model. Note also that, if S_i is a submanifold that satisfies a *reach* condition, then **(G2)** is automatically fulfilled (see Biau et al., 2007, and references therein). This assumption also implies, by taking $r = \Delta_i$, that $0 < \mathcal{H}^{s_i}(S_i) < +\infty$.

4.2.3 Sparsity assumption

To state the next assumptions we introduce some notations used throughout the paper. Set

$$\gamma_* = \min\{\gamma_i : i \in [\![M]\!]\}, \gamma^* = \max\{\gamma_i : i \in [\![M]\!]\}, \ \not\gamma = \gamma_* - \gamma^*/2$$
$$\kappa^* = \max\{\kappa_i : i \in [\![M]\!]\}, \eta_*(d) = \min\{\eta(s) : 0 \leq s \leq d\} = \min(\eta(d), 1).$$

(S1) Assume that $\gamma^* \leq 2\gamma_*$ and $0 < \varepsilon < \not\gamma/(1 + \not\gamma)$.

(S2) Set $\mathfrak{a} = (1-\varepsilon)\gamma_*(\kappa^*\kappa_c)^{-1}\eta_*(d)$ and assume that

$$\delta > r_n \quad \text{where} \quad r_n = \left(\frac{\log n}{\mathfrak{a} n}\right)^{\frac{1}{d}}$$

Assumption **(S1)** relates to the size of the clusters. It implies that these sizes are of the same order and that the number of outliers is smaller than the number of points in the S_i's. Assumption **(S2)** ensures that minimal distance between the actual clusters cannot be too small, it is a classical assumption in clustering. The constant $\mathfrak{a}$ depends on both the density in the S_i and the proportion of outliers. It increases when $1/\kappa_i$ (the density in the S_i) increases or when the proportion of outliers ε decreases. For large values of $\mathfrak{a}$, the supports S_i can get closer. Thus, if $\mathfrak{a}$ is large then Assumption **(S2)** is weak.

4.2.4 Examples

We consider three clustering problems in $\mathbb{R}^2$. Each model is defined based on the design introduced Section 4.2.1. Moreover, for any $i \in [\![0, M]\!]$, the distribution $\mathbb{P}_i$, which is the distribution of the data belonging to the i-th cluster, is uniform over its support S_i. So formally, for any $i \in [\![0, M]\!]$ and $A \subset S_i$, we have

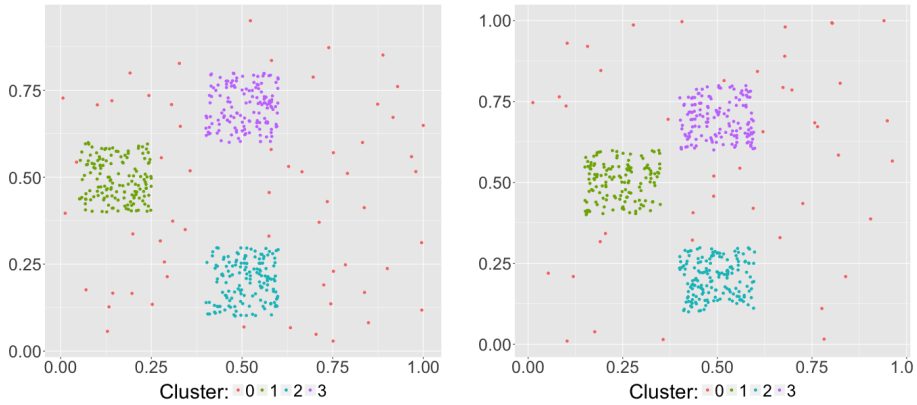
$$\mathbb{P}_i(A) = \frac{\mathcal{H}^{s_i}(A)}{\mathcal{H}^{s_i}(S_i)},$$

where $s_i = \dim_H(S_i)$ denotes the Hausdorff-dimension of S_i.

The three models, which involve various shapes of clusters, are described as follows. For each model, two scenarios are introduced so as to make the minimal distance δ between clusters vary. For the sake of clarity, we call respectively these scenarios the *easy case* (i.e. clusters are not too close, so δ is not too small) and the *tricky case* (i.e. clusters are closer, so δ is smaller).

The "squares" model (Auray et al., 2015): there are $M = 3$ compact clusters. The support of $\mathbb{P}$ is $S = [-1, 1]^2$. For any $i = 1, \ldots, 3$, the support S_i is the square $[a_i - r; a_i + r] \times [b_i - r; b_i + r]$ with $r > 0$. We fix $\gamma_1 = \gamma_2 = \gamma_3 = 1/3$, $b_1 = 0.5$, $a_2 = 0.5$, $b_2 = 0.2$,

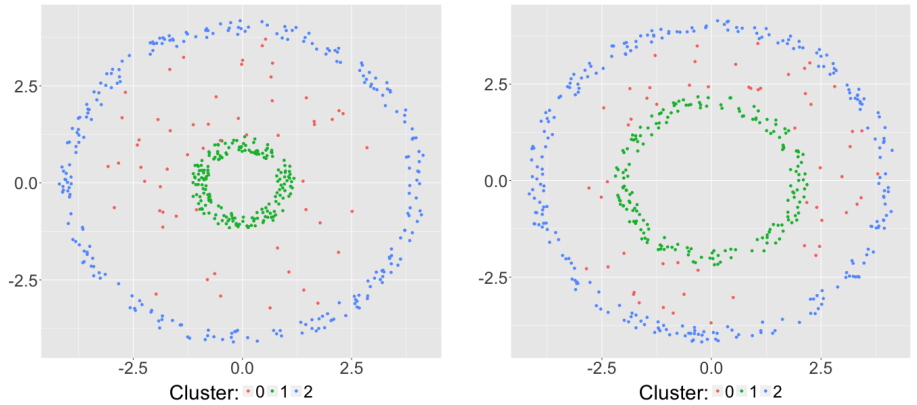
$a_3 = 0.5$ and $b_3 = 0.2$ and $r = 0.1$. Two values for a_1 are considered: $a_1 = 0.15$ (the *easy case*, $\delta = 0.15$) and $a_1 = 0.25$ (the *tricky case*, $\delta = 0.05$). Figure 4.1 provides an illustration of these scenarios.



(a) (b)

Figure 4.1: A sample of $n = 500$ observations for the "squares" model with $\varepsilon = 0.1$, $\delta = 0.15$ (a) and $\delta = 0.05$ (b).

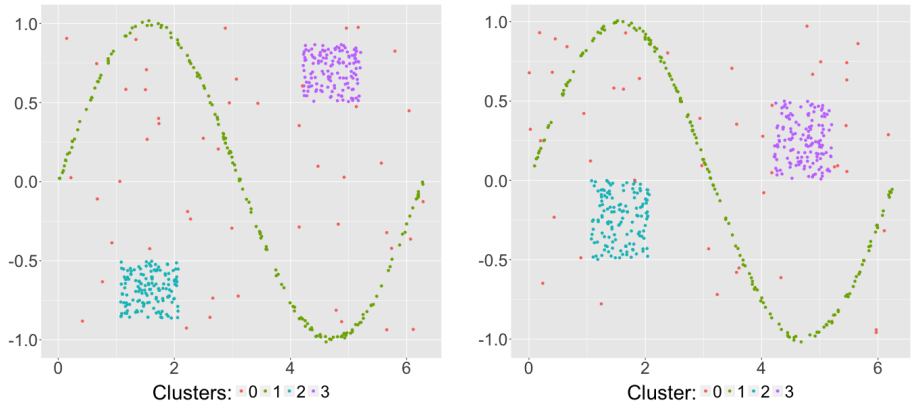
The "concentric-circles" model (Ng et al., 2002): this model includes 2 clusters which are represented by 2 concentric circles. For any $i = 1, 2$, the support S_i is the ring $\mathcal{C}(r_i + w, r_i - w)$, where for $r \geq w > 0$, $\mathcal{C}(r + w, r - w)$ denotes the set between the two circles with center 0 and radius $r + w$ and $r - w$. The support S of the distribution $\mathbb{P}$ is defined by the ring $\mathcal{C}(r_2 + w, r_1 - w)$. We fix $\gamma_1 = 0.4$, $\gamma_2 = 0.6$, $r_2 = 4$, $w = 0.2$ and we consider two values for r_1: $r_1 = 1$ (the *easy case*, $\delta = 2.6$) and $r_1 = 2$ (the *tricky case*, $\delta = 1.6$). An illustration is provided in Figure 4.2.



(a) (b)

Figure 4.2: A sample of $n = 500$ observations for the "concentric-circles" model with $\varepsilon = 0.1$, $\delta = 2.6$ (a) and $\delta = 1.6$ (b).

The "sine" model (Giulini, 2016): this model includes 3 clusters with various shapes. The support of $\mathbb{P}$ is $S = [0, 2\pi] \times [-1, 1]$. The first cluster is tight and represents the sine curve, so $S_1 = \{(x, y) : x \in [0, 2\pi], y = \sin(x)\}$. The two other are compact clusters and for any $i = 2, 3$, S_i is the square $[a_i, b_i] \times [c_i, d_i]$, with $0 \leq a_i < b_i \leq 2\pi$ and $-1 \leq c_i < d_i \leq 1$. We fix $\gamma_1 = \gamma_2 = \gamma_3 = 1/3$, $a_1 = \frac{\pi}{2} - 0.5$, $b_1 = \frac{\pi}{2} + 0.5$, $a_2 = \frac{3\pi}{2} - 0.5$, $b_2 = \frac{3\pi}{2} - 0.5$. We set $c_1 = -d_2$ and $d_1 = -c_2$ and we consider two pairs of values for c_1 and d_1: $(c_1, d_1) = (-\frac{\sqrt{3}}{2}, -0.5)$ (the *easy case*, $\delta = \frac{\sqrt{5\pi}}{6} - 0.5$) and $(c_1, d_1) = (-0.5, 0)$ (the *tricky case*, $\delta = \frac{\sqrt{\pi}}{2} - 0.5$). Figure 4.3 displays these two situations.



(a) (b)

Figure 4.3: A sample of $n = 500$ observations for the "sine" model with $\varepsilon = 0.1$, $\delta = \frac{\sqrt{5\pi}}{6} - 0.5$ (a) and $\delta = \frac{\sqrt{\pi}}{2} - 0.5$ (b).

These models satisfy assumptions **(P1)-(S2)** with $\kappa_c = 4$, $\kappa_0 = [\mathcal{H}^d(S) - \sum_{i=1}^{M} \mathcal{H}^d(S_i)]^{-1}$ and $\kappa_i = \mathcal{H}^{s_i}(S_i)$ for any $i \in [\![M]\!]$ and $d = 2$. Remark that in these models the Hausdorff dimension $s_i = \dim_H(S_i)$ of S_i equals 2 (and therefore agrees with the area of S_i), excepted when S_i is the sine curve. In this case, the Hausdorff dimension is 1, $\mathcal{H}^d(S_i) = 0$ and $\mathcal{H}^1(S_i)$ is the length of the sine curve.

4.2.5 Clustering risk

Given the n-sample $X_1, \ldots, X_n$, the objective is to recover the clusters S_i. More precisely, we aim at finding a clustering procedure that group together the data that lie within the same set S_i, for each $i \in [\![M]\!]$. The outliers, that belong to the set S_0, can be affected to any other group or garbabge into a specific group by the procedure. In other words, a clustering procedure consists of finding a partition of the data into $M + 1$ clusters $\mathcal{X}_0, \ldots, \mathcal{X}_M$ such that, with high probability

$$X_{[\![n]\!]} \cap S_i \subseteq \mathcal{X}_{\pi(i)}$$

where π is a permutation of $[\![M]\!]$. Here, for any $I \subseteq [\![n]\!]$, we consider $X_I = \{X_i \colon i \in I\}$, so that $X_{[\![n]\!]}$ denotes the data. Moreover, the clusters $\mathcal{X}_1, \ldots, \mathcal{X}_M$ are not empty and satisfy

$$\bigcup_{i=0}^{M} \mathcal{X}_i = X_{[\![n]\!]} \quad \text{and} \quad \forall\, i \neq j,\ \mathcal{X}_i \cap \mathcal{X}_j = \emptyset.$$

The cluster $\mathcal{X}_0$ contains the observations which are considered as the outliers by the clustering procedure. More precisely this cluster contains all the observations that are not assigned to one of the groups $\mathcal{X}_1, \ldots, \mathcal{X}_M$ by the clustering procedure.

We measure the performances of a clustering procedure $\mathcal{X} = \{\mathcal{X}_0, \ldots, \mathcal{X}_M\}$ by using the *clustering risk* defined as

$$\mathcal{R}_n(\mathcal{X}) = \mathbb{P}(\forall \pi \in \Pi_M, \exists i \in [\![M]\!],\ X_{[\![n]\!]} \cap S_i \nsubseteq \mathcal{X}_{\pi(i)}), \tag{4.1}$$

where Π_M denotes the set of all permutations of $[\![M]\!]$. This quantity is the probability that the clustering procedure does not correctly recover one subset of observations for at least on S_i.

