UNIVERSITÉ DU QUÉBEC

MÉMOIRE PRÉSENTÉ À
L'UNIVERSITÉ DU QUÉBEC À TROIS-RIVIÈRES

COMME EXIGENCE PARTIELLE
DE LA MAÎTRISE EN MATHEMATIQUES ET INFORMATIQUE APPLIQUEES

PAR
ELHADJI DIARAFF DIEGANE DIAGNE

Analyse discriminante et perceptron multicouche-liens formels et applications

Septembre 2019

Table des matières

Liste des tableaux

Table des figures

Avant-propos

Ce document s'intéresse au lien existant entre le perceptron multicouche (réseaux de neurones à plusieurs couches cachées) et l'analyse discriminante qui est une méthode de classification et de prédiction classique. La description de chacune des méthodes permettra de bien distinguer les bases de chacune d'elles. Des applications sur des jeux de données variés seront développées sur la base de modèles d'analyse discrinminante et de perceptron multicouche.

Je tiens à adresser mes sincères remerciements à ma directrice de recherche Mme Nadia Ghazzali ainsi qu'à mon codirecteur M. Jean-François Quessy, pour leur disponibilité et leur appui tant du côté financier que pédagogique. Merci infiniment pour les efforts consentis dans la lecture et des corrections de fond et de forme apportées, sans lesquelles ce document ne pourrait atteindre ce stade de clarté et de précision.

Je reste entièrement reconnaissant à mes parents qui m'ont toujours orienté, appuyé et motivé. Mes remerciements vont aussi à l'endroit de mes frères, mes collègues, mes amis et tous ceux qui ont contribués de près ou de loin à la rédaction de ce mémoire.

Résumé

Le but principal de ce mémoire de recherche est de contribuer à faire le lien entre des méthodes statistiques classiques et neuronales, notamment l'analyse discriminante et le perceptron multicouche (PMC). L'analyse discriminante est une méthode classique de classification qui s'applique sur des données quantitatives à des fins de réduction de la dimensionnalité et de prédiction d'une variable catégorielle. Elle a diverses variantes dont la quadratique (ADQ), la mixte (ADM), la flexible (ADF) ainsi que la pénalisée (ADP). Toutes ces techniques permettent de discriminer des données non linéairement séparables. L'ADQ traite des situations d'hétéroscédasticité (i.e les matrices de variances-covariances intra-groupes (W_k) ne sont pas toutes identiques). L'ADM est utilisée dans les situations où les données peuvent être modélisées par des lois de densité gaussiennes. En ce qui concerne l'ADF, elle est une régression multiple appliquée aux variables explicatives dites prédicateurs sur une matrice indicatrice correspondant à une décomposition de la variable à expliquer Y en variables indicatrices. Elle est substituée à l'ADL au cas où le nombre de prédicateurs est très élevé (Hastie et al., 1994 [1]). En outre, si les variables explicatives sont fortement corrélées, l'ADP devient la méthode idéale du fait de la possibilité de pouvoir perturber la liaison existante entre ces dernières par une pénalisation de la matrice de variance-covariances intra-groupes (Hastie et al., 1995 [2]).

Par ailleurs, le PMC est un modèle de réseau de plusieurs couches formées de neurones artificiels (discriminants linéaires) permettant de résoudre le problème de classification de données non linéairement séparables. Chaque modèle est bien illustré avec les Iris

de Fisher. Ainsi, pour comparer techniquement les modèles, deux jeux de données sont exploités avec chacune des techniques susmentionnées. En ce qui concerne les données provenant du centre de transfusion sanguine d'une ville de Taïwan *RFMT* (4 variables explicatives ventilées sur 748 observations), les meilleurs modèles de prédiction sur la base du Taux de Bonne Classification (TBC) sont l'ADF et l'ADQ (TBC = 81% pour chaque modèle). Cependant, pour le jeu de données *PULSAR* (8 variables explicatives ventilées sur 17898 observations), issu de l'enquête de haute résolution sur l'univers, le meilleur modèle est fourni par le PMC selon les indicateurs TBC (99,995%) et l'aire sous la courbe de ROC ($AUC = 0{,}977$).

Introduction

Les travaux de McCulloch et Pitts (1943) [3] et ceux de Rosenblatt (1958)[4] sur la conception de modèles de neurones artificiels inspiré du neurone physiologique sont les premiers pas vers l'intelligence artificielle. Le physiologiste canadien Hebb (1949) [5] met en place un réseau où les synapses entre neurones sont pondérés par des poids modifiables à des fins d'apprentissage. Cependant, ces systèmes de réseaux n'étaient pas aussi bien structurés en taille (nombre de neurones) et en capacité pour gérer de grands volumes de données. C'est ainsi que les travaux de McCullotch et Pitts et ceux de Hebb ont été contesté par Minsky et Papert (1969) [6] du fait de leur inaptitude à modéliser des situations plus complexes en terme de taille des données. Ainsi, l'idée d'améliorer les premiers modèles de neurones artificiels est en phase avec l'avènement du *Big Data* où le recueil, l'exploitation et la modélisation d'une large gamme de données s'effectuent en un temps record. Cependant, comme les premiers modèles d'apprentissage susmentionnés, les méthodes classiques d'exploitation de données notamment les méthodes de régression et d'analyse discriminante, doivent être relativisées dans la mesure où leur application pourrait avoir un coût en temps et en logistique (capacités des ordinateurs) trop élevé. A cet effet, de nouveaux algorithmes sont développés vers les années 50 en l'occurrence le *« Machine Learning »* ou apprentissage automatique avec le Perceptron Multicouche (PMC), les Supports à Vaste Marge (SVM), les arbres de décision, les forêts aléatoires etc. Le *« Machine Learning »* est une branche de l'intelligence artificielle qui consiste à développer des algorithmes et qui génèrent de l'information de façon automatique sur la base de données souvent

de très grandes tailles (Hinnovic, 2018 [7]). Il existe des algorithmes d'apprentissage supervisé (mise en place de modèles à des fins de prédiction) et non supervisé (mise en place de modèles permettant de caractériser les données en classes d'individus homogènes). Le cas supervisé regroupe l'ensemble des techniques classiques (régression, analyse discriminante etc.) et nouvelles (perceptron multicouche, SVM etc.) qui permettent de prédire la classe ou groupe d'appartenance de toute nouvelle observation d'une population donnée. Ainsi, ce projet s'articule sur une description des méthodes d'Analyse discriminante (AD) et du Perceptron multicouche (PMC), puis une analyse comparative entre elles sur la base d'applications. Le but principal de ce mémoire est de contribuer à faire le lien avec les méthodes statistiques classiques et neuronales à des fins de classification et de prédiction. Il comporte quatre chapitre, le premier chapitre décrit l'ADL et ses quelques variantes (ADQ et ADM), le chapitre 2 traite des généralités de l'ADL notamment l'ADF et l'ADP, le chapitre 3 aborde le fonctionnement du perceptron multicouche et enfin une analyse comparative des modèles de ces différentes techniques est traitée au niveau du chapitre quatre.

Chapitre 1

L'analyse discriminante linéaire de Fisher et ses quelques variantes

1.1 Notations et définitions

Soit un échantillon $(x_1, \cdots, x_n)$ subdivisé en K groupes $\mathcal{G} = (G_1, \cdots, G_K)$ décrits par p variables $(X_1, \cdots, X_p)$ quantitatives nommées prédicteurs. Ces variables permettent d'expliquer ou de prédire les valeurs d'une autre variable dépendante appelée variable à expliquer $Y = (Y_1, \cdots, Y_n) \in \mathbb{R}^n$, avec Y_i la valeur prise par Y mesurée sur le $i$ème individu. Les modalités de la variable Y représentent les groupes. En considérant le cas d'un diagnostic effectué par un médecin pour détecter si une personne est atteinte d'une maladie ou pas, la variable Y prend pour modalités : « Malade » et « Non Malade ».

Dans le cadre du modèle linéaire de façon générale, la relation liant $X = (X_1, \cdots, X_p)$ et Y est donnée par :

$$Y = X\beta + \epsilon \tag{1.1}$$

Avec β le vecteur des coefficients du modèle et ϵ le terme d'erreur.