4.3 Agglomerative clustering

The proposed model is closed to the framework of Hartigan (1975) who defines clusters as the connected components of the level sets of the density of the observations. This amounts to saying that clusters represent high density regions of the data separated by low density regions. In this context, many authors have studied theoretical properties of various clustering algorithms. For instance, Maier et al. (2009) and Arias-Castro (2011) prove that algorithms based on pairwise distances (k-nearest neighbor graph, spectral clustering...) are efficient as soon as these connected components are separated enough. Hierarchical clustering algorithms are also known to perform well in this situation. They are defined as follows.

Let $r \geq 0$, two observations x and y in $\mathbb{R}^d$ are said to be r-connected if and only if

$$B\left(x, \frac{r}{2}\right) \cap B\left(y, \frac{r}{2}\right) \neq \emptyset$$

or equivalently

$$d(x, y) \leq r,$$

where $B(x, r)$ is the closed euclidean ball centered at $x \in \mathbb{R}^d$ with radius r and d denotes the Euclidean distance. Let $M(r)$ denotes the number of connected components of

$$\bigcup_{i=1}^{n} B\left(X_i, \frac{r}{2}\right).$$

For any $A \subset \mathbb{R}^d$ and any $r \geq 0$, let

$$B(A, r) = A \oplus B(0, r) = \{x + y : x \in A, y \in B(0, r)\},$$

where $\oplus$ is the *Minkowski addition*. We said that A is r-connected if $B(A, r/2)$ is a connected set. With these notations, the set $B(X_{[\![n]\!]}, r/2)$ may be split into $M(r)$ disjoint connected components $B_m(X_{[\![n]\!]}, r/2)$, $m \in [\![M(r)]\!]$ such as

$$B\left(X_{[\![n]\!]}, \frac{r}{2}\right) = \bigcup_{m=1}^{M(r)} B_m\left(X_{[\![n]\!]}, \frac{r}{2}\right) \quad \text{and} \quad B_m\left(X_{[\![n]\!]}, \frac{r}{2}\right) \cap B_{m'}\left(X_{[\![n]\!]}, \frac{r}{2}\right) = \emptyset,$$

for $m \neq m'$. In this setting, a fixed radius $r/2$ provides a partition of $X_{[\![n]\!]}$ into $M(r)$ clusters defined by

$$\mathcal{Y}_m(r) = B_m\left(X_{[\![n]\!]}, \frac{r}{2}\right) \cap X_{[\![n]\!]}, \quad m \in [\![M(r)]\!].$$

We denote by $\mathcal{Y}(r)$ this partition

$$\mathcal{Y}(r) = \{\mathcal{Y}_m(r) \colon m \in [\![M(r)]\!]\}.$$

The distance between two r-connected components $\mathcal{Y}_m(r)$ and $\mathcal{Y}_{m'}(r)$ is defined as the distance between the two closest members

$$d'(\mathcal{Y}_m(r), \mathcal{Y}_{m'}(r)) = \inf\{d(X_k, X_l) \colon X_k \in \mathcal{Y}_m(r),\, X_l \in \mathcal{Y}_{m'}(r)\}.$$

Agglomerative clustering consists in recursively grouping clusters. At the beginning, we consider $\rho_0 = 0$ and

$$\mathcal{Y}(\rho_0) = \{\mathcal{Y}_m(\rho_0),\, m \in [\![M(\rho_0)]\!]\}.$$

Next the two closest clusters according to d' are merged. Denote by $\rho_1 > 0$ the distance between the two closest clusters in $\mathcal{Y}(\rho_0)$ and define the set $\mathcal{Y}(\rho_1) = \{\mathcal{Y}_m(\rho_1), m \in [\![M(\rho_1)]\!]\}$ as the new partition obtained after merging the two closest clusters in $\mathcal{Y}(\rho_0)$. This process is then recursively repeated until all (distinct) observations belong to a single cluster. We denote by K the (random) number of iterations, observe that $K \leq n - 1$ almost surely.

Remark 4.3.1.

- *Observe that, since $\mathbb{P}(X = x) = 0$ for all $x \in S$, $M(\rho_0) = n$ almost surely and each point is treated as a singleton cluster at the first step. Moreover, in this case $K = n - 1$ almost surely.*
- *At every step k with $1 \leq k \leq K$, the new selected radius ρ_k is larger than the previous radius: $\rho_k > \rho_{k-1}$. This radius corresponds to the distance between the closest clusters belonging to $\mathcal{Y}(\rho_{k-1})$. Moreover, for any $\rho \in [\rho_k; \rho_{k+1}[$ with $0 \leq k \leq K - 1$ and $\rho_K = \infty$, we have*

$$\mathcal{Y}(\rho) = \mathcal{Y}(\rho_k).$$

- *At the end of the process, we have a sequence $\mathcal{Y}(\rho_0), \ldots, \mathcal{Y}(\rho_{K-1})$ of partitions of the data at hand. The aim is to determine how to choose one partition in this sequence. It remains to select one radius in the sequence $\rho_0, \ldots, \rho_{K-1}$.*

Single linkage clustering

Since the number of clusters is known, the classical single linkage clustering algorithm selects the radius such that the associated number of clusters is close to M. More precisely, we choose

$$\widehat{\rho}_{n,SL} \in \underset{\rho \in \{\rho_k \,:\, k \in [\![0, K-1]\!]\}}{\operatorname{argmax}} \{M(\rho) \geq M\}.$$

Observe that $\widehat{\rho}_{n,SL}$ exists as soon as each support S_i contains at least one observation. This algorithm is known to be consistent without outlier and under assumptions close to ours (Arias-Castro, 2011; Auray et al., 2015).

However, in the presence of outliers the algorithm may fail to recover the true clusters with high probability. For example, let us consider the following toy example. Assume that

$$\mathbb{P} = \frac{1-\varepsilon}{2}(\mathbb{P}_1 + \mathbb{P}_2) + \varepsilon \mathbb{P}_0$$

where $\mathbb{P}_1 = \delta_{-1}$, $\mathbb{P}_2 = \delta_1$ and $\mathbb{P}_0 = \mathcal{U}([-3, 3])$. For ε small enough, our assumptions are satisfied. However the algorithm fails with probability

$$\frac{2\mathbb{P}(\mathcal{A}_\varepsilon)}{3} \geq \frac{8}{27}\left[1 - (1-\varepsilon)^{n-1}(1 + (n-1)\varepsilon)\right], \tag{4.2}$$

for all $n > 2$ and with $\mathcal{A}_\varepsilon$ denoting the following event: at least one outlier falls respectively into $[-3; -1[\cup]1; 3]$ and $]-1; 1[$. The inequality 4.2 is valid for all $\varepsilon \geq 0$. When the outlier proportion grows, i.e. when ε increases, the lower bound of the probability that the single linkage algorithm fails increases. Moreover, the lower bound of the probability that the single linkage algorithm fails is governed by the first term, the second term tends quickly to zero when n grows.

To deal with outliers, Arias-Castro (2011) considers a modified version of this procedure that requires the knowledge of the minimal separation distance δ instead of the knowledge of M. He assumes that the minimal distance between the outliers and the clusters is at least δ. It is not the case in our model. In the following section, we propose another version of the single linkage clustering algorithm in the context of outliers.

Robust single linkage clustering

As explained in the last section, the classical single linkage procedure is not adapted to the presence of outliers. To remedy, we propose to consider only the largest clusters.

Recall that, for a fixed radius $r > 0$, the agglomerative clustering presented at the beginning of Section 4.3 provides $M(r)$ clusters

$$\mathcal{Y}(r) = \{\mathcal{Y}_m(r), m \in [\![M(r)]\!]\}.$$

With no loss of generality, indices of the r-connected components in $\mathcal{Y}(r)$ are rearranged such that

$$|\mathcal{Y}_1(r)| \geq |\mathcal{Y}_2(r)| \geq \ldots \geq |\mathcal{Y}_{M(r)}(r)|.$$

Always for fixed $r > 0$, our robust single linkage clustering proposes to consider only the M largest clusters and to merge the other (small) clusters together. Formally, we fix

$$\mathcal{X}_1(r) = \mathcal{Y}_1(r), \ldots, \mathcal{X}_M(r) = \mathcal{Y}_M(r) \quad \text{and} \quad \mathcal{X}_0(r) = \bigcup_{m=M+1}^{M(r)} \mathcal{Y}_m(r). \tag{4.3}$$

Remark 4.3.2.

- *This new algorithm, which is called the robust single linkage clustering, only requires the knowledge of M and returns $M + 1$ clusters.*
- *The cluster $\mathcal{X}_0(r)$ collects the data that are considered as outliers by the procedure whereas the clusters $\mathcal{X}_1(r), \ldots, \mathcal{X}_M(r)$ corresponds, up to a permutation of the indices, to the data that lie near to the sets of interest $S_1, \ldots, S_M$.*
- *The major difference compared to the single linkage clustering is that this algorithm selects the partition which maximizes the size of the M-th cluster and merges the other clusters whose the size is smaller than the one of the M-th cluster into the cluster $\mathcal{X}_0(r)$ which contains the outliers. This step enables to avoid that some isolated observations are treated as singleton clusters. Furthermore, this step uses the assumption that clusters are dense sets whereas the outliers are sparse.*

Under our assumptions, for a safe choice of r, observations in each support $S_i, i = 1, \ldots, M$ should be in the same cluster $\mathcal{X}_j(r)$ (for a given $j \in [\![M]\!]$) and cluster $\mathcal{X}_0(r)$ should contain only observations in S_0. This is proved in the following proposition which controls the clustering risk (4.1) for the partition

$$\mathcal{X}(r) = \{\mathcal{X}_m(r), m \in [\![0, M]\!]\}.$$

Proposition 4.3.1. *Set*

$$\Theta = \kappa^* \kappa_c \kappa_0 \varepsilon$$

and let τ be such that $1/\mathfrak{a} \leq \tau \leq \delta^d n / \log n$. Define $\nu = \mathfrak{a}\tau - 1$. Under the assumptions presented in sections 4.2.2 and 4.2.3, the clustering risk for clusters $\mathcal{X}(r_n)$ with $r_n = (\tau \log n/n)^{1/d}$ satisfies

$$\mathcal{R}_n(\mathcal{X}(r_n)) \leq \frac{\Lambda n^{-\nu}}{\tau \log n} + n\varepsilon(C(d, \gamma_*, \nu)\Theta \log n)^{\lfloor \frac{\delta}{r_n} \rfloor} + 2M \exp(-\psi(\eta)(1 - \varepsilon) \curlywedge n) \tag{4.4}$$

for all η such that $0 < \eta < [(1 - \varepsilon)(1 - \curlywedge)]^{-1}$. Here $\Lambda = \sum_{i=1}^M \Lambda_i = \sum_{i=1}^M (8\Delta_i \sqrt{d})^d$, $\psi(\eta) = (1 + \eta)(\log(1 + \eta) - 1) + 1 > 0$ and $C(d, \gamma_*, \nu) = (1 + \nu)\eta(d)/(\gamma_* \eta_*(d))$.

We emphasize that inequality (4.5) is valid for all n. The upper bound of the clustering risk is governed by the two first terms since the last term tends to zero much faster than the two firsts.

Observe that without outlier, i.e. when $\varepsilon = 0$, the clustering risk tends to 0 at the rate $n^{-\nu}$, as proved in Arias-Castro (2011) and Auray et al. (2015). Parameter ν depends on τ, i.e., on the separation between the clusters: the larger τ, the faster $n^{-\nu}$.

The second term can appear as the price to be paid for the presence of outliers. This term depends on the *sparsity parameter* Θ. This parameter measures the influence of the outliers in the model. Indeed it increases, when ε and κ_0 increases. If we intend to prove any consistency results regarding $\mathcal{R}_n(\mathcal{X}(r_n))$, the sparsity parameter Θ should be such that the second term in the right hand side of (4.5) should tend to 0. For instance, if there exists $\theta_n < 1$ such that

$$\Theta \leq \frac{\theta_n}{C(d, \gamma_*, \nu) \log n}$$

then θ_n should be such that $n\varepsilon\theta_n^{\lfloor \delta/r_n \rfloor}$ tends to zero. If in addition there exists φ_n such that $0 \leq \varphi_n \leq \lfloor \frac{\delta}{r_n} \rfloor - \log n$ and

$$\theta_n \leq \left(1 - \frac{\log n + \varphi_n}{\lfloor \frac{\delta}{r_n} \rfloor}\right)$$

then the rate for the second term in the right hand side of (4.5) is governed by φ_n and

$$\mathcal{R}_n(\mathcal{X}(r_n)) = O\left(\min(n^{-\nu}, \exp(-\varphi_n))\right).$$

Proposition 4.3.1 proves that there exists a fixed radius r_n for which the procedure is efficient. However, this radius depends on τ which is unknown in practice. In this context, we propose a data-driven procedure to select the radius. Recall that the agglomerative procedure provides a sequence of partition $\mathcal{Y}(\rho_0), \ldots, \mathcal{Y}(\rho_{K-1})$. We denote by $\mathcal{X}(\rho_0), \ldots, \mathcal{X}(\rho_{K-1})$ the associated sequence of partitions defined by (4.3). Then, we propose to select the radius in $\{\rho_k \colon k \in [\![0, K-1]\!]\}$ which maximizes the size of the M-th cluster:

$$\widehat{r}_n = \max \operatorname*{argmax}_{\rho \in \{\rho_k \colon k \in [\![0, K-1]\!]\}} |\mathcal{X}_M(\rho)|.$$

The following theorem ensures that this automatic procedure is efficient under our assumptions.

Theorem 4.3.1. *Under the assumptions presented in sections 4.2.2 and 4.2.3, the clustering risk for clusters $\mathcal{X}(\widehat{r}_n)$ satisfies*

$$\mathcal{R}_n(\mathcal{X}(\widehat{r}_n)) \leq \frac{\Lambda n^{-\nu}}{\tau \log n} + n\varepsilon(C(d, \gamma_*, \nu)\Theta \log n)^{\lfloor \frac{\delta}{r_n} \rfloor} + 2M \exp(-\psi(\eta)(1-\varepsilon) \not\asymp n) \tag{4.5}$$

for all η *such that* $0 < \eta < \min(\eta_0, \eta_1)$ *where* $\Lambda, \eta_0, \psi(\eta), \Theta, \nu$ *are defined in Proposition 4.3.1 and* η_1 *is such that*

$$\frac{4\eta_1}{1-\eta_1} = \frac{\gamma_*}{\gamma^*} - \frac{1}{2} > 0.$$

4.4 Simulation study

In this section, the efficiency of the proposed clustering algorithm is highlighted through simulations. An exhaustive comparison with the single linkage algorithm, the k-means algorithm and the spectral clustering is also performed.

For the implementation of the proposed procedure and the single linkage algorithm, we used `R` and the functions `hclust` and `cutree` in the package `stats`. The single linkage, the k-means algorithm and the spectral clustering were implemented respectively with the function `kmeans` in the package `stats` and the function `specc` in the package `kernlab`.

4.4.1 Description of the models

The three models introduced in Section 4.2.4 are considered. Moreover, for each model the two scenarios previously described and respectively called, for the sake of clarity, the *easy case* vs. the *tricky case* are used to highlight the behavior of the clustering approaches through various levels of complexity (see Figures 4.1-4.3).

As we are interested in the performances of the clustering approaches in presence of outliers, in each scenario we make ε to vary between 0 and 0.5 (with a step of 0.02). Furthermore, several values for the sample size n are also considered (200 and 500).

We perform 2000 replications of each experiment. Denote by $\mathcal{X} = \{\mathcal{X}_0, \ldots, \mathcal{X}_M\}$ the partition of the data induced by a clustering procedure. For each replication, the error $err_n(\mathcal{X})$ of the clustering procedure is computed

$$err_n(\mathcal{X}) = \begin{cases} 0 & \text{if } \exists\, \pi \in \Pi_M,\ \forall i = 0, \ldots, M,\ X_{[\![n]\!]} \cap S_i \subseteq \mathcal{X}_{\pi(i)}, \\ 1 & \text{otherwise,} \end{cases}$$

where π is a permutation of the indexes of the clusters and Π_M denotes the set of all permutations of the indexes of the clusters. Then, for each experiment, the clustering risk 4.1 is estimated by the error term averaged over the 2000 replications.

For the spectral clustering, in the function `specc`, we set the number of clusters (parameter `centers`) at M and use the defaults values for all other tuning parameters. For the k-means

approach, in the function `kmeans`, we also set the number of clusters (parameter `centers`) at M, the number of initial configurations (parameter `nstart`) at 10 and consider the default values for the other tuning parameters. For the robust single linkage algorithm and the single linkage algorithm, in the function `hclust`, we set the number of clusters (parameter `centers`) at M and the initial number of clusters at the value $M/2$. Observe that the number M of clusters is required in all these clustering approaches.

4.4.2 Results

Performances of the four clustering methods are assessed and compared based on the clustering risk. Figures 4.4-4.6 display the clustering risk as a function of ε, in each scenario. Tables 4.1 and 4.2 provide the clustering risk for some values of ε in each experiment. Moreover, Figures 4.7 and 4.8 show respectively the results of the algorithms single linkage, k-means, robust single linkage and spectral clustering for data displayed respectively in Figures 4.1b and 4.3b.

First of all, as expected, the clustering risk of the four methods behaves as an increasing function of ε in all experiments. Moreover, for a fixed value of ε, the risk of clustering of all methods seems to increase quickly when the number n of observations decrease.

In all experiments, when there is no outlier (i.e. $\varepsilon = 0$), the clustering risk of both the single linkage algorithm and the robust single linkage is roughly zero. This result agrees with Theorem 4.3.1, Arias-Castro (2011) and Auray et al. (2015) which prove that under assumptions close to **(P1)-(S2)** and when $\varepsilon = 0$, the robust single linkage algorithm and the single linkage linkage algorithm are consistent and their clustering risk tends quickly to zero. Observe that this result does not seem to be verified for both the k-means method and the spectral algorithm.

Furthermore, as proved in Section 4.3, in presence of outliers the single linkage algorithm often fails to recover the true clusters. Moreover, the clustering risk of the single linkage algorithm increases quickly when ε increases (see Figures 4.4-4.3). The robust single algorithm seems less sensitive to the outlier proportion ε and works generally better than the single linkage algorithm, in these experiments.

Overall, compared to the other clustering methods, the robust single linkage clustering seems to quite well perform in all experiments. Indeed, the proposed algorithm is part of the two best methods in each experiment.

Moreover, observe that compared to the k-means algorithm and the spectral clustering approach, our proposed method as well as the single linkage algorithm are exact, in the sense

that they do not involve any random process (for instance the use of random starts in the k-means algorithm and the spectral clustering methods). Furthermore, another major asset of the proposed algorithm is that, contrary to the other assessed clustering methods, this approach also allows identifying some outliers (see Figures 4.7 and 4.8).

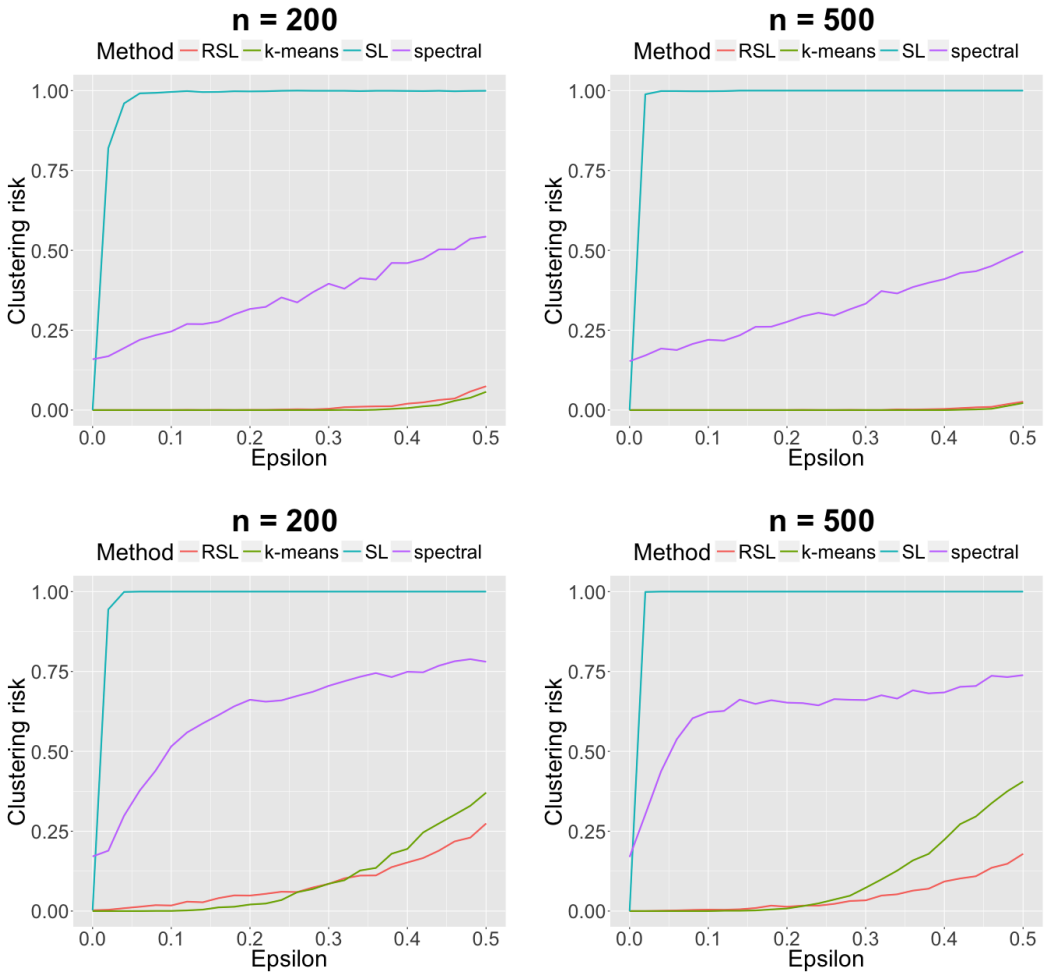


Figure 4.4: Clustering risk according to ε and n in the "squares" model with $\delta = 0.15$ (top) and $\delta = 0.05$ (bottom).

For the "squares" model, the robust single linkage algorithm and the k-means method well perform compared to the single linkage and the spectral clustering whatever the values of the minimal distance δ between the clusters, the number n of observations and the outlier proportion ε.

In experiments based on the "concentric-circles" model, the robust single linkage algorithm and the spectral clustering have higher performances than the k-means method and the single linkage algorithm, for every value of δ, n and ε.

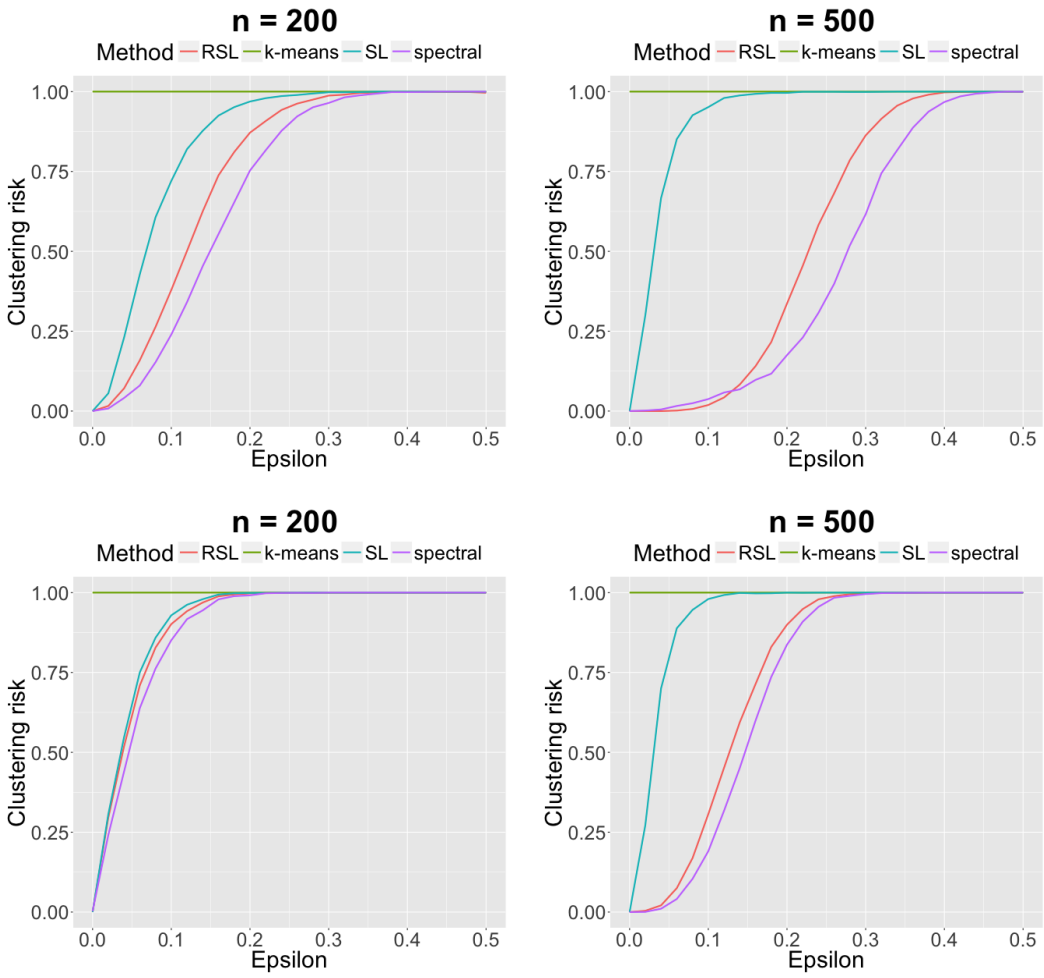


Figure 4.5: Clustering risk according to ε and n in the "concentric-circles" model with $\delta = 2.6$ (top) and $\delta = 1.6$ (bottom).

For the "sine" model, in all experiments, the robust single linkage and the spectral clustering outperform the other clustering approaches. Moreover, for small values of ε (i.e. $\varepsilon \leq 0.05$), the robust single linkage gives better performances than the spectral clustering.