Notons :

$x_i = (X_1, \cdots, X_p)$, le vecteur ligne du i-è*me* individu ;

$X_j = (x_1, \cdots, x_n)'$, le vecteur colonne de la j−è*me* variable ;

$\pi_k = \frac{n_k}{n}$, la probabilité a priori du groupe G_k ;

La matrice de variance-covariance totale $T(p \text{ x } p) = \frac{1}{n}\sum_{i=1}^{n}(x_i - g)'(x_i - g)$, avec $g = \frac{1}{n}\sum_{i=1}^{n}x_i$, le centre de gravité global ; $g_k = \frac{1}{n_k}\sum_{i=1}^{n_k}x_i$, le centre de gravité du groupe G_k ;

La matrice de variance-covariance intra-groupe $W(p \text{ x } p)$ est donnée par : $W = \sum_{k=1}^{K}\pi_k T_k$,

— La matrice $B(p \text{ x } p)$ de variance-covariance intergroupe : $B = \sum_{k=1}^{K}\pi_k(g_k - g)'(g_k - g)$,

— Décomposition de la matrice de covariance :

$T = W + B$, la covariance totale se décompose en covariance intragroupes (W) et en covariance intergroupes (B).

Distance de Mahalanobis

C'est une distance introduite par Prasanta Chandra Mahalanobis en 1936. Elle permet de calculer la distance entre deux points dans un espace de dimension p en tenant compte de la structure de variance-covariance sur ces p dimensions. La métrique utilisée ici est l'inverse de la matrice de covariance intragroupe W^{-1}.

La distance de Mahalanobis entre un point x au centre de gravité g du groupe G est donnée par l'expression :

$$D^2(x, g) = (x-g)'W^{-1}(x-g)$$

Dans le cas où les variables sont centrées, réduites et indépendantes, la distance euclidienne coïncide avec la distance de Mahalanobis du fait que la matrice de variance-covariances est la matrice identité I_p (Mclachlan, 1999[8]).

1.2 Généralités sur l'analyse discriminante linéaire (ADL)

L'objectif principal de l'analyse discriminante linéaire de Fisher (ADL) est de pouvoir classer de nouveaux individus n'appartenant pas aux données initiales. L'idée se base sur une méthode cherchant une combinaison linéaire des variables X_j qui maximise la ressemblance entre les éléments au sein d'un même groupe. En d'autres termes, il s'agit de trouver une combinaison linéaire des X_j qui maximise l'inertie ou encore la variance intergroupe et donc celle qui minimise la variance intragroupe. On distingue deux approches de l'ADL à savoir celle factorielle ou descriptive et celle probabiliste ou prédictive.

1.2.1 Aspect factoriel

L'analyse factorielle discriminante (AFD) a été introduite par Fisher en 1936 et appliquée aux fleurs de type *setosa, versicolor et virginica* pour les distinguer à partir de leurs caractéristiques physiques. Aujourd'hui, les applications de l'AFD sont nombreuses et concernent des domaines très variés : à partir d'un ensemble de données quantitatives observées sur n individus, il s'agit de caractériser la partition de ces derniers en g classes (Casin, 2015) [9]. Comme les autres méthodes d'analyse factorielle, l'AFD a pour objectif de trouver les X_j qui séparent le plus possible les groupes. En effet, il s'agit de déterminer comment les variables d'entrée ou explicatives permettent de séparer les groupes a priori. Dans le cas de prédicteurs quantitatifs, l'analyse en composantes principales (ACP) joue un rôle prépondérant dans la description des individus et variables en l'absence même de l'information sur les groupes a priori. Cependant, pour réussir à bien discriminer les groupes, l'AFD reste la méthode la plus connue.

Elle cherche de nouvelles variables appelées fonctions discriminantes correspondant à des combinaisons linéaires des p variables explicatives qui permettent de séparer le mieux les K groupes.

La fonction discriminante f définie dans R^p, est de la forme :

$$f(a) = a'\mathrm{X} = a_1\mathrm{X}_1, \cdots, a_p\mathrm{X}_p\,;$$

avec $a = (a_1, \cdots, a_p)' \in \mathbb{R}^p$ le facteur discriminant.

L'AFD cherche donc à trouver a pour que f discrimine le plus possible les groupes, mais également à mesurer la qualité de la discrimination.

Pour répondre à cette problématique, on définit un critère à optimiser qui mesure la capacité d'un axe à discriminer le mieux les groupes.

Un axe est considéré comme discriminant si la détermination des coefficients a_j se base sur la résolution du problème d'optimisation :

$$S(a) = \max\left(\frac{a'Ba}{a'Ta}\right), \tag{1.2}$$

sous la contrainte $a'Ta = 1$.

Littéralement, cela se traduit par la maximisation de la variance intergroupe B et la minimisation de la variance intragroupe W simultanément. Maximiser $S(a)$ équivaut à maximiser la forme quadratique $a'Ba$ sous la contrainte $a'Ta = 1$.

Le lagrangien permettant d'optimiser l'équation (1.2) est donné par :

$$\mathcal{L} = a'Ba - \lambda(a'Ta - 1);$$

avec λ le multiplicateur de Lagrange. Les valeurs optimales de S sont obtenues en annulant les dérivées premières de $\mathcal{L}$ par rapport à a :

$$\frac{\partial \mathcal{L}}{\partial a} = \frac{\partial [a'Ba - \lambda(a'Ta - 1)]}{\partial a};$$
$$= 2Ba - 2\lambda Ta;$$

$$\frac{\partial \mathcal{L}}{\partial a} = 0 \text{ équivaut à } 2Ba - 2\lambda Ta = 0. \tag{1.3}$$

La première dérivée partielle de $\mathcal{L}$ donne :

$$Ba = \lambda Ta. \tag{1.4}$$

En multipliant d'une part et d'autre l'équation (1.4) par a', on obtient : $a'Ba = \lambda a'Ta$. En tenant compte de la contrainte $a'Ta = 1$, on se retrouve avec $\lambda = a'Ba$. La valeur de λ obtenue après dérivée première de $\mathcal{L}$ correspond donc à la plus grande valeur propre (λ_1). La matrice de variance covariance totale T étant définie positive, l'équation 1.4 devient :

$$T^{-1}Ba_1 = \lambda_1 a_1.$$

Ainsi, a_1 représente le vecteur propre correspondant à la plus grande valeur propre λ_1 de $T^{-1}B$.

La deuxième valeur propre λ_2 est obtenue en maximisant $a'Ba$ sous les contraintes : $a_2'Ta_2 = 1$ et $a_2'Ta_1 = 0$ (conditions d'orthogonalités des vecteurs propres). Cela revient à annuler la dérivée seconde du lagrangien :

$$\mathcal{L}_2 = a_2'Ba_2 - \lambda_2(a_2'Ta_2) - \mu_2 a_2'Ta_1,$$

avec λ_2 et μ_2 des multiplicateurs de Lagrange.

Exemple d'analyse factorielle discriminante dans le cas de deux groupes : Dans le cas de deux groupes linéairement séparables, la situation devient plus simple car un seul axe suffit pour séparer ces groupes. La variance-covariance $B = (b_{jj'})$ entre les deux groupes est donnée par :

$$b_{jj'} = n_1(X_{1j} - \bar{X}_j)(X_{1j'} - \bar{X}_{j'}) + n_2(X_{2j} - \bar{X}_j)(X_{2j'} - \bar{X}_{j'}). \tag{1.5}$$

Or, on a les relations :

$$\bar{X}_j = \frac{n_1}{n}\bar{X}_{1j} + \frac{n_2}{n}\bar{X}_{2j};$$
$$\bar{X}_{j'} = \frac{n_1}{n}\bar{X}_{1j'} + \frac{n_2}{n}\bar{X}_{2j'}.$$

En remplaçant $\bar{X}_j$ et $\bar{X}_{j'}$ par leurs valeurs dans (1.5), on montre que :

$$b_{jj'} = \frac{n_1 n_2}{n}(\bar{X}_{1j} - \bar{X}_{2j})(\bar{X}_{1j'} - \bar{X}_{2j'}) \tag{1.6}$$

$$\text{En posant } C = \sqrt{\frac{n_1 n_2}{n}}(\bar{X}_{11} - \bar{X}_{21}, \cdots, \bar{X}_{1p} - \bar{X}_{2p})'.$$

Les expressions de la fonction discriminante f et du facteur discriminant a sont données par :

$$a = T^{-1/2}C = \sqrt{\frac{n_1 n_2}{n}}T^{-1/2}(\bar{X}_{11} - \bar{X}_{21}, \cdots, \bar{X}_{1p} - \bar{X}_{2p})'; \tag{1.7}$$
$$f(a) = a'X = \sqrt{\frac{n_1 n_2}{n}}(\bar{X}_{11} - \bar{X}_{21}, \cdots, \bar{X}_{1p} - \bar{X}_{2p})T^{-1/2}X. \tag{1.8}$$

1.2.2 Aspect probabiliste de l'analyse discriminante

Contrairement à l'ADF, le but de l'aspect probabilité est plutôt prédictif que factoriel. En effet, l'échantillon utilisé dans l'analyse discriminante probabiliste est

considéré comme une base d'apprentissage. Cela suppose que toute l'information disponible sur les données initiales est bien maîtrisée. Ainsi le problème se trouve sur l'appartenance d'un nouveau élément x à un des K groupes. Pour prédire le groupe de x, l'aspect probabiliste de l'analyse discriminante se base sur des règles d'affectation fondées sur des arguments de calcul des probabilités. Pour cela, on définit une approche basée sur les règles de décision bayésienne qui permet de déterminer le groupe d'appartenance le plus probable pour l'élément x.