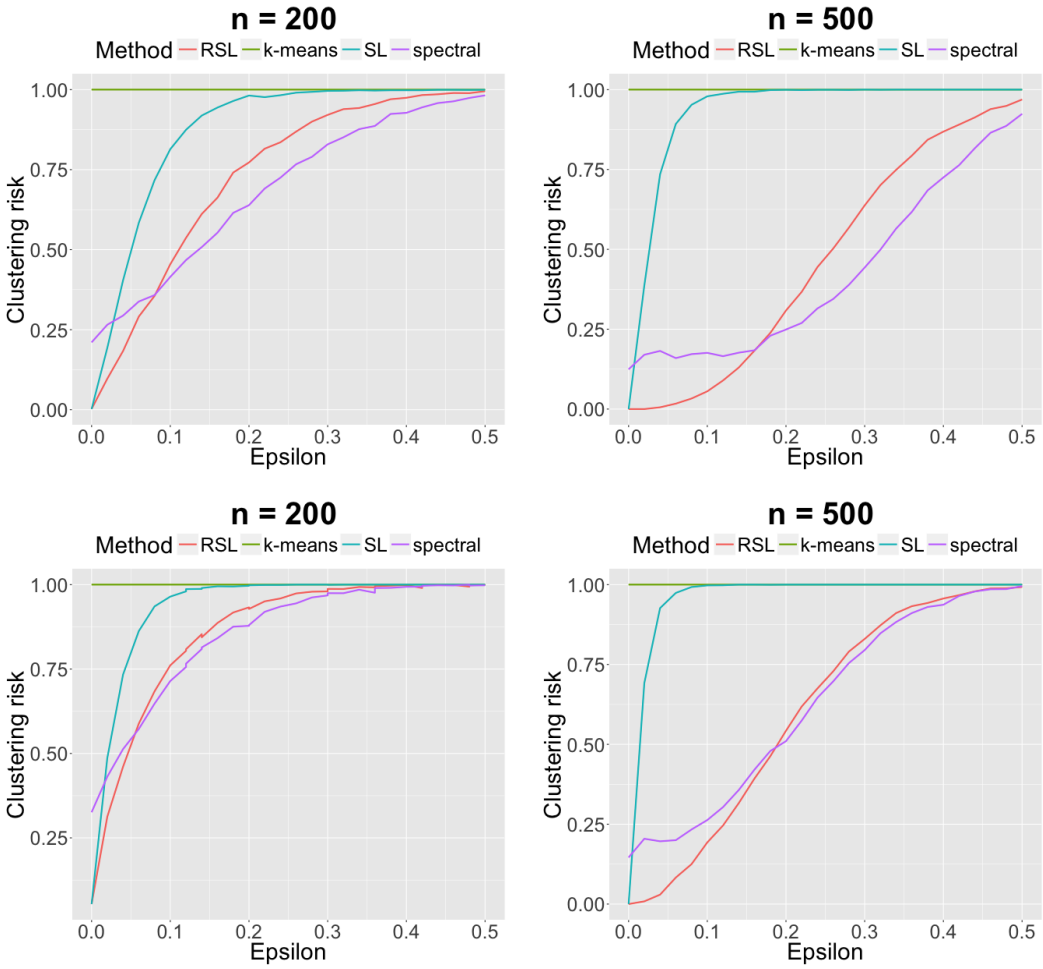


Figure 4.6: Clustering risk according to ε and n in the "sine" model with $\delta = \frac{\sqrt{5\pi}}{6} - 0.5$ (top) and $\delta = \frac{\sqrt{\pi}}{2} - 0.5$ (bottom).

As one might expect, the k-means method works well with compact clusters (see Figures 4.4 and 4.7). Nevertheless, the method often performs badly when clusters are not linearly separable, such as the clusters in the "concentric-circles" model (see Figure 4.5) and in the "sine" model (see Figures 4.5 and 4.8). For experiments in these models, the algorithm works badly whatever the values of n, ε and δ.

Unlike the k-means clustering, the spectral clustering seems more adapted to connected clusters and works quite badly in the "squares" models (see Figures 4.1 and 4.7). This limitation of the algorithm has already been underlined and are well illustrated by for instance Nadler & Galun (2007).

	$\varepsilon = 0$	$\varepsilon = 0.1$	$\varepsilon = 0.2$
"Squares" model, $\delta = 0.25$			
Robust single linkage	0.000 (0.000)	0.000 (0.000)	0.000 (0.000)
Single linkage	0.000 (0.000)	0.996 (0.001)	0.998 (0.001)
K-means	0.000 (0.000)	0.000 (0.000)	0.000 (0.000)
Spectral clustering	0.159 (0.008)	0.246 (0.010)	0.316 (0.010)
"Squares" model, $\delta = 0.15$			
Robust single linkage	0.002 (0.001)	0.018 (0.003)	0.048 (0.005)
Single linkage	0.002 (0.001)	1.000 (0.000)	1.000 (0.000)
K-means	0.000 (0.000)	0.000 (0.000)	0.020 (0.003)
Spectral clustering	0.171 (0.008)	0.516 (0.011)	0.662 (0.011)
"Concentric-circles" model, $\delta = 2.6$			
Robust single linkage	0.000 (0.000)	0.379 (0.011)	0.872 (0.007)
Single linkage	0.000 (0.000)	0.720 (0.010)	0.969 (0.004)
K-means	1.000 (0.000)	1.000 (0.000)	1.000 (0.000)
Spectral clustering	0.000 (0.000)	0.240 (0.010)	0.753 (0.010)
"Concentric-circles" model, $\delta = 1.6$			
Robust single linkage	0.001 (0.000)	0.902 (0.007)	0.998 (0.001)
Single linkage	0.000 (0.000)	0.929 (0.006)	0.999 (0.001)
K-means	1.000 (0.000)	1.000 (0.000)	1.000 (0.000)
Spectral clustering	0.003 (0.001)	0.851 (0.008)	0.992 (0.002)
"Sine" model, $\delta = (\sqrt{5\pi}/6) - 0.5$			
Robust single linkage	0.002 (0.001)	0.454 (0.011)	0.772 (0.009)
Single linkage	0.002 (0.001)	0.814 (0.009)	0.982 (0.003)
K-means	1.000 (0.000)	1.000 (0.000)	1.000 (0.000)
Spectral clustering	0.210 (0.009)	0.415 (0.011)	0.639 (0.011)
"Sine" model, $\delta = (\sqrt{\pi}/2) - 0.5$			
Robust single linkage	0.054 (0.005)	0.761 (0.010)	0.928 (0.006)
Single linkage	0.054 (0.005)	0.964 (0.004)	0.998 (0.001)
K-means	1.000 (0.000)	1.000 (0.000)	1.000 (0.000)
Spectral clustering	0.326 (0.010)	0.714 (0.010)	0.880 (0.007)

Table 4.1: Clustering risk evaluated over 2000 replications for $\varepsilon = 0$ (low) $\varepsilon = 0.1$ (medium) and $\varepsilon = 0.2$ (quite high) and $n = 200$ in each experiment. The standard deviation of the clustering risk is provided in brackets.

	$\varepsilon = 0$	$\varepsilon = 0.1$	$\varepsilon = 0.2$
"Squares" model, $\delta = 0.25$			
Robust single linkage	0.000 (0.000)	0.000 (0.000)	0.000 (0.000)
Single linkage	0.000 (0.000)	0.998 (0.001)	1.000 (0.000)
K-means	0.000 (0.000)	0.000 (0.000)	0.000 (0.000)
Spectral clustering	0.153 (0.008)	0.220 (0.009)	0.276 (0.010)
"Squares" model, $\delta = 0.15$			
Robust single linkage	0.000 (0.000)	0.004 (0.001)	0.014 (0.003)
Single linkage	0.000 (0.000)	1.000 (0.000)	1.000 (0.000)
K-means	0.000 (0.000)	0.000 (0.000)	0.008 (0.002)
Spectral clustering	0.170 (0.008)	0.622 (0.011)	0.652 (0.011)
"Concentric-circles" model, $\delta = 2.6$			
Robust single linkage	0.000 (0.000)	0.019 (0.003)	0.336 (0.011)
Single linkage	0.000 (0.000)	0.952 (0.005)	0.996 (0.001)
K-means	1.000 (0.000)	1.000 (0.000)	1.000 (0.000)
Spectral clustering	0.000 (0.000)	0.038 (0.004)	0.175 (0.008)
"Concentric-circles" model, $\delta = 1.6$			
Robust single linkage	0.000 (0.000)	0.306 (0.010)	0.900 (0.007)
Single linkage	0.000 (0.000)	0.980 (0.003)	1.000 (0.000)
K-means	1.000 (0.000)	1.000 (0.000)	1.000 (0.000)
Spectral clustering	0.000 (0.000)	0.190 (0.009)	0.837 (0.008)
"Sine" model, $\delta = (\sqrt{5\pi}/6) - 0.5$			
Robust single linkage	0.000 (0.000)	0.056 (0.005)	0.308 (0.010)
Single linkage	0.000 (0.000)	0.979 (0.003)	1.000 (0.000)
K-means	1.000 (0.000)	1.000 (0.000)	1.000 (0.000)
Spectral clustering	0.124 (0.007)	0.176 (0.009)	0.249 (0.010)
"Sine" model, $\delta = (\sqrt{\pi}/2) - 0.5$			
Robust single linkage	0.000 (0.000)	0.193 (0.009)	0.542 (0.011)
Single linkage	0.000 (0.000)	0.998 (0.001)	1.000 (0.000)
K-means	1.000 (0.000)	1.000 (0.000)	1.000 (0.000)
Spectral clustering	0.146 (0.008)	0.263 (0.010)	0.510 (0.011)

Table 4.2: Clustering risk evaluated over 2000 replications for $\varepsilon = 0$ (low) $\varepsilon = 0.1$ (medium) and $\varepsilon = 0.2$ (quite high) and $n = 500$ in each experiment. The standard deviation of the clustering risk is provided in brackets.

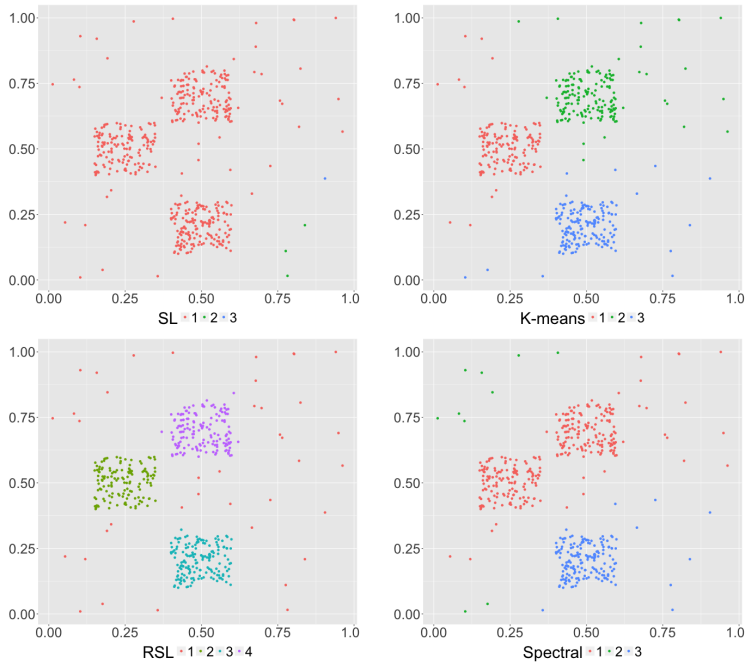


Figure 4.7: Results of the algorithms single linkage (SL), k-means, robust single linkage (RSL) and spectral clustering for the data displayed in Figure 4.1b.

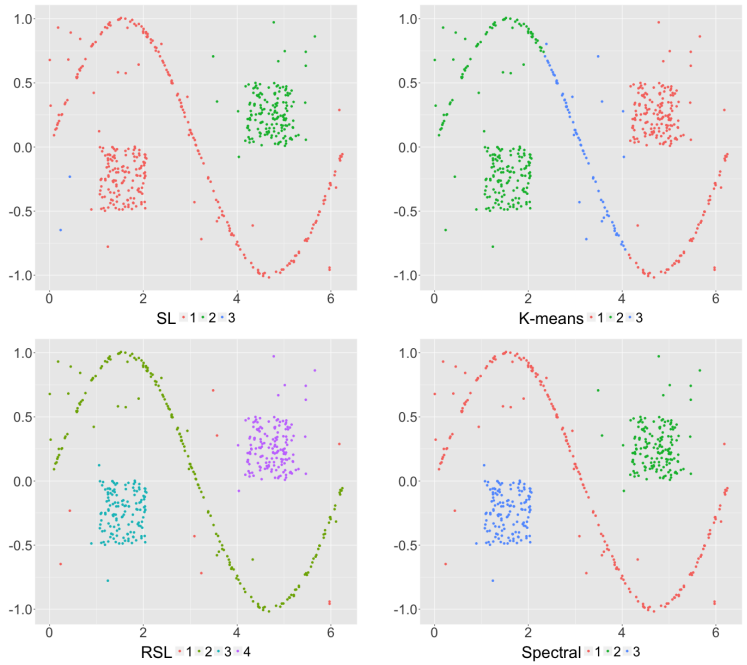


Figure 4.8: Results of the algorithms single linkage (SL), k-means, robust single linkage (RSL) and spectral clustering for the data displayed in Figure 4.3b.