Cadre probabiliste

Soient $(\pi_1, \cdots, \pi_K)$ la distribution de la variable à expliquer Y où $\pi_k = P(G_k)$ est la probabilité a priori du groupe G_k, avec $\sum_{k=1}^{K} \pi_k = 1$. On s'intéresse ici au théorème de Bayes qui s'appuie sur la densité conditionnelle de la loi de x connaissant son groupe G_k et des probabilités a priori pour calculer les probabilités a posteriori.

– ***Probabilité a priori*** $\pi_k = P(G_k)$: Il s'agit de la probabilité qu'un élément x de l'échantillon appartienne à un groupe donné G_k. Les proportions observées dans l'échantillon d'apprentissage peuvent fournir une estimation des $P(G_k)$. Il y a une équiprobabilité a priori pour tous les éléments d'un même groupe G_k, i.e tous les éléments d'un groupe G_k ont la même probabilité d'appartenir à G_k :

$$P(G_k) = \frac{n_k}{n}.$$

– ***Densité conditionnelle :*** $f_k(\mathrm{x}) = P(\mathrm{x}/G_k)$: La probabilité d'un élément $\mathrm{x} = (\mathrm{x}_1, \cdots, \mathrm{x}_p)$ connaissant son groupe G_k.

– **Probabilité a posteriori** : Dans la démarche de la prédiction, le groupe d'appartenance n'est pas connu. Ainsi, sur la base de la disponibilité de l'information sur les variables explicatives X_j, l'idée de l'approche probabiliste se résume au calcul de la probabilité d'appartenance au groupe G_k, $P(G_k/\mathrm{x})$ appelée la Probabilité a posteriori.

Cette dernière se calcule d'après le théorème de Bayes en utilisant π_k et $P(\mathrm{x}/G_k)$ respectivement la probabilité a priori et la probabilité conditionnelle de x sachant qu'il est dans le groupe G_k.

Théorème de Bayes

Ce théorème en l'honneur du Mathématicien Statisticien britannique Thomas Bayes (1702-1761) est un résultat de base en théorie des probabilités. En classification, il permet d'exprimer une probabilité a posteriori d'appartenance à un groupe pour un élément donné en fonction de la probabilité a priori et de la probabilité conditionnelle. La règle de classification de Bayes suppose que pour tout groupe G_k, le vecteur x des observations suit une loi de densité $f_k(\mathrm{x}) = P(\mathrm{x}/G_k)$.

Définition :

La règle de décision bayésienne pour l'affectation d'un individu x dans un groupe donné fait intervenir les probabilités a posteriori, les probabilités a priori et les coûts de mauvaise classification. Deux situations se présentent selon les données en question : Le cas où les coûts de mauvaise classification sont connus et le cas où ils ne le sont(Ulmo, 1973 [10]).

Pour le dernier cas, l'affectation se base sur les probabilités a posteriori fournies par le théorème de Bayes. Un élément x est affecté au groupe ayant la plus grande probabilité a posteriori $P(G_k/\mathrm{x})$ donnée par :

$$P(G_k/\mathrm{x}) = \frac{P(\mathrm{x}/G_k)\pi_k}{\sum_{k=1}^{K} P(\mathrm{x}/G_k)\pi_k} \tag{1.9}$$

Après le calcul des $P(G_k/\mathrm{x})$, l'élément x sera affecté au groupe G_o tel que :

$$P(G_o/\mathrm{x}) \geq P(G_k/\mathrm{x}) \text{ pour tout k } \in (1, \cdots, K)$$

Autrement dit, x est affecté au groupe G_o tel que :

$$P(G_o/\mathrm{x}) = \max_k P(G_k/\mathrm{x}) = \max_k \left[\frac{P(\mathrm{x}/G_k)\pi_k}{\sum_{k=1}^{K} P(\mathrm{x}/G_k)\pi_k} \right]$$

Notion de coût moyen de mauvaise classification

Pour un nouvel individu x, on peut noter éventuellement la possibilité d'erreur de classification ; soit $\mathrm{x} \in G_k$ alors qu'il est affecté à G_j ou inversement pour $k = \{1, \cdots, K\}$ et $j = \{1, \cdots, K\}$; $k \neq j$.
Notons $C(j/k)$ le coût d'affecter un élément x dans le groupe G_j alors qu'il appartient au groupe G_k, appelé coût de mauvaise classification

$$\begin{cases} C(j/k) = 0, & j = k \\ \\ C(j/k) \geq 0, & j \neq k \end{cases}$$

En d'autres termes, le coût associé à l'affectation de x dans le groupe G_j alors qu'il provient de ce groupe est nul. Par contre, le coût associé à l'affectation de x dans G_j alors qu'il appartient au groupe G_k est positif.
Le coût moyen résultant de l'affectation d'un individu dans le groupe G_j est donné par :

$$\bar{C}(j) = \sum_{k=1}^{K} C(j/k) P(\mathrm{x}/G_k) = \sum_{k=1}^{K} C(j/k) \int_{\mathrm{x}/G_j} f_k(\mathrm{x}) d\mathrm{x}$$

avec

$$\int_{\mathrm{x}/G_j} f_k(\mathrm{x}) d\mathrm{x},$$

, la probabilité d'affecter x dans le groupe G_j alors qu'il est dans G_k.

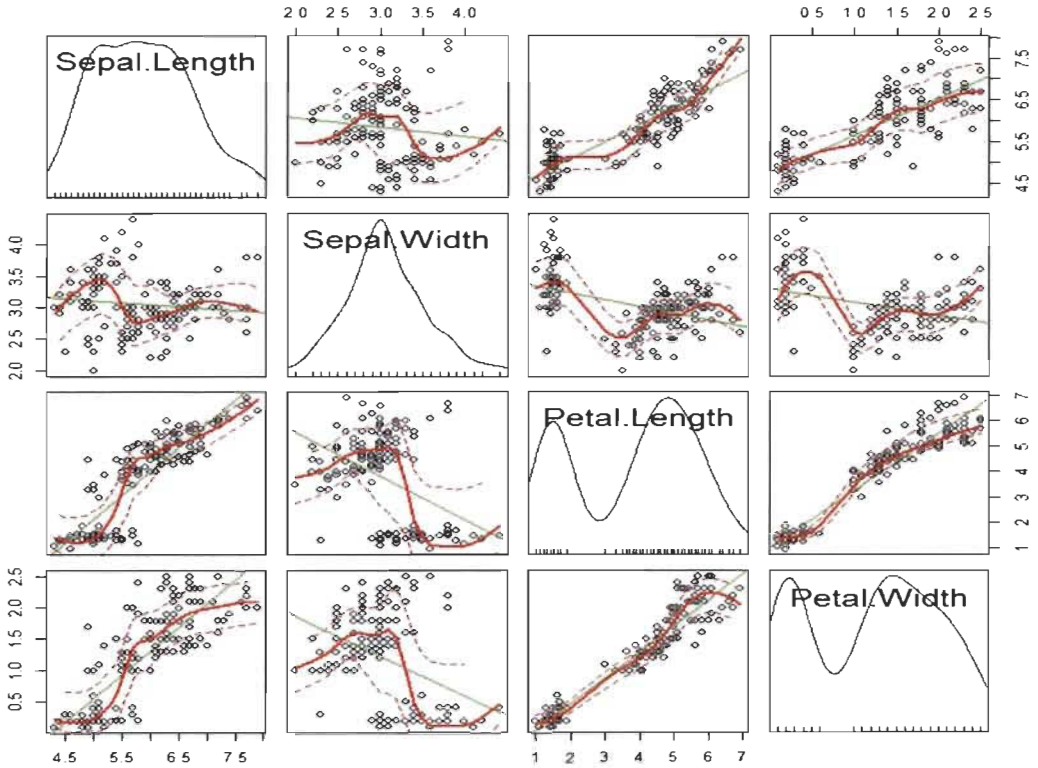
1.3 Application de l'ADL sur les données Iris de Fisher

Le logiciel R intègre diverses fonctions permettant de compiler des programmes en apprentissage supervisé. La fonction lda du package "Mass" de R permet de faire une analyse discriminante linéaire.

Dans cette partie, nous illustrons la méthode ADL sur le jeu de données Iris de Fisher afin de mieux comprendre le fonctionnement de la méthode et l'interprétation de ses résultats. Le jeu de données Iris est constitué de quatre variables quantitatives explicatives notamment *Sepal.Length* (longueur des sépales), *Sépal.Width* (largeur des sepales), Petal.Length (longueur des pétales) et Petal.Width (largeur des pétales) et une variable catégorielle à expliquer (Species) mesurées sur 150 espèces d'Iris de types setosa, virginica et versicolor.

Parmi les hypothèses du modèle linéaire en général et de l'analyse discriminante linéaire en particulier, il y a la normalité des données, le Graphique 1.1 représente en diagonale, la distribution de chacune des quatre variables explicatives. Hors de la diagonale, on retrouve les nuages de points permettant d'expliquer la corrélation existante entre deux variables. Seule la variable *Sepal.Width* a une distribution proche de celle d'une loi normale. Un nuage où les points sont groupés et situés sur une même diagonale indique une forte corrélation positive ou négative entre les deux variables. C'est le cas des variables *Petal.Length* et *Petal.Width* de même que *Sepal.Length* et *Petal.Length*. Un nuage couvrant un grand espace laisse apparaître une faible corrélation entre les variables (cas des variables *Sepal.Width* et *Petal.Length*). Par contre, si le nuage n'a aucune forme particulière, il n'existe aucune corrélation entre les deux variables.

Graphique 1.1 - Diagramme de dispersion matricielle des données Iris



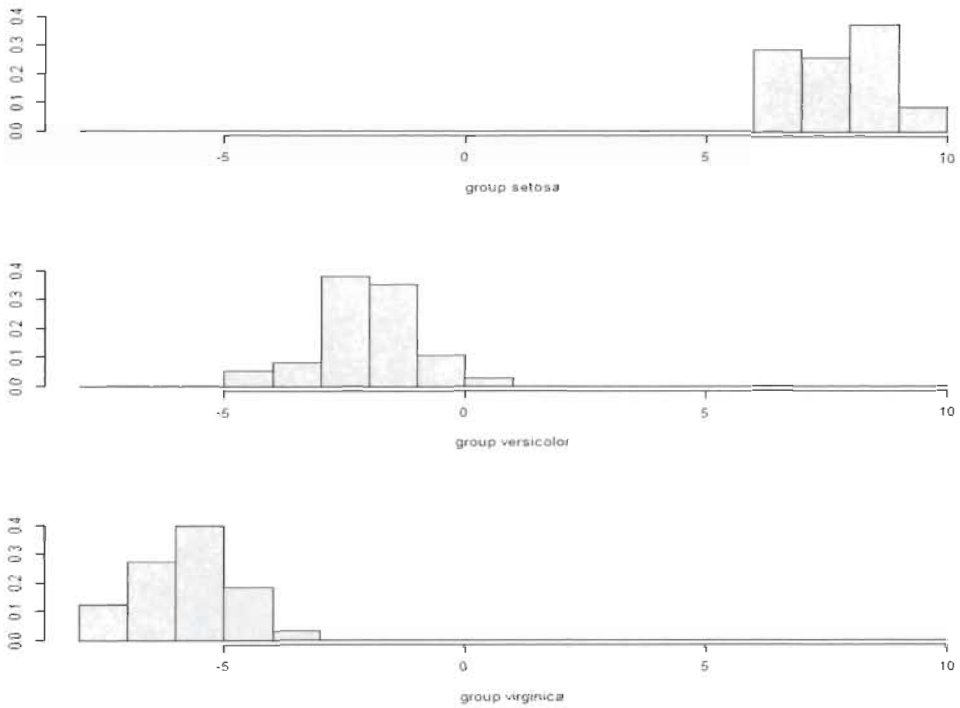
L'objectif de l'analyse discriminante factorielle étant de réduire la dimensionnalité comme les autres méthodes factorielles, nous cherchons les axes (2 axes du fait que K=3) qui sont combinaisons linéaires des variables permettant de maximiser la variance intergroupe. Les facteurs discriminants LD1 et LD2 obtenus sont données par le Tableau 1.1.

Tableau 1.1 – Coefficients des axes discriminants

	LD1	LD2
Sepal.Length	0,8293776	0,02410215
Sepal.Width	1,5344731	2,16452123
Petal.Length	-2,2012117	-0,93192121
Petal.Width	-2,8104603	2,83918785
% d'inertie	99,12	0,88

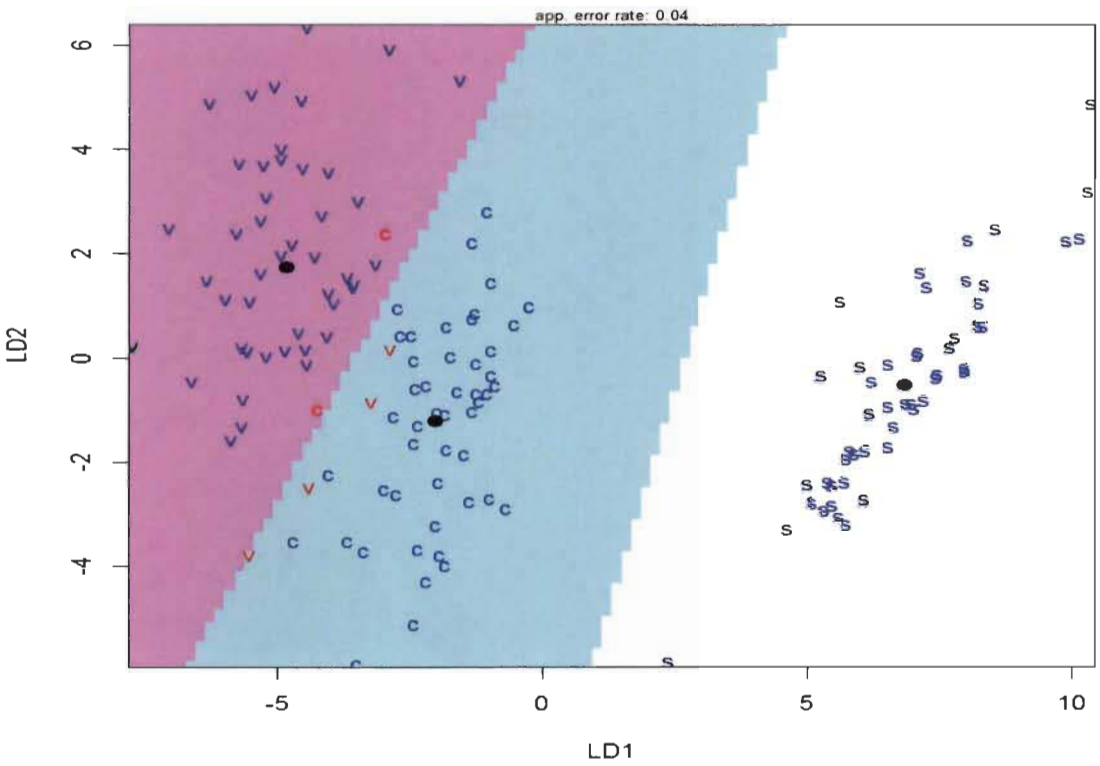
La première fonction discriminante $f_{LD1} = 0,8293776 * Sepal.Length + 1,5344731 * Sepal.Width - 2,2012117 * Petal.Length - 2,8104603 * Petal.Width$ restitue 99,12% de la variance totale. L'histogramme des valeurs empilées de $LD1$ représentant la distribution des trois groupes selon les coefficients de la première fonction discriminante montre une bonne séparation des fleurs de type *setosa* par rapport aux autres types de fleurs. Par contre, $LD1$ ne parvient pas à séparer parfaitement les groupes *versicolor* et *virginica*. En effet, un léger chevauchement entre les groupes *virginica* et *versicolor* est observé (voir Graphique 1.2).