4.5 Proofs

4.5.1 Technical lemmas

We assume that assumptions introduced in sections 4.2.2 and 4.2.3 are satisfied.

Lemma 4.5.1. *Fix $r > 0$ and $i = 1, \dots, M$. Denote $\Lambda_i = (8\Delta_i\sqrt{d})^d$. Then S_i can be recovered by at most $\Lambda_i r^{-d}$ balls centered at a point in S_i with radius $r/4$.*

Proof. Set $r > 0$, $i \in \{1, \dots, M\}$ ans $s_0 \in S_i$. Let $B_\infty(s_0, \Delta_i) = \{x \in \mathbb{R}^d : \|x - s_0\|_\infty \leq \Delta_i\}$. Let $a_j, j = 1, \dots, d$ and $b_j, j = 1, \dots, d$ with $(b_j - a_j) \leq \Delta_j$ such that

$$B_\infty(s_0, \Delta_i) = \prod_{j=1}^{d} [a_j, b_j].$$

Fix

$$\mathcal{K} = \left\{ (k_1, \dots, k_d) \in \mathbb{N}^d \colon 1 \leq k_j \leq \left\lfloor \frac{8\Delta_i\sqrt{d}}{r} \right\rfloor \right\}$$

and for $k \in \mathcal{K}$

$$I(k) = \prod_{j=1}^{d} \left[\left(a_j + \frac{k_j r}{8\sqrt{d}} \right) - \frac{r}{8\sqrt{d}}, \left(a_j + \frac{k_j r}{8\sqrt{d}} \right) + \frac{r}{8\sqrt{d}} \right].$$

It is easily seen that $B_\infty(s_0, \Delta_i) \subset \bigcup_{k \in \mathcal{K}} I(k)$ and $S_i \subset \bigcup_{k \in \mathcal{K}_0} I(k)$ where $\mathcal{K}_0 = \{k \in \mathcal{K} \colon I(k) \cap S_i \neq \emptyset\}$. Now $\forall k \in \mathcal{K}_0$, let $y(k) \in I(k) \cap S_i$ and remark that, using triangle inequality, $I(k) \subset B_\infty((y_k), r/(4\sqrt{d}))$. We have

$$S_i \subset \bigcup_{k \in \mathcal{K}_0} B_\infty \left(y(k), \frac{r}{4\sqrt{d}} \right) \subset \bigcup_{k \in \mathcal{K}_0} B \left(y(k), \frac{r}{4} \right).$$

Since $|\mathcal{K}_0| \leq \Lambda_i r^{-d}$, result follows.

Lemma 4.5.2. *Fix $r > 0$ and $i = 1, \dots, M$. Under the assumptions presented in sections 4.2.2 and 4.2.3, we have*

$$\psi_{n,i}(r) = \mathbb{P}^n(X_{[\![n]\!]} \cap S_i \text{ is not } r\text{-connected}) \leq \Lambda_i r^{-d} \exp(-\mathfrak{a} n r^d).$$

Proof.
From Lemma 4.5.1, there exists $(B_\ell)_{\ell \in \mathcal{L}_i}$ a coverage of S_i by $\Lambda_i = |\mathcal{L}_i| = (8\Delta_i\sqrt{d})^d$ balls centered at a point that belongs to S_i and with radius $r/4 > 0$. Observe that if, for any

$\ell \in \mathcal{L}_i$ $(X_{[\![n]\!]} \cap S_i) \cap B_\ell \neq \emptyset$, then there exists $\alpha \in [\![n]\!]$ such that $X_\alpha \in B_\ell$ and thus $B(X_\alpha, r/2) \supset B_\ell$ and $X_{[\![n]\!]} \cap S_i$ is r-connected. Therefore,

$$\begin{aligned}
\psi_{n,i}(r) &\leq \mathbb{P}^n\left(\exists \ell \in \mathcal{L}_i, B_\ell \cap (X_{[\![n]\!]} \cap S_i) = \emptyset\right) \\
&\leq \mathbb{P}^n\left(\exists \ell \in \mathcal{L}_i,\ \forall X_k \in X_{[\![n]\!]}, \quad X_k \notin S_i \text{ or } (X_k \in S_i, X_k \notin B_\ell)\right) \\
&\leq \sum_{\ell \in \mathcal{L}_i} \left(\mathbb{P}(X \notin S_i) + \mathbb{P}(X \notin B_\ell \mid X \in S_i)\mathbb{P}(X \in S_i)\right)^n \\
&\leq \sum_{\ell \in \mathcal{L}_i} \left(1 - \mathbb{P}(X \in B_\ell \mid X \in S_i)\mathbb{P}(X \in S_i)\right)^n \\
&\leq \sum_{\ell \in \mathcal{L}_i} \left(1 - (1-\varepsilon)\gamma_i \mathbb{P}_i(X \in B_\ell)\right)^n
\end{aligned}$$

Now, using **(P1)**, **(P3)** and **(G2)** we obtain:

$$\begin{aligned}
\mathbb{P}_i(X \in B_\ell) &= \mathbb{P}_i(X \in B_\ell \cap S_i) && \text{from } \textbf{(P1)} \\
&\geq \kappa_i^{-1} \mathcal{H}^{s_i}(B_\ell \cap S_i) && \text{from } \textbf{(P3)} \\
&\geq (\kappa_i \kappa_c)^{-1} \eta(s_i) r^{s_i} && \text{from } \textbf{(G2)} \\
&\geq (\kappa^* \kappa_c)^{-1} \eta_*(d) r^d,
\end{aligned}$$

where we recall that $\kappa^* = \max\{\kappa_i : i \in [\![M]\!]\}$ and $\eta_*(d) = \min\{\eta(s) : 0 \leq s \leq d\} = \min(\eta(d), 1)$ as defined in Section 4.2.3.
This implies that

$$\begin{aligned}
\psi_{n,i}(r) &\leq |\mathcal{L}_i|\left(1 - (1-\varepsilon)\gamma_*(\kappa^*\kappa_c)^{-1}\eta_*(d) r^d\right)^n \\
&\leq \Lambda_i r^{-d} \exp(-\mathfrak{a} n r^d),
\end{aligned}$$

with $\mathfrak{a} = (1-\varepsilon)\gamma_*(\kappa^*\kappa_c)^{-1}\eta_*(d)$.

Lemma 4.5.3. *Let $0 < r < 1$ and denote by $\varphi_n(m, r)$ the probability that there exists, in S_0, a r-path of at least m r-connected observations. We have*

$$\varphi_n(m, r) \leq n\varepsilon(\mathfrak{b} n r^d)^{m-1}$$

where $\mathfrak{b} = \eta(d)\kappa_0\varepsilon$.

Proof. Let $0 < r < 1$ and denote by $\varphi_n(m, r)$ the probability that there exists, in S_0, a r-path of at least m connected observations. For any $I \subseteq [\![n]\!]$ we denote by $\mathcal{A}_I$ the following event: there exists a permutation $i_1 < \ldots < i_m$ of I such that $\|X_{i_j} - X_{i_{j+1}}\| \leq r$ for any $j = 1, \ldots, m-1$. We have:

$$\begin{aligned}
\varphi_n(m, r) &\leq \sum_{\substack{I \subseteq [\![n]\!] \\ |I| = m}} \mathbb{P}^n(\mathcal{A}_I \cap \{X_I \subseteq S_0\}) \\
&\leq \sum_{\substack{I \subseteq [\![n]\!] \\ |I| = m}} \varepsilon^m \mathbb{P}_0^m(\mathcal{A}_I).
\end{aligned}$$

Now remark that

$$\begin{aligned}\mathbb{P}_0^m(\mathcal{A}_I) &\leq m!\mathbb{E}_0^m\left(\prod_{j=1}^{m-1}\mathbf{1}_{[0,r]}(\|X_{i_j}-X_{i_{j+1}}\|)\right)\\ &= m!\int_{S_0}\cdots\int_{S_0}\mathbf{1}_{[0,r]}(\|x_1-x_2\|)\ldots\mathbf{1}_{[0,r]}(\|x_{m-1}-x_m\|)\mathrm{d}\mathbb{P}_0^m(x_1,\ldots,x_m)\end{aligned}$$

Note also that, using **(P2)**:

$$\int_{S_0}\mathbf{1}_{[0,r]}(\|x-y\|)\mathrm{d}\mathbb{P}_0(y)\leq\mathbb{P}_0(B(x,r))\leq\kappa_0\mathcal{H}^d(B(x,r))=\kappa_0\eta(d)r^d.$$

This, combined with Fubini's theorem implies that:

$$\mathbb{P}_0^m(\mathcal{A}_I)\leq m!(\eta(d)\kappa_0 r^d)^{m-1}.$$

Finally, we obtain:

$$\begin{aligned}\varphi_n(m,r)&\leq\frac{n!}{(n-m)!}\varepsilon^m(\eta(d)\kappa_0 r^d)^{m-1}\\ &\leq n\varepsilon(\mathfrak{b}nr^d)^{m-1}.\end{aligned}$$

Lemma 4.5.4. *Let*

$$\Omega_\eta=\bigcap_{i=1}^{M}\{(1-\eta)(1-\varepsilon)\gamma_i n<N_i<(1+\eta)(1-\varepsilon)\gamma_i n\}$$

where $\eta_0=1-[(1-\varepsilon)(1-\not\gamma)]^{-1}>0$, $0<\eta\leq\eta_0$ *and* $N_i=|X_{[\![n]\!]}\cap S_i|$. *Then*

(i) we have

$$\mathbb{P}(\overline{\Omega_\eta})\leq 2M\exp(-\psi(\eta)(1-\varepsilon)\not\gamma n)$$

where $\psi(\eta)=(1+\eta)(\log(1+\eta)-1)+1>0$.

(ii) Under Ω_η,

$$N_0<\frac{\not\gamma}{\gamma_*}\min_{i=1,\ldots,M}N_i,$$

with $N_0=\leq n-\sum_{i=1}^{M}N_i$.

Proof. Since $N_i\sim B(n,(1-\varepsilon)\gamma_i)$, (i) is a direct consequence of (Shorack & Wellner, 1986, page 440). For (ii), observe that $(1-\varepsilon)(1-\eta_0)=1/(1+\not\gamma)$. Since $0<\eta\leq\eta_0$, it follows that

$$1-(1-\varepsilon)(1-\eta)\leq(1-\varepsilon)(1-\eta)\not\gamma.$$

Thus, under Ω_η,

$$\begin{aligned} N_0 &\le n - \sum_{i=1}^{M} N_i \le n\left(1 - (1-\varepsilon)(1-\eta)\sum_{i=1}^{M}\gamma_i\right) \\ &\le n(1-(1-\varepsilon)(1-\eta)) \le n(1-\varepsilon)(1-\eta)\not\gamma \\ &\le \frac{\not\gamma}{\gamma_*} n(1-\varepsilon)(1-\eta)\gamma_i < \frac{\not\gamma}{\gamma_*} N_i \quad \forall i = 1,\dots,M. \end{aligned}$$

Lemma 4.5.5. *Let $\eta_0 = 1 - [(1-\varepsilon)(1-\not\gamma)]^{-1} > 0$ and η_1 be such that*

$$\frac{4\eta_1}{1-\eta_1} = \frac{\gamma_*}{\gamma^*} - \frac{1}{2} > 0.$$

For each $\eta \le \min(\eta_0, \eta_1)$ we have

$$\frac{1+\eta}{1-\eta}\frac{\gamma^*}{\gamma_*} + \frac{\not\gamma}{\gamma_*} \le 2.$$

Proof. Let $\eta \le \min(\eta_0, \eta_1)$, then

$$\begin{aligned} \frac{1+\eta}{1-\eta}\frac{\gamma^*}{\gamma_*} + \frac{\not\gamma}{\gamma_*} &= \frac{1-\eta+2\eta}{1-\eta}\frac{\gamma^*}{\gamma_*} + \frac{\not\gamma}{\gamma_*} \\ &= \frac{\gamma * + \not\gamma}{\gamma_*} + \frac{1}{2}\frac{4\eta}{1-\eta}\frac{\gamma^*}{\gamma_*} \\ &\le \frac{\gamma^*/2 + \gamma_*}{\gamma_*} + \frac{1}{2}\left(\frac{\gamma_*}{\gamma^*} - \frac{1}{2}\right)\frac{\gamma^*}{\gamma_*} \\ &= \frac{3}{2} + \frac{\gamma^*}{4\gamma_*} = 2. \end{aligned}$$

Lemma 4.5.6. *For $r > 0$, let*

$$\mathcal{E}(r) = \left\{\exists \pi \in \Pi_M \, \forall i = 1,\dots,M \; X_{[\![n]\!]} \cap S_i \subseteq \mathcal{X}_{\pi(i)}(r)\right\}$$

Let $\eta \le \min(\eta_0, \eta_1)$, then under $\mathcal{E}(r_n) \cap \Omega_\eta$ we have

1. *$\widehat{r}_n \ge r_n$ almost surely;*
2. *There exists $\pi \in \Pi_M$ such that, $\forall i = 1,\dots,M$ $\mathcal{X}_i(r_n) \subseteq \mathcal{X}_{\pi(i)}(\widehat{r}_n)$.*