Graphique 1.2 - Histogramme des valeurs empilées de LD1 Iris



Le taux de chevauchement des groupes *virginica* et *versicolor* représente 4% des espèces (voir Graphique 1.3). En d'autres termes, le taux d'espèces appartenant au groupe des *virginica* considérés comme des *versicolor* et vice versa représente 4% du nombre total de fleurs. Dans la suite du document, des indicateurs comme le Taux de Bonne Classification (TBC) et son complémentaire Taux de Mauvaise Classification (TMC) seront utilisés pour évaluer et comparer la performance des modèles.

Graphique 1.3 – Frontière des groupes avec la fonction lda



1.4 Quelques avantages et limites de l'analyse discriminante linéaire

L'analyse discriminante linéaire est une méthode de prédiction assez simple du fait que les frontières des groupes sont données par des droites, un nouvel individu est affecté au groupe pour lequel, la distance au sens de Mahalanobis au centre de gravité dudit groupe, est la plus faible. L'ADL suppose également une égalité des matrices de covariances intergroupes, ce qui rend facile l'estimation des paramètres.

Cependant, elle présente certaines limites notamment :

- Dans des situations d'hétéroscédasticité, les frontières de groupes fournies par l'ADL ne permettent pas une meilleure séparation de ces derniers du fait de la variation de

la dispersion des éléments d'un groupe à un autre ;

- Pour le cas de grands jeux de données comme en reconnaissance d'image, des droites linéaires ne permettent pas souvent d'avoir une bonne séparation des groupes. L'analyse discriminante quadratique, qui sera présentée dans la section suivante, pourrait donner de meilleurs résultats dans de telles situations ;

- De même, l'ADL n'est pas la méthode idéale au cas où les variables explicatives sont fortement corrélées. Dans de telles situations, il est nécessaire de la régulariser afin de perturber la corrélation des prédicteurs pour aboutir à des résultats plus probants (Hastie et al., 1995 [2]).

Pour contourner ces limites, il existe d'autres méthodes visant à étendre l'ADL afin d'avoir une meilleure classification notamment l'analyse discriminante flexible (ADF) et l'analyse discriminante pénalisée (ADP).

1.5 L'Analyse Discriminante Quadratique (ADQ)

1.5.1 Définition

Dans la situation où les matrices de variance-covariances intra-groupes (W_k) ne sont pas identiques pour tous les groupes, l'analyse discriminante linéaire reste limitée car une droite linéaire ne peut pas bien dissocier deux groupes. Ainsi, une fonction quadratique (ellipse, parabole) serait plus robuste pour la discrimination des groupes d'où l'analyse discriminante quadratique. Cependant, l'ADQ ne fonctionne pas bien lorsque les densités de probabilités des groupes sont très différentes de la densité d'une distribution gaussienne.

Rappelons que la fonction de densité d'une loi gaussienne $N(\mu_k, W_k)$ d'un vecteur x du groupe G_k est donnée par :

$$f_k(\mathrm{x}) = \frac{1}{(2\pi)^{\mathrm{p}/2}|W_k|^{1/2}} \exp -[\tfrac{1}{2}D^2(\mathrm{x}, \mu_k)]\,;$$

avec $|W_k|$, le déterminant de la matrice de variance-covariance du groupe G_k et $D^2(\mathrm{x}, \mu_k)$, la distance de Mahalanobis entre x et μ_k.

$$\log \pi_k f_k(\mathrm{x}) = \log(\pi_k) - \tfrac{\mathrm{p}}{2}\log(2\pi) - \tfrac{1}{2}\log|W_k| - \tfrac{1}{2}D^2(\mathrm{x}, \mu_k)$$

L'expression $-\frac{\mathrm{p}}{2}\log(2\pi)$ étant différent de k, on définit la fonction quadratique pour chaque groupe G_k :

$$q_k(\mathrm{x}) = \log(\pi_k) - \frac{1}{2}\log|W_k| - \frac{1}{2}D^2(\mathrm{x}, \mu_k). \tag{1.10}$$

1.5.2 Règle de classification

L'analyse discriminante quadratique reste pertinente dans des situations d'hétéroscédasticité i.e les W_k sont distinctes. Dans chaque groupe G_k, les valeurs estimées des paramètres μ_k et W_k par la méthode du vraisemblance (Ghojogh et Crowley, 2015 [11]) sont :

$$\begin{cases} \widehat{\mu_k} = \frac{1}{n_k}\sum_{i \in G_k} \mathrm{x}_i \\ \\ \widehat{W_k} = \frac{1}{n_k}\sum_{i \in G_k}(\mathrm{x}_i - g_k)(\mathrm{x}_i - g_k)' \end{cases}$$

En considérant les valeurs estimées, on a : $q^*(\mathrm{x}) = \log(\pi_k) - \frac{1}{2}\log|\widehat{W_k}| - \frac{1}{2}D^2(\mathrm{x}, \widehat{\mu_k})$, la fonction quadratique de classement du groupe G_k.

Ainsi, maximiser $\pi_k f_k(x)$ revient à maximiser $\log(\pi_k f_k(\mathrm{x}))$.

L'élément x est ainsi affecté au groupe G_k vérifiant :

$$\max_{k=1,\cdots,K}[q_k^*(\mathrm{x})].$$

1.5.3 Cas particulier de deux groupes

Dans le cadre de deux groupes G_1 et G_2, un vecteur aléatoire X est pris pour un individu x pour lequel on a les fonctions de densités $f_1(\mathrm{x})$ et $f_2(\mathrm{x})$. L'élément x est classé dans l'un des groupes G_1 et G_2 répondant au critère de Bayes. En considérant une partition R_1 et R_2 de l'ensemble $(\mathrm{x}_1,\mathrm{x}_2,\cdots,\mathrm{x}_n)$:

$$R_1 = \left\{\mathrm{x};\ \frac{f_1(\mathrm{x})}{f_2(\mathrm{x})} \geq 1\right\} = \left\{\mathrm{x};\ \log\frac{f_1(\mathrm{x})}{f_2(\mathrm{x})} \geq 0\right\}$$

$$R_2 = \left\{\mathrm{x};\ \frac{f_1(\mathrm{x})}{f_2(\mathrm{x})} < 1\right\} = \left\{\mathrm{x};\ \log\frac{f_1(\mathrm{x})}{f_2(\mathrm{x})} < 0\right\}$$

Si x$\in R_1$, donc l'élément correspondant sera affecté au groupe G_1 ou dans G_2 dans le cas contraire (Bose et al., 2015) [12]. Si f_1 et f_2 sont gaussiennes de moyennes μ_1 et μ_2, de matrices de covariances intra-groupes correspondantes W_1 et W_2 respectivement et de même π_k i.e $\pi_1 = \pi_2 = 1/2$ on a :

$$\begin{aligned}
\log\frac{f_1(\mathrm{x})}{f_2(\mathrm{x})} &= \frac{1}{2}\log\left(\frac{|W_2|}{|W_1|}\right) + \frac{1}{2}\left[(\mathrm{x}-\mu_2)'W_2^{-1}(\mathrm{x}-\mu_2) - (\mathrm{x}-\mu_1)'W_1^{-1}(\mathrm{x}-\mu_1)\right] \\
&= \frac{1}{2}\log\left(\frac{|W_2|}{|W_1|}\right) + \frac{1}{2}\Delta_d^2 \\
&= \frac{1}{2}\left(\log\left(\frac{|W_2|}{|W_1|}\right) + \Delta_d^2\right)
\end{aligned}$$

Avec $\Delta_d^2 = (\mathrm{x}-\mu_2)'W_2^{-1}(\mathrm{x}-\mu_2) - (\mathrm{x}-\mu_1)'W_1^{-1}(\mathrm{x}-\mu_1)$, la différence des carrés des distances de Mahalanobis de x aux points moyens des groupes G_1 et G_2.

Ainsi, selon Bose et al. (2015) [12], la règle de décision de l'ADQ devient :

$$R_1 = \left\{ \frac{1}{2}\log(\frac{|W_2|}{|W_1|}) + \frac{1}{2}\Delta_d^2 \geq 0 \right\}, \text{ pour } |W_2| \neq |W_1| \tag{1.11}$$

En cas d'égalité des matrices de variances-covariances, si $|W_2| = |W_1|$, on se retrouve avec : $R_1 = \{\Delta_d^2 \geq 0\}$, cas de l'ADL.

1.6 Application de l'ADQ aux Iris de Fisher

La fonction *qda* du package *MASS* de R permet d'effectuer une analyse discriminante quadratique. Pour le cas des données Iris de Fisher, comme la fonction lda, qda permet de bien séparer les *setosa* des autres types de fleurs. Cependant, le chevauchement entre *virginica* et *Vversicolor* est toujours observé, mais à un niveau plus faible que celui de l'ADL, soit 2% des espèces (voir Tableau 1.2). En effet, deux espèces de type *versicolor* sont classées comme *virginica* et une espèce de type *virginica* est classée comme *versicolor*.

Tableau 1.2 - Matrice de confusion de l'ADQ des Iris de Fisher

		Classe réelle		
		Setosa	versicolor	virginica
Classe prédite	setosa	50	0	0
	versicolor	0	48	1
	virginica	0	2	49
Taux de mauvaise classification			**2%**	

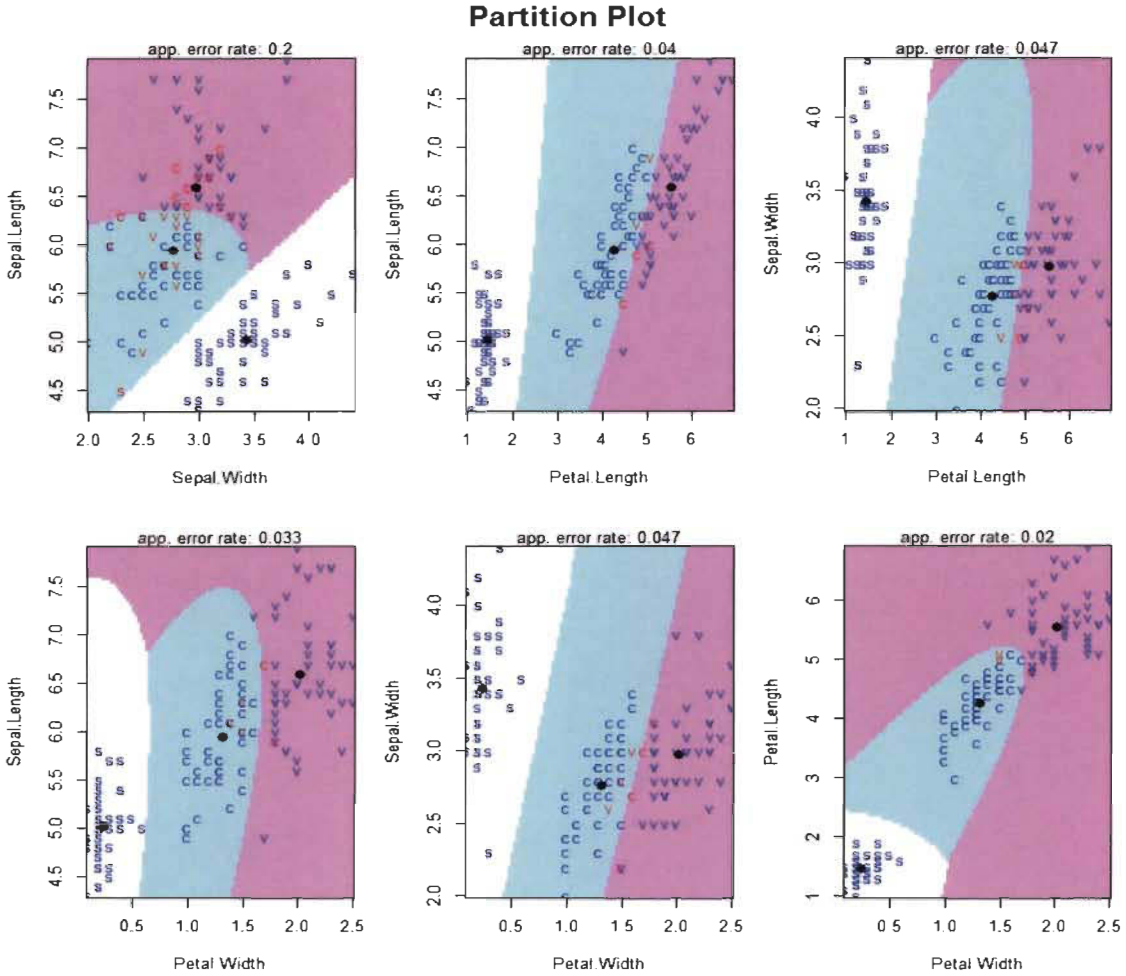
En considérant la distribution des espèces par couples de variables, nous pouvons visualiser les frontières des groupes définies par la fonction qda et les erreurs de clas-

sification correspondantes. Il apparait que plus les variables sont corrélées (Tableau 1.3), plus la fonction qda assure une bonne séparation des groupes. En effet, le taux de mauvaise classification est plus élevé (20%) au niveau du nuage des variables les moins corrélées notamment *Sepal.Length* et *Sepal.Width* (cor = -0,12). Par contre, le taux d'erreur le plus faible (2%) est obtenu avec le nuage des variables *Petal.Length* et *Petal.Width* (cor = 0,96). Ces résultats, représentés dans le Graphique 1.4 montre que contrairement à l'ADL, l'ADQ est plus performante au cas où les variables explicatives sont fortement corrélées entre elles pour les Iris de Fisher.

Tableau 1.3 – Matrice de corrélation des données Iris de Fisher

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width
Sepal.Length	1	-0.118	0.872	0.818
Sepal.Width	-0.118	1	-0.428	-0.366
Petal.Length	0.872	-0.428	1	0.963
Petal.Width	0.818	-0.366	0.963	1

Graphique 1.4 - Frontière des groupes avec la fonction qda



1.7 L'analyse discriminante par mélanges gaussiens (ADM)

1.7.1 Notion de modèle de mélange, cas gaussien

Un modèle est dit de mélange si la population étudiée, subdivisée en K groupes, est modélisée par plusieurs lois de densité $f(\mathrm{x}) = \sum_{k=1}^{K} \pi_k f_k(\mathrm{x})$; les π_k représentent

les proportions du mélange, $\sum_{k=1}^{K} \pi_k = 1$.

Soit $(\mathrm{X}_j, G_k)_{1 \leq j \leq q}$ constitué de q vecteurs aléatoires X_j et de groupes d'individus G_k. On parle de modèle de mélange gaussien si les densités f_k sont des lois gaussiennes de moyennes μ_k et de même covariance intra-groupes W. Ce dernier est l'un des plus simples à utiliser parmi les modèles de mélange du fait qu'il donne la possibilité de travailler sous l'hypothèse d'homoscédasticité, c'est-à-dire sur les mêmes covariances intra-groupes et ainsi estimer un seul W au lieu de trouver les paramètres estimés des covariances intra-groupes dans chaque groupe. L'ADM fonctionne en partitionnant chaque groupe G_k en R_k sous groupes G_{kr}, $r = 1, \cdots, R_k$ (Hastie et al., 2001 [13]). Les matrices de covariances (W_{kr}) des sous groupes sont également identiques.

Les probabilités de mélange π_{kr} sont estimées par : $\widehat{\pi_{kr}} = \frac{n_{kr}}{n_k}$, avec n_{kr} et n_k l'effectif du sous groupe G_{kr} et l'effectif de G_k respectivement.

Dans chaque groupe, la densité de mélange est donnée par :

$$f_k(\mathrm{x}) = P(X = \mathrm{x}/G = k) = (2\pi)^{-\frac{p}{2}} |W|^{-\frac{1}{2}} \sum_{r=1}^{R_k} \pi_{kr} \exp[-D^2(\mathrm{x}, \mu_{kr})/2] \qquad (1.12)$$

Avec π_{kr} et μ_{kr}, la proportion du mélange et la moyenne du sous groupe G_{kr}, et $D^2(\mathrm{x}, \mu_{kr})$, la distance de Mahalanobis entre x et μ_{kr}. Les paramètres de ce modèle sont estimés à l'aide de l'algorithme "Expectation-Maximisation" (EM) de Dempster (2015) [14].