Proof. Let $\eta \leq \min(\eta_0, \eta_1)$ and assume that Ω_η is true. We first prove that $\widehat{r}_n \geq r_n$ with a reductio ad absurdum. Assume that $\widehat{r}_n < r_n$. Observe that

$$|\mathcal{Y}_M(\widehat{r}_n)| > |\mathcal{Y}_M(r_n)|$$

since $\widehat{r}_n$ is the largest r which maximizes $|\mathcal{Y}_M(r)|$. It follows that

$$|\mathcal{Y}_1(\widehat{r}_n)| \geq \ldots \geq |\mathcal{Y}_M(\widehat{r}_n)| > |\mathcal{Y}_M(r_n)|\,.$$

Since $\widehat{r}_n < r_n$, we deduce that one of the $\mathcal{Y}_i(r_n), i = 1, \ldots, M-1$ contains observations of at least two clusters among $\mathcal{Y}_i(\widehat{r}_n), i = 1, \ldots, M$. It implies that

$$|\mathcal{Y}_{M-1}(r_n)| > 2\,|\mathcal{Y}_M(r_n)|\,. \tag{4.6}$$

Moreover, under $\mathcal{E}(r_n)$ we have $N_{(i)} \leq |\mathcal{Y}_i(r_n)| \leq N_{(i)} + N_0$ where $N_i = |X_{[\![n]\!]} \cap S_i|$ and $N_{(i)}, i = 1, \ldots, M$ are such that

$$N_{(1)} \leq \ldots \leq N_{(M)}.$$

Thus, under $\mathcal{E}(r_n) \cap \Omega_\eta$, we have from Lemmas 4.5.4 and 4.5.6

$$\left|\mathcal{Y}_{(M-1)}(r_n)\right| \leq N_{(M-1)} + N_0 \leq N_{(M-1)} + \frac{\gamma}{\gamma_*} N_{(M)}$$

and

$$\frac{\left|\mathcal{Y}_{(M-1)}(r_n)\right|}{\left|\mathcal{Y}_{(M)}(r_n)\right|} \leq \frac{N_{(M-1)}}{N_{(M)}} + \frac{\gamma}{\gamma_*} \leq \frac{1+\eta}{1-\eta}\frac{\gamma^*}{\gamma_*} + \frac{\gamma}{\gamma_*} \leq 2$$

which is a contradiction with (4.6). We deduce that $\widehat{r}_n \geq r_n$ almost surely.

For the second point, observe that since $\widehat{r}_n \geq r_n$, each $\mathcal{X}_i(\widehat{r}_n), i = 1, \ldots, M$ may be written as the union of clusters in

$$\mathcal{X}_1(r_n), \ldots, \mathcal{X}_M(r_n), \mathcal{Y}_{M+1}(r_n), \ldots, \mathcal{Y}_{M(r_n)}(r_n).$$

Moreover for each $i = 1, \ldots, M$ there exists an unique $j = 1, \ldots, M$ and a subset $\mathcal{T}(\widehat{r}_n)$ of $\{M+1, \ldots, M(r_n)\}$ such that

$$\mathcal{X}_i(\widehat{r}_n) = \mathcal{X}_j(r_n) + \bigcup_{\ell \in \mathcal{T}(\widehat{r}_n)} \mathcal{Y}_\ell(r_n). \tag{4.7}$$

Indeed if there exists $i = 1, \ldots, M$ and $1 \leq j \neq j' \leq M$ such that

$$\mathcal{X}_j(r_n) \cup \mathcal{X}_{j'}(r_n) \subseteq \mathcal{X}_i(\widehat{r}_n)$$

then $\mathcal{X}_M(\widehat{r}_n)$ is the union of clusters in

$$\{\mathcal{Y}_{M+1}(r_n), \ldots, \mathcal{Y}_{M(r_n)}(r_n)\},$$

and thus $|\mathcal{X}_M(\widehat{r}_n)| \leq N_0$. This is not possible since, by definition of $\widehat{r}_n$ and by Lemma 4.5.4 we have

$$|\mathcal{X}_M(\widehat{r}_n)| \geq |\mathcal{X}_M(r_n)| \geq N_{(M)} > \frac{\gamma_*}{\gamma} N_0 \geq N_0.$$

We deduce from (4.7) that there exists $\pi \in \Pi_M$ such that, $\forall i = 1, \ldots, M$ $\mathcal{X}_i(r_n) \subseteq \mathcal{X}_{\pi(i)}(\widehat{r}_n)$.

4.5.2 Proof of proposition 4.3.1

First observe that for $r > 0$,

$$\begin{aligned} 1 - \mathcal{R}_n(\mathcal{X}(r)) &= \mathbb{P}\left(\exists \pi \in \Pi_M \, \forall i = 1, \dots, M \; X_{[\![n]\!]} \cap S_i \subseteq \mathcal{X}_{\pi(i)}(r)\right) \\ &\geq \mathbb{P}\left(\exists \pi \in \Pi_M \, \forall i = 1, \dots, M \; X_{[\![n]\!]} \cap S_i \subseteq \mathcal{X}_{\pi(i)}(r), \Omega_\eta\right) \end{aligned} \tag{4.8}$$

where Ω_η is the event defined in Lemma 4.5.4. The event in (4.8) is similar to the intersection of

$$\begin{cases} \forall i = 1, \dots, M, \; X_{[\![n]\!]} \cap S_i \text{ are } r\text{-connected} \\ \forall i \neq j, \text{ there is no } r\text{-connected path between} X_{[\![n]\!]} \cap S_i \text{ and } X_{[\![n]\!]} \cap S_j \\ \Omega_\eta \end{cases}$$

which contains

$$\begin{cases} \forall i = 1, \dots, M, \; X_{[\![n]\!]} \cap S_i \text{ are } r\text{-connected} \\ \text{there is no } r\text{-connected path in } S_0 \text{ with at least } \lfloor \delta/r \rfloor + 1 \text{ observations} \\ \Omega_\eta. \end{cases}$$

We deduce that

$$\mathcal{R}_n(\mathcal{X}(r)) \leq \sum_{i=1}^{M} \psi_{n,i}(r) + \varphi_n\left(\left\lfloor \frac{\delta}{r} \right\rfloor + 1, r\right) + \mathbb{P}(\overline{\Omega_\eta}) \tag{4.9}$$

where $\psi_{n,i}$ and φ_n are defined in Lemmas 4.5.2 and 4.5.3. For

$$r = r_n = \left(\tau \frac{\log n}{n}\right)^d,$$

we obtain the first term

$$\sum_{i=1}^{M} \psi_{n,i}(r_n) = \Lambda r^{-d} \exp(-\mathfrak{a}\tau \log(n)) = \Lambda \frac{n^{-\nu}}{\tau \log n},$$

with $\Lambda = \sum_{i=1}^{M} \Lambda_i = \sum_{i=1}^{M} (8\Delta_i \sqrt{d})^d$.
For the second term, we have from Lemma 4.5.3

$$\varphi_n\left(\left\lfloor \frac{\delta}{r} \right\rfloor + 1, r\right) = n\varepsilon(\mathfrak{b}\tau \log n)^{\lfloor \frac{\delta}{r} \rfloor}.$$

Since

$$\mathfrak{b}\tau = (1+\nu)\frac{\mathfrak{b}}{\mathfrak{a}} = (1+\nu)\eta(d)\frac{\kappa_0 \varepsilon}{\mathfrak{a}} \leq \frac{(1+\nu)\eta(d)}{\gamma_* \eta_*(d)} \, \kappa_0 \kappa_c \kappa^* \varepsilon = C(d, \gamma_*, \nu)\Theta,$$

where

$$C(d, \gamma_*, \nu) = \frac{(1+\nu)\eta(d)}{\gamma_* \eta_*(d)} \quad \text{and} \quad \Theta = \kappa^* \kappa_c \kappa_0 \varepsilon.$$

The last term of (4.9) is bounded in Lemma 4.5.4.

4.5.3 Proof of Theorem 4.3.1

For $r > 0$, let

$$\mathcal{E}(r) = \left\{\exists \pi \in \Pi_M \, \forall i = 1, \dots, M \; X_{[\![n]\!]} \cap S_i \subseteq \mathcal{X}_{\pi(i)}(r)\right\} \tag{4.10}$$

Let $\eta \leq \min(\eta_0, \eta_1)$. Observe that

$$\begin{aligned}
1 - \mathcal{R}_n(\mathcal{X}(\widehat{r}_n)) &= \mathbb{P}\left(\exists \pi \in \Pi_M \, \forall i = 1, \dots, M \; X_{[\![n]\!]} \cap S_i \subseteq \mathcal{X}_{\pi(i)}(\widehat{r}_n)\right) \\
&\geq \mathbb{P}\left(\exists \pi \in \Pi_M \, \forall i = 1, \dots, M \; X_{[\![n]\!]} \cap S_i \subseteq \mathcal{X}_{\pi(i)}(\widehat{r}_n), \mathcal{E}(r_n), \Omega_\eta\right) \\
&\geq \mathbb{P}\left(\exists \pi \in \Pi_M \, \forall i = 1, \dots, M \; X_{[\![n]\!]} \cap S_i \subseteq \mathcal{X}_{\pi(i)}(r_n), \mathcal{E}(r_n), \Omega_\eta\right)
\end{aligned}$$

since from Lemma 4.5.6 there exists $\pi \in \Pi_M$ such that, $\forall i = 1, \dots, M \; \mathcal{X}_i(r_n) \subseteq \mathcal{X}_{\pi(i)}(\widehat{r}_n)$ under $\mathcal{E}(r_n) \cap \Omega_\eta$. Since

$$\mathcal{E}(r_n) = \left\{\exists \pi \in \Pi_M \, \forall i = 1, \dots, M \; X_{[\![n]\!]} \cap S_i \subseteq \mathcal{X}_{\pi(i)}(r)\right\}$$

we deduce that

$$\mathcal{R}_n(\mathcal{X}(\widehat{r}_n)) \leq 1 - \mathbb{P}(\Omega_\eta, \mathcal{E}(r_n))$$

and the result follows from Proposition 4.3.1.

4.6 A brief review of the Hausdorff measure

Here, $d \in \mathbb{N}^*$ stands for the dimension of the ambiant euclidean space $\mathbb{R}^d$ endowed with the euclidean norm $\|\cdot\|$. We also consider the diameter of A denoted by $\Delta(A) = \max\{\|x - y\| : x \in A, y \in A\}$. For any $r > 0$ we define the set $B(A, r) = \{y \in \mathbb{R}^d : \|x - y\| \leq r \text{ for any } x \in A\}$. If $A = \{x\}$ this set correspond to the ball $B(x, r)$ centered at point x with radius r.
In what follows, $\mathbb{X}$ is an open subset of $\mathbb{R}^d$ and $2^{\mathbb{X}}$ denotes the collection of all subsets of $\mathbb{X}$. For any $0 \leq s \leq d$ and $\delta > 0$, the δ-approximate s-dimensional Hausdorff outer measure is defined, for any $A \in 2^{\mathbb{X}}$, by

$$\mathcal{H}_\delta^s(A) = \inf\left\{\eta(s) \sum_{j=1}^{+\infty} \left(\frac{\Delta(C_j)}{2}\right)^s : A \subseteq \bigcup_{j=1}^{+\infty} C_j \text{ and } \Delta(C_j) \leq \delta\right\},$$

where $\eta(s) = \pi^{s/2}/\Gamma(1 + s/2)$ is a normalizing constant and Γ is the usual gamma function. The s-dimension Hausdorff outer measure $\mathcal{H}^s$ is then defined, for any $A \in 2^{\mathbb{X}}$, by:

$$\mathcal{H}^s(A) = \sup_{\delta > 0} \mathcal{H}_\delta^s(A).$$

We recall that Hausdorff outer measures allow to measure "small" subsets of $\mathbb{R}^d$. For the convience of the reader we state some classical results that will be usefull in our context (see Evans & Gariepy, 2015, for more details).

Lemma 4.6.1. *Let $0 \leq s \leq d$ and $A \in 2^{\mathbb{X}}$. The following properties hold:*

- *If s is an integer, the Hausdorff measure $\mathcal{H}^s$ agrees with ordinary s-dimensional surface area on regular sets.*
- *$\mathcal{H}^s$ is invariant under the action of any affine isometry. Moreover, for any $\lambda > 0$ we have: $\mathcal{H}^s(\lambda A) = \lambda^s \mathcal{H}^s(A)$.*
- *If $\mathcal{H}^s(A) < +\infty$ then for any $t > s$ we have $\mathcal{H}^t(A) = 0$. If $\mathcal{H}^s(A) > 0$ then for any $t < s$ we have $\mathcal{H}^t(A) = +\infty$. The Hausdorff dimension of A is defined as: $\dim_H(A) = \inf\{s \geq 0 : \mathcal{H}^s(A) = 0\}$.*
- *Assume that $0 < s < d$ and assume also that A is a $\mathcal{H}^s$-measurable set such that $0 < \mathcal{H}^s(A) < +\infty$. For $\mathcal{H}^s$-a.e. $x \in A$:*

$$2^{-s} \leq \limsup_{r \to 0} \frac{\mathcal{H}^s(A \cap B(x,r))}{\eta(s) r^s} \leq 1.$$

Chapitre 5

Conclusion

Table des Matières

5.1 Bilan 145
5.2 Perspectives 146

5.1 Bilan

L'objectif de ce travail de thèse était de développer des méthodes permettant de construire des arbres de décision et des forêts aléatoires à partir de données ayant une structure de groupe, en apprentissage supervisé. Dans ce contexte, deux approches par arbres ont tout d'abord été proposées.