1.7.2 Estimation des paramètres dans un modèle gaussien

A partir d'un échantillon $(\mathrm{x}_1, \cdots, \mathrm{x}_n)$ de $\mathbb{R}^p$, on peut estimer le paramètre $\theta = (\pi_1, \cdots, \pi_k, \mu_1, \cdots, \mu_k, W_1, \cdots, W_k)$ d'un modèle gaussien $N(\mu_k, W_k)$ à l'aide de la maximisation de la vraisemblance $L(\theta) = \prod_{i=1}^{n} f_X(\mathrm{x}_i) = \prod_{k=1}^{K} \prod_{i \in G_k} \pi_k f_k(\mathrm{x}_i)$, ce qui revient aussi à maximiser $log(L(\theta))$.

La log vraisemblance s'écrit :

$$\begin{aligned}
\log(L(\theta)) &= \log(\prod_{i=1}^{n} f_X(\mathrm{x}_i)) \\
&= \log(\prod_{k=1}^{K} \prod_{i \in G_k} \pi_k f_k(\mathrm{x}_i)) \\
&= \sum_{k=1}^{K} \sum_{i \in G_k} \log(\pi_k f_k(\mathrm{x}_i)); \\
\log(\pi_k f_k(x)) &= \log(\pi_k) + \log(f_k(\mathrm{x})) \\
&= \log(\pi_k) + \log(\frac{1}{(2\pi)^{\mathrm{p}/2}|W|^{1/2}} \exp -[D^2(\mathrm{x}, \mu_k)/2]) \\
&= \log(\pi_k) - \frac{\mathrm{p}}{2}\log(2\pi) - \frac{1}{2}\log|W| - \frac{1}{2}D^2(\mathrm{x}, \mu_k)
\end{aligned}$$

Pourvu que $-\frac{\mathrm{p}}{2}\log(2\pi)$ n'est pas fonction de k, maximiser $\log(\pi_k f_k(x))$ revient à maximiser l'expression :

$$\log(\pi_k) - \tfrac{1}{2}\log|W| - \tfrac{1}{2}D^2(\mathrm{x}, \mu_k).$$

Selon Hastie et al., 2001) [13], les valeurs estimées sont donc données par :

$$\hat{\pi}_k = \frac{n_k}{n}$$

$$\hat{\mu}_k = \frac{1}{n_k}\sum_{i \in G_k} x_i$$

$$\hat{W} = \frac{1}{n}\sum_{k=1}^{K}\sum_{i \in G_k} D^2(\mathrm{x}, \mu_k) = \frac{1}{n}\sum_{k=1}^{K}\sum_{i \in G_k}(\mathrm{x} - \mu_k)'(\mathrm{x} - \mu_k), \text{un estimateur biaisé de } W$$

i.e $E(\hat{W}) \neq W$.

Un estimateur non biaisé de W est donné par Hastie et al. (2001) [13].

$$\hat{W}\frac{1}{n-K}\sum_{k=1}^{K}\sum_{i\in G_k}(\mathrm{x}-\mu_k)'(\mathrm{x}-\mu_k)$$

En se basant sur la probabilité a postériori du théorème de Bayes de l'équation 1.9, on a : $P(G_k/\mathrm{x}) = \frac{\pi_k P(\mathrm{x}/G_k)}{\sum_{k=1}^{K} P(\mathrm{x}/G_k)\pi_k} = \frac{\pi_k P(\mathrm{x}/G_k)}{P(\mathrm{x})}$.
Etant donné que $P(\mathrm{x})$ est connue, $\max[P(G_k/\mathrm{x})] \sim \max[\pi_k P(\mathrm{x}/G_k)] = \max[\pi_k f_k(\mathrm{x})]$.
D'après l'expression de la fonction de densité définie à l'équation (1.12), maximiser $f_k(\mathrm{x})$ revient à maximiser $\sum_{r=1}^{R_k} \pi_{kr} \exp[-D^2(\mathrm{x}, \mu_{kr})/2]$.
D'où $\max[P(G_k/\mathrm{x})] \sim \max\left[\pi_k \sum_{r=1}^{R_k} \pi_{kr} \exp[-D^2(\mathrm{x}, \mu_{kr})/2\right]$ et l'élément x est affecté au groupe G_k qui maximise $P(G_k/\mathrm{x})$ (Hastie et al., 1995) [15].

1.7.3 Estimation des paramètres d'un modèle de mélange gaussien par l'algorithme EM

Considérons l'échantillon $(\mathrm{x}_1, \cdots, \mathrm{x}_n)$ de $\mathbb{R}^p$, composé de K groupes suivant une loi de paramètre $\theta = (\theta_1, \cdots, \theta_K)$. Dans le cadre d'un modèle de mélange de densité, l'estimation des paramètres par la méthode de la maximisation de vraisemblance reste difficile voir même impossible avec le nombre élevé de paramètres issus de distributions différentes à estimer. Cependant, en ce qui concerne l'ADM, il est nécessaire que les lois des K groupes soient toutes gaussiennes.
En outre, l'algorithme EM de Dempster et al. (1977) [14] est l'une des méthodes d'optimisation les plus utilisées pour des cas de mélange de densité. C'est un algorithme itératif permettant d'estimer les paramètres d'un modèle probabiliste dépendant des variables latentes (variables qui ne sont pas mesurées directement, mais supposées être à la base des variables observées). Il est souvent utilisé en apprentissage automatique et en imagerie médicale.

Au départ, la fonction de densité f est spécifiée et les valeurs de départ des K groupes des vecteurs $\pi(\pi_0)$ et $\theta(\theta_0)$ fixées. L'algorithme est itératif suivant deux étapes (Lebart et Morineau (1995) [?]) :

– Etape E (itération l) :

Il s'agit de l'étape d'estimation des probabilités conditionnelles $t_k^l(\mathrm{x}_i)$ d'appartenance de x_i au groupe k à la $l^{ième}$ itération connaissant les paramètres.

$$t_k^l(\mathrm{x}_i) = \frac{\pi_k^l f(\mathrm{x}_i, \theta_k^l)}{\sum_{j=1}^{K} \pi_j^l f(\mathrm{x}_i, \theta_j^l)}$$

– Etape M (itération $l+1$) :

On calcule $\pi_k^{l+1} = \frac{1}{n}\sum_{i=1}^{n} t_k^l(\mathrm{x}_i)$ puis on détermine une approximation de la vraisemblance des paramètres en maximisant la log-vraisemblance trouvée à l'étape E. Dans le cas de la loi gaussienne, d'après Lebart et al. (1995) [16], les estimations de μ_k et W_k à l'itération l+1 sont données par :

$$\mu_k^{l+1} = \frac{\sum_i t_k^l(\mathrm{x}_i)\mathrm{x}_i}{\sum_{t_k^l(\mathrm{x}_i)}}$$

L'estimation de la matrice des covariances :

$$W_k^{l+1} = \frac{\sum_i t_k^l(\mathrm{x}_i)(\mathrm{x}_i - \mu_k^l + 1)(\mathrm{x}_i - \mu_k^{l+1})}{\sum_i t_k^l(\mathrm{x}_i)}$$

L'itération est arrêtée lorsque la convergence de la vraisemblance est atteinte.

Avec le logiciel R, la fonction *mda* (*mixture discriminant analysis*) du package Mass permet d'effectuer une analyse discriminante par mélange gaussien. Il y a également le package *mclust* qui intègre la fonction *MclustDA*, plus adaptée aux modèles de mélange gaussiens permettant d'effectuer une analyse discriminante. La *MclustDA*, estime les paramètres du modèle par le biais de l'algorithme EM en donnant l'information sur le nombre d'itérations effectuées avant d'atteindre la convergence de la vraisemblance.