La première méthode, que nous avons appelée TPLDA (pour Tree Penalized Linear Discriminant Analysis), permet de construire des arbres binaires en classification. La méthode bâtit un arbre maximal au moyen d'un partitionnement récursif et binaire de l'espace des données. La division d'un nœud est définie à partir d'un groupe de variables et d'une combinaison des variables du dit groupe, qui est estimée par une analyse discriminante régularisée. Une stratégie d'élagage se basant sur la profondeur de l'arbre maximal a été proposée pour sélectionner l'arbre final.

La seconde méthode, appelée CARTGV (pour Classification And Regression Trees for Grouped Variables), peut être vue comme une extension de la méthode CART (Breiman et al., 1984) aux groupes de variables. Contrairement à la méthode TPLDA, cette nouvelle approche ne fait aucune hypothèse sur la forme de la relation entre les variables d'un même groupe. Utilisable à la fois en régression et en classification, cette méthode originale construit d'abord un arbre non-binaire, maximal, dans lequel chaque coupure est un arbre binaire. L'arbre ainsi obtenu est ensuite élagué selon une généralisation de la méthode *minimal cost-complexity*

pruning aux arbres non binaires.

Les arbres de décision étant connus pour être instables, nous avons également développé un algorithme de forêts aléatoires pour variables groupées. Cette méthode, que nous avons appelée RFGV (pour Random Forests for Grouped Variables), consiste en une agrégation d'arbres aléatoires construits selon une variante de la méthode CARTGV. Contrairement aux forêts de Breiman (2001), cette nouvelle méthode perturbe l'espace des variables à deux niveaux : au niveau des groupes de variables mais aussi au niveau des variables individuelles.

Outre la construction de règles de prédiction, ces trois nouvelles approches peuvent aussi être utilisées pour faire de la sélection de groupes de variables grâce à l'introduction pour chacune d'elles de scores d'importance. Pour TPLDA et CARTGV, un indice d'importance reposant sur une réduction "corrigée" de l'impureté a été défini. Pour CARTGV, un second score a aussi été proposé. Il utilise les coupures par substitution, notion introduite par Breiman et al. (1984) et que nous avons adaptée aux groupes de variables dans le cadre des arbres CARTGV. Enfin, le score d'importance pour les groupes de variables introduit par Gregorutti et al. (2015) a naturellement été adapté aux forêts RFGV.

Évaluées et comparées à d'autres méthodes de référence (comme le Group lasso, CART ou encore les forêts aléatoires de Breiman) lors d'études de simulations, ces nouvelles approches ont montré de bonnes performances et semblent particulièrement adaptées pour construire des règles de prédiction à partir de données groupées.

Enfin, dans une dernière partie, nous nous sommes placés en apprentissage non-supervisé. Nous avons proposé une nouvelle méthode de clustering hiérarchique. Cette approche, que nous avons appelée *robust single linkage*, peut être vue comme une version "robuste" de l'algorithme *single linkage* en présence d'outliers. Dans un cadre mathématique proche de ceux proposés par Arias-Castro (2011) et Maier et al. (2009) et sous certaines hypothèses portant notamment sur la séparabilité et la régularité des clusters, nous avons obtenu des vitesses de convergence pour le *risque de clustering* de l'algorithme proposé. Les bonnes performances de cette nouvelle approche ont aussi été démontrées par des études de simulations dans lesquelles la méthode a été comparée à d'autres algorithmes de clustering classiques (k-means, clustering spectral, *single linkage*).

5.2 Perspectives

Dans cette section, nous exposons quelques perspectives et prolongements naturels du travail de thèse présenté dans ce manuscrit.

Tout d'abord, dans la méthode TPLDA, pour découper un nœud la méthode réalise une analyse discriminante régularisée sur chaque groupe de variables. Cette approche régularisée

utilise un paramètre de pénalité qui est sélectionné en pratique par validation croisée. Bien que la complexité (algorithmique) de la méthode TPLDA s'avère beaucoup faible que la plupart des méthodes de partitionnement utilisant des coupures multivariées (voir Chapitre 2), le recours à la validation croisée pour chaque nœud et chaque groupe de variables peut rapidement devenir coûteux en temps de calcul, notamment quand le nombre de groupes est très grand. Ainsi, il pourrait être intéressant d'envisager d'autres méthodes pour la sélection du paramètre de pénalité.

D'autre part, dans le cadre des arbres CARTGV, nous avons proposé une définition du concept de coupures par substitution pour les groupes de variables (voir Chapitre 3). Cette définition peut aussi être naturellement étendue aux arbres TPLDA. En effet, l'idée serait d'estimer les coupures par substitution en utilisant des analyses discriminantes pénalisées et non plus des arbres binaires. De plus, en se basant sur les travaux de Breiman et al. (1984), les coupures par substitution pourraient dans le cadre des arbres TPLDA et CARTGV être utilisées pour le traitement des valeurs manquantes, notion qui n'a pas été abordée dans ce travail de thèse.

Un des avantages des méthodes par arbres est la facilité d'interprétation. Dans la méthode CARTGV, l'arbre final n'étant plus binaire, représenter la règle de prédiction sous la forme d'un arbre ne semble plus très adapté. Il pourrait être pertinent d'utiliser d'autres approches. Une première idée serait d'utiliser une représentation circulaire centrée en ce qui correspond actuellement à la racine de l'arbre. Dans cette nouvelle modélisation, à chaque étape, les nœuds créés seraient ajoutés autour du nœud dont ils sont issus. La structure finale aurait alors une forme similaire à celle d'un flocon.

Actuellement, les méthodes TPLDA, CARTGV et RFGV ont été implémentées en langage `R` pour répondre à des problèmes en classification binaire. Nous envisageons de développer un package `R` contenant toutes les méthodes proposées pour la régression et la classification. D'autre part, les méthodes par arbres et particulièrement les forêts aléatoires sont souvent utilisées pour traiter de gros volumes de données. Il serait donc pertinent d'utiliser le calcul distribué et d'autres langages de programmation, comme par exemple le langage `C`, afin de permettre le traitement de bases de données volumineuses en un temps raisonnable.

Tout au long de ce travail de thèse, nous avons supposé que les groupes de variables étaient connus. Or dans de nombreux problèmes en apprentissage supervisé, il semble naturel et/ou pertinent d'utiliser des groupes de variables pour élaborer une règle de décision. Cependant, les groupes ne sont pas définis et il n'existe pas de méthodes permettant leur identification. C'est le cas par exemple des données d'images ou de type fonctionnelle. Ainsi, une autre piste de recherche serait de proposer une approche qui permette d'identifier de manière automatique les groupes de variables afin d'appliquer ensuite nos méthodes sur les données ainsi regroupées. Dans cette optique, une première idée serait de définir des groupes a priori, de différentes façons, et d'appliquer nos méthodes sur ces données augmentées. Les

"bons" groupes seraient sélectionnés automatiquement par l'algorithme grâce aux scores d'importance. Par exemple, pour des données fonctionnelles, on pourrait utiliser des projections sur des bases d'ondelettes (ou de splines) à différentes résolutions et identifier les bandes discriminantes et la (ou les) "bonne(s)" résolution(s) pour ces bandes.

Par ailleurs, le boosting, méthode d'agrégation introduite pour la première fois par Freund & Schapire (1996), figure souvent avec les forêts aléatoires de Breiman parmi les algorithmes les plus performants dans bons nombres de problèmes concrets d'apprentissage supervisé (citons par exemple, les concours *Kaggle* de prévision). Au cours de cette thèse, nous avons développé des méthodes par arbres et de forêts aléatoires adaptées aux données ayant une structure de groupe. Par la suite, il pourrait être intéressant de proposer des méthodes de type boosting adaptées aux variables groupées.

Dans la dernière partie de la thèse, l'algorithme de clustering que nous proposons requiert la connaissance du nombre de clusters qui en pratique n'est pas toujours garantie. Cette problématique a déjà été abordée par plusieurs auteurs. Par exemple, dans le cadre du clustering spectral, Giulini (2015) propose une approche automatique pour la sélection de ce paramètre. Un axe de recherche serait donc de développer une approche qui permette, à partir de l'algorithme robuste single linkage, de déterminer le nombre de clusters en présence d'outliers. Une autre piste consisterait à utiliser des méthodes d'agrégation du type bagging afin d'obtenir des partitions plus stables.

Bibliography

Alon, U., Barkai, N., Notterman, D. A., Gish, K., Ybarra, S., Mack, D., & Levine, A. J. (1999). Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays. *Proceedings of the National Academy of Sciences*, *96*, 6745–6750.

Altmann, A., Toloşi, L., Sander, O., & Lengauer, T. (2010). Permutation importance: a corrected feature importance measure. *Bioinformatics*, *26*, 1340–1347.

Archer, K. J., & Kimes, R. V. (2008). Empirical characterization of random forest variable importance measures. *Computational Statistics & Data Analysis*, *52*, 2249–2260.

Arias-Castro, E. (2011). Clustering based on pairwise distances when the data is of mixed dimensions. *IEEE Transaction on Information Theory*, *57*, 1692–1706.

Arias-Castro, E., Chen, G., & Lerman, G. (2011). Spectral clustering based on local linear approximations. *Electronic Journal of Statistics*, *5*, 1537–1587.

Auray, S., Klutchnikoff, N., & Rouvière, L. (2015). On clustering procedure and nonparametric mixture estimation. *Electronic Journal of Statistics*, *9*, 266–297.

Bernard, S., Heutte, L., & Adam, S. (2008). Forest-rk: A new random forest induction method. In *International Conference on Intelligent Computing* (pp. 430–437). Springer.

Biau, G. (2012). Analysis of a random forests model. *Journal of Machine Learning Research*, *13*, 1063–1095.

Biau, G., Cadre, B., & Pelletier, B. (2007). A graph-based estimator of the number of clusters. *ESAIM. Probability and Statistics*, *11*, 272–280.

Biau, G., Cadre, B., & Rouvière, L. (2018). Accelerated gradient boosting. ArXiv preprint arXiv:1803.02042.

Biau, G., Fischer, A., Guedj, B., & Malley, J. D. (2016). Cobra: A combined regression strategy. *Journal of Multivariate Analysis*, *146*, 18–28.

Biau, G., & Scornet, E. (2016). A random forest guided tour. *TEST*, *25*, 197–227.

Boulesteix, A.-L., Bender, A., Lorenzo Bermejo, J., & Strobl, C. (2011). Random forest gini importance favours snps with large minor allele frequency: impact, sources and recommendations. *Briefings in Bioinformatics*, *13*, 292–304.

Boulesteix, A.-L., Janitza, S., Kruppa, J., & König, I. R. (2012). Overview of random forest methodology and practical guidance with emphasis on computational biology and bioinformatics. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, *2*, 493–507.

Boulesteix, A.-L., Tutz, G., & Strimmer, K. (2003). A cart-based approach to discover emerging patterns in microarray data. *Bioinformatics*, *19*, 2465–2472.

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

Breiman, L. (1996). Bagging predictors. *Machine learning*, *24*, 123–140.

Breiman, L. (2001). Random forests. *Machine learning*, *45*, 5–32.

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

Brodley, C. E., & Utgoff, P. E. (1995). Multivariate decision trees. *Machine learning*, *19*, 45–77.

Chakraborty, D., & Pal, N. R. (2008). Selecting useful groups of features in a connectionist framework. *IEEE transactions on neural networks*, *19*, 381–396.

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, *16*, 321–357.

Cutler, D. R., Edwards, T. C., Beard, K. H., Cutler, A., Hess, K. T., Gibson, J., & Lawler, J. J. (2007). Random forests for classification in ecology. *Ecology*, *88*, 2783–2792.

Devroye, L., Györfi, L., & Lugosi, G. (1996). *A probabilistic theory of pattern recognition* volume 31. Springer-Verlag.

Díaz-Uriarte, R., & Alvarez de Andrés, S. (2006). Gene selection and classification of microarray data using random forest. *BMC Bioinformatics*, *7*, 3.

Duda, R. O., Hart, P. E., & Stork, D. G. (2012). *Pattern classification*. John Wiley & Sons.

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

Evans, L. C., & Gariepy, R. F. (2015). *Measure theory and fine properties of functions*. CRC press.

Filipovych, R., Resnick, S. M., & Davatzikos, C. (2011). Semi-supervised cluster analysis of imaging data. *NeuroImage*, *54*, 2185–2197.

Freund, Y., & Schapire, R. E. (1996). Experiments with a new boosting algorithm. In *Machine Learning: Proceedings of the Thirteenth International Conference* (pp. 148–156).

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

Genuer, R. (2010). *Forêts aléatoires: aspects théoriques, sélection de variables et applications*. Ph.D. thesis Université Paris Sud-Paris XI.

Genuer, R. (2012). Variance reduction in purely random forests. *Journal of Nonparametric Statistics*, *24*, 543–562.

Genuer, R., & Poggi, J.-M. (2018). Chapter 8: Arbres CART et Forêts aléatoires,Importance et sélection de variables. In M.-B. Myriam, S. G., & T. A. C. (Eds.), *Apprentissage Statistique et Données Massives* (pp. 295–342). Technip.

Genuer, R., Poggi, J.-M., & Tuleau-Malot, C. (2010). Variable selection using random forests. *Pattern Recognition Letters*, *31*, 2225–2236.

Geurts, P., Irrthum, A., & Wehenkel, L. (2009). Supervised learning with decision tree-based methods in computational and systems biology. *Mol. BioSyst.*, *5*, 1593–1605.

Gey, S. (2012). Risk bounds for cart classifiers under a margin condition. *Pattern Recognition*, *45*, 3523 – 3534.

Gey, S., & Nedelec, E. (2005). Model selection for cart regression trees. *IEEE Transactions on Information Theory*, *51*, 658–670.

Ghattas, B. et al. (2000). *Importance des variables dans les méthodes CART*. Universites d'Aix-Marseille II et III.

Giulini, I. (2015). *Generalization bounds for random samples in Hilbert spaces*. Ph.D. thesis Ecole normale supérieure-ENS PARIS.

Giulini, I. (2016). Kernel spectral clustering. ArXiv preprint arXiv:1606.06519.

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. (1999). Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. *Science*, *286*, 531–537.

Gordon, A. D. (1999). Classification, (2nd edition), .

Gregorutti, B., Michel, B., & Saint-Pierre, P. (2013). Correlation and variable importance in random forests. *Statistics and Computing*, (pp. 1–20).

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.

Grimonprez, Q. (2016). *Sélection de groupes de variables corrélées en grande dimension*. Ph.D. thesis Lille 1.

Guo, Y., Hastie, T., & Tibshirani, R. (2006). Regularized linear discriminant analysis and its application in microarrays. *Biostatistics*, *8*, 86–100.

Hartigan, J. A. (1975). *Clustering algorithms*. Wiley New York.

Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The elements of statistical learning*. Springer series in statistics Springer, Berlin.

Huang, D., Quan, Y., He, M., & Zhou, B. (2009). Comparison of linear discriminant analysis methods for the classification of cancer based on gene expression data. *Journal of Experimental & Clinical Cancer Research*, *28*, 149.

Huang, J., Breheny, P., & Ma, S. (2012). A selective review of group selection in high-dimensional models. *Statistical science: a review journal of the Institute of Mathematical Statistics*, *27*.

Jain, A. K., & Dubes, R. C. (1988). *Algorithms for clustering data*. Prentice-Hall, Inc.

Jiang, D., Tang, C., & Zhang, A. (2004). Cluster analysis for gene expression data: A survey. *IEEE Transactions on knowledge and data engineering*, *16*, 1370–1386.

Johnson, S. C. (1967). Hierarchical clustering schemes. *Psychometrika*, *32*, 241–254.

Kass, G. V. (1980). An exploratory technique for investigating large quantities of categorical data. *Applied statistics*, (pp. 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. (2014). Fifty years of classification and regression trees. *International Statistical Review*, *82*, 329–348.

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

MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability* 14 (pp. 281–297). Oakland, CA, USA.

Maier, M., Hein, M., & Von Luxburg, U. (2009). Optimal construction of k-nearest-neigbor graphs for identifying noisy clusters. *Theoritical Computer Science*, *410*, 1749–1764.

McLachlan, G. J., & Basford, K. E. (1988). *Mixture models: Inference and applications to clustering* volume 84. Marcel Dekker.

Meier, L., Geer, S. V. D., & Bühlmann, 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.

Nadler, B., & Galun, M. (2007). Fundamental limitations of spectral clustering. In *Advances in neural information processing systems* (pp. 1017–1024).

Ng, A. Y., Jordan, M. I., & Weiss, Y. (2002). On spectral clustering: Analysis and an algorithm. In *Advances in neural information processing systems* (pp. 849–856).

Nicodemus, K. K. (2011). Letter to the editor: On the stability and ranking of predictors from random forest variable importance measures. *Briefings in Bioinformatics*, *12*, 369–373.

Pedrycz, W. (2002). Collaborative fuzzy clustering. *Pattern Recognition Letters*, *23*, 1675–1686.

Pesch, R., Schmidt, G., Schroeder, W., & Weustermann, I. (2011). Application of cart in ecological landscape mapping: Two case studies. *Ecological Indicators*, *11*, 115–122.

Picheny, V., Servien, R., & Villa-Vialaneix, N. (2016). Interpretable sparse sir for functional data. ArXiv preprint arXiv:1606.00614.

Prasad, A. M., Iverson, L. R., & Liaw, A. (2006). Newer classification and regression tree techniques: Bagging and random forests for ecological prediction. *Ecosystems*, .

Questier, F., Put, R., Coomans, D., Walczak, B., & Heyden, Y. V. (2005). The use of cart and multivariate regression trees for supervised and unsupervised feature selection. *Chemometrics and Intelligent Laboratory Systems*, *76*, 45–54.

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.

Rand, W. M. (1971). Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical association*, *66*, 846–850.

Rousseeuw, P. J., & Kaufman, L. (1990). *Finding groups in data*. Wiley Online Library Hoboken.

Sathyadevi, G. (2011). Application of cart algorithm in hepatitis disease diagnosis. In *Recent Trends in Information Technology (ICRTIT), 2011 International Conference on* (pp. 1283–1287). IEEE.

Satish, D. S., & Sekhar, C. C. (2006). Kernel based clustering and vector quantization for speech segmentation. In *Neural Networks, 2006. IJCNN'06. International Joint Conference on* (pp. 1636–1641). IEEE.

Scornet, E. (2015). *Apprentissage et forêts aléatoires*. Ph.D. thesis Paris 6.

Scornet, E., Biau, G., & Vert, J.-P. (2015). Consistency of random forests. *Ann. Statist.*, *43*, 1716–1741.

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.

Shipp, M. A., Ross, K. N., Tamayo, P., Weng, A. P., Kutok, J. L., Aguiar, R. C., Gaasenbeek, M., Angelo, M., Reich, M., Pinkus, G. S. et al. (2002). Diffuse large b-cell lymphoma outcome prediction by gene-expression profiling and supervised machine learning. *Nature medicine*, *8*, 68.

Shorack, R., & Wellner, J. (1986). *Empirical Processes with Applications to Statistics*. SIAM.

Shotton, J., Fitzgibbon, A., Cook, M., Sharp, T., Finocchio, M., Moore, R., Kipman, A., & Blake, A. (2011). Real-time human pose recognition in parts from single depth images. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on* (pp. 1297–1304). Ieee.

Sjostrand, K., Rostrup, E., Ryberg, C., Larsen, R., Studholme, C., Baezner, H., Ferro, J., Fazekas, F., Pantoni, L., Inzitari, D. et al. (2007). Sparse decomposition and modeling of anatomical shape variation. *IEEE Transactions on Medical Imaging*, *26*, 1625–1635.

Strobl, C., Boulesteix, A.-L., Zeileis, A., & Hothorn, T. (2007). Bias in random forest variable importance measures: Illustrations, sources and a solution. *BMC Bioinformatics*, *8*, 25.

Tai, F., & Pan, W. (2007). Incorporating prior knowledge of gene functional groups into regularized discriminant analysis of microarray data. *Bioinformatics*, *23*, 3170–3177.

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.

Tuleau, C., & Poggi, J.-M. (2006). Classification supervisée en grande dimension applicationa l'agrément de conduite automobile. *Revue de Statistiques Appliquée*, *54*, 41–60.

Vapnik, V. (1995). *The nature of statistical learning theory*. Springer Velag, New York.

Vapnik, V. (1998). *Statistical learning theory*. Wiley-Intersciences.

Villa-Vialaneix, N., Liaubet, L., Laurent, T., Cherel, P., Gamot, A., & SanCristobal, M. (2013). The structure of a gene co-expression network reveals biological functions underlying eqtls. *PloS one*, *8*, e60045.

Von Luxburg, U. (2007). A tutorial on spectral clustering. *Statistics and computing*, *17*, 395–416.

Wager, S. (2014). Asymptotic theory for random forests. ArXiv preprint arXiv:1405.0352.

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

Weston, J., Elisseeff, A., Schölkopf, B., & Tipping, M. (2003). Use of the zero-norm with linear models and kernel methods. *Journal of machine learning research*, *3*, 1439–1461.

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

Witten, I. H., Frank, E., Hall, M. A., & Pal, C. J. (2016). *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann.

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.

Yamanishi, Y., Vert, J.-P., & Kanehisa, M. (2004). Protein network inference from multiple genomic data: a supervised approach. *Bioinformatics*, *20*, i363–i370.

Yengo, L., Jacques, J., & Biernacki, C. (2014). Variable clustering in high dimensional linear regression models. *Journal de la Societe Française de Statistique*, *155*, 19.

Yuan, M., & Lin, Y. (2006). Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, *68*, 49–67.

Zeng, E., Yang, C., Li, T., & Narasimhan, G. (2012). Clustering genes using heterogeneous data sources. In *Computational Knowledge Discovery for Bioinformatics Research* (pp. 67–83). IGI Global.

Zhang, H. H., Liu, Y., Wu, Y., & Zhu, J. (2008). Variable selection for the multicategory svm via adaptive sup-norm regularization. *Electron. J. Statist.*, *2*, 149–167.

INSA de RENNES
Service des Formations

AVIS DU JURY SUR LA REPRODUCTION DE LA THESE SOUTENUE

Titre de la thèse:
Arbres de décision et forêts aléatoires pour variables groupées

Nom Prénom de l'auteur : POTERIE AUDREY

Membres du jury :
- Monsieur BIAU Gérard
- Madame TULEAU Christine
- Monsieur ROUVIERE Laurent
- Monsieur DUPUY Jean-François
- Madame MONBET Valérie
- Monsieur POGGI Jean-Michel
- Monsieur BIERNACKI Christophe

Président du jury : BIAU Gérard

Date de la soutenance : 18 Octobre 2018

Reproduction de la these soutenue

☒ Thèse pouvant être reproduite en l'état
☐ Thèse pouvant être reproduite après corrections suggérées

Fait à Rennes, le 18 Octobre 2018

Signature du président de jury

Le Directeur,

M'hamed DRISSI

Titre : Arbres de décision et forêts aléatoires pour variables groupées

Mots clés : Apprentissage statistique, groupes de variables, arbres de décision, forêts aléatoires, sélection de groupes de variables, clustering

Résumé : Dans de nombreux problèmes en apprentissage supervisé, les entrées ont une structure de groupes connue et/ou clairement identifiable. Dans ce contexte, l'élaboration d'une règle de prédiction utilisant les groupes plutôt que les variables individuelles peut être plus pertinente tant au niveau des performances prédictives que de l'interprétation. L'objectif de la thèse est de développer des méthodes par arbres adaptées aux variables groupées. Nous proposons deux approches qui utilisent la structure groupée des variables pour construire des arbres de décisions. La première méthode permet de construire des arbres binaires en classification. Une coupure est définie par le choix d'un groupe et d'une combinaison linéaire des variables du dit groupe. La seconde approche, qui peut être utilisée en régression et en classification, construit un arbre non-binaire dans lequel chaque coupure est un arbre binaire. Ces deux approches construisent un arbre maximal qui est ensuite élagué. Nous proposons pour cela deux stratégies d'élagage dont une est une généralisation du *minimal cost-complexity pruning*. Les arbres de décision étant instables, nous introduisons une méthode de forêts aléatoires pour variables groupées. Outre l'aspect prédiction, ces méthodes peuvent aussi être utilisées pour faire de la sélection de groupes grâce à l'introduction d'indices d'importance des groupes. Ce travail est complété par une partie indépendante dans laquelle nous nous plaçons dans un cadre d'apprentissage non supervisé. Nous introduisons un nouvel algorithme de clustering. Sous certaines hypothèses classiques, nous obtenons des vitesses de convergence pour le *risque de clustering* de l'algorithme proposé.

Title : Decision trees and random forests for grouped variables

Keywords : Statistical learning, groups of variables, decision trees, random forests, group variable selection, clustering

Abstract: In many problems in supervised learning, inputs have a known and/or obvious group structure. In this context, elaborating a prediction rule that takes into account the group structure can be more relevant than using an approach based only on the individual variables for both prediction accuracy and interpretation. The goal of this thesis is to develop some tree-based methods adapted to grouped variables. Here, we propose two new tree-based approaches which use the group structure to build decision trees. The first approach allows building binary decision trees for classification problems. A split of a node is defined according to the choice of both a splitting group and a linear combination of the inputs belonging to the splitting group. The second method, which can be used for prediction problems in both regression and classification, builds a non-binary tree in which each split is a binary tree. These two approaches build a maximal tree which is next pruned. To this end, we propose two pruning strategies, one of which is a generalization of the minimal cost-complexity pruning algorithm. Since decisions trees are known to be unstable, we introduce a method of random forests that deals with groups of inputs. In addition to the prediction purpose, these new methods can be also use to perform group variable selection thanks to the introduction of some measures of group importance, This thesis work is supplemented by an independent part in which we consider the unsupervised framework. We introduce a new clustering algorithm. Under some classical regularity and sparsity assumptions, we obtain the rate of convergence of the clustering risk for the proposed algorithm.