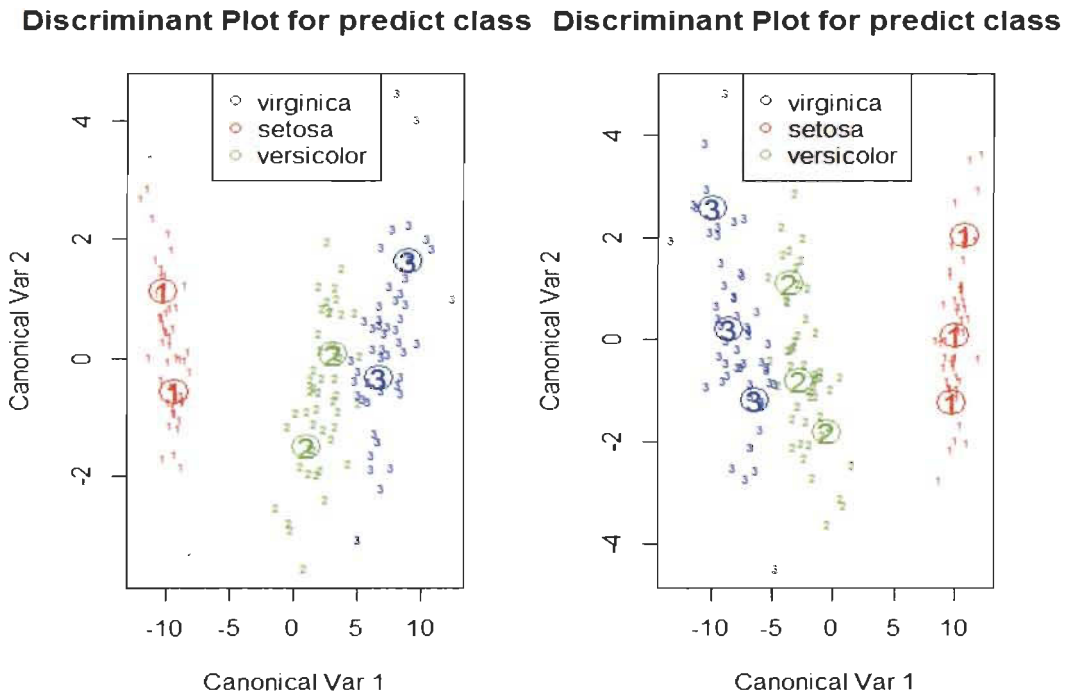
Cependant, *MclustDA* nécessite une durée d'exécution beaucoup plus importante comparée à celle de mda.

1.8 Application des fonctions mda et MclustDA aux Iris de Fisher

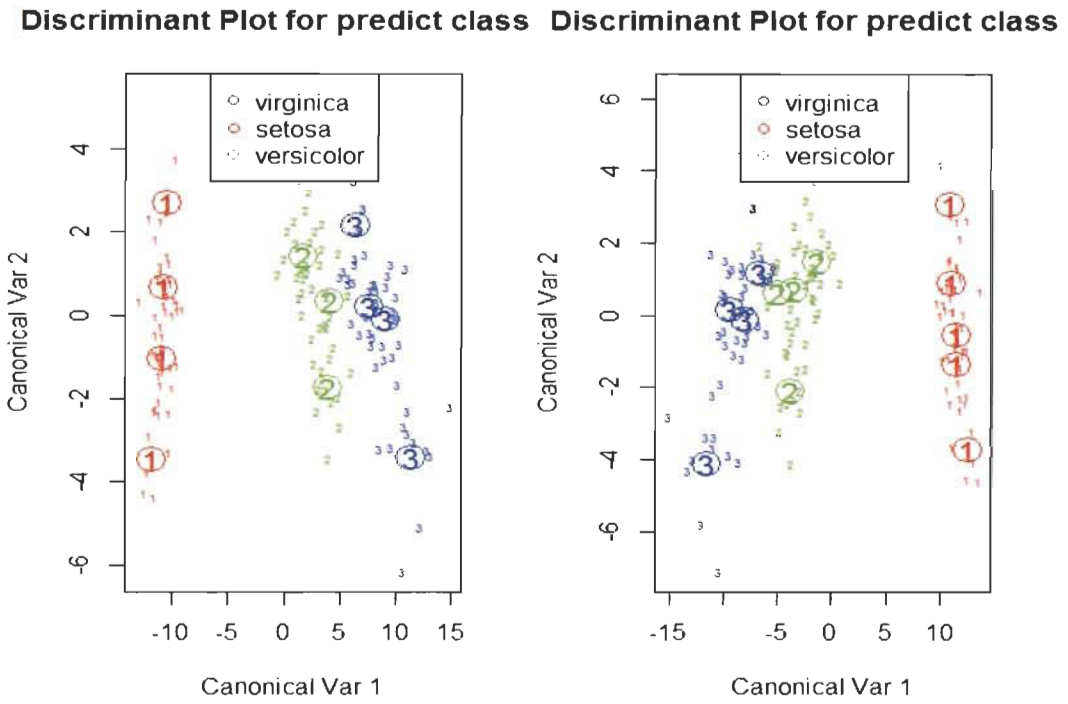
Avec la fonction *mda*, les paramètres sont estimés avec la méthode du maximum de vraisemblance tandis que MclustDA estime les paramètres μ_k et W_k avec l'algorithme EM.

Par défaut, *mda* subdivise chaque groupe en 3 sous-groupes. Avec les données Iris, le taux d'erreur de classification obtenu avec cette partition est de 2%. Les Graphiques 1.5 et 1.6 sont des représentations de la discrimination des groupes pour les différents cas de 2 à 5 sous-groupes. La représentation des situations de l'ADM pour ces différentes partitions montre que plus on augmente le nombre de sous-groupes, plus les éléments extrêmes se rapprochent à l'un des noyaux du groupe correspondant. Cela diminue le risque de les classer dans un autre groupe, d'où la baisse du taux de mauvaise classification (TMC).

Graphique 1.5 – Séparation des classes avec mda (1 et 2 sous groupes)

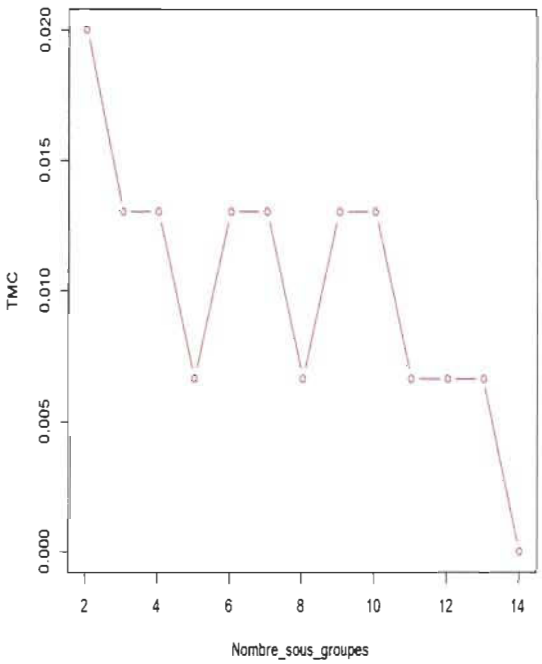


Graphique 1.6 Séparation des classes avec mda (4 et 5 sous groupes)

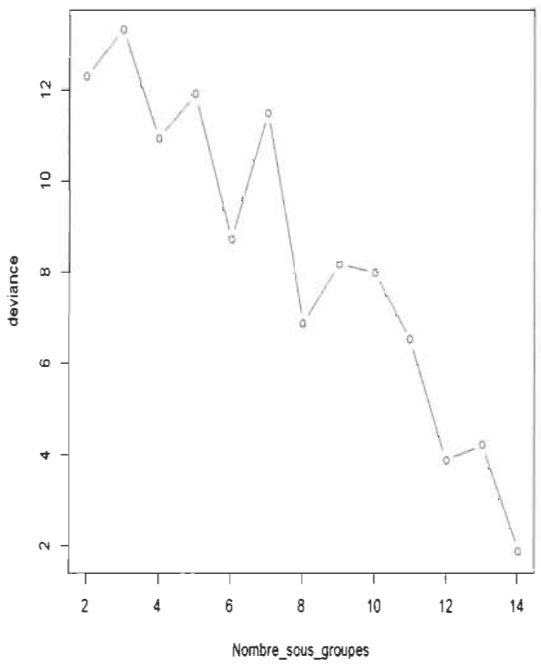


L'analyse du Graphique 1.7 montre qu'il est possible d'améliorer le taux de mauvaise classification en jouant sur la partition des groupes. En effet, de façon globale, plus le nombre de sous-groupes augmente plus le taux d'erreur diminue. Dans le cas d'une partition de 14 sous-groupes, les trois espèces sont parfaitement séparées par la fonction *mda* sans erreur de classification (TMC = 0%). Cependant, il est nécessaire de tenir compte de la maximisation de la vraisemblance qui a permis l'estimation des paramètres (μ_{kr}). Sinon, le taux de mauvaise classification peut être bien amélioré, mais avec des paramètres non optimaux. Le maximum de la vraisemblance (13.23) est atteint en considérant une partition de 3 sous-groupes par espèce avec un taux d'erreur de classification de 1.3% (voir Graphique 1.8).

Graphique 1.7 – Évolution du taux d'erreur de classification selon la partition des groupes.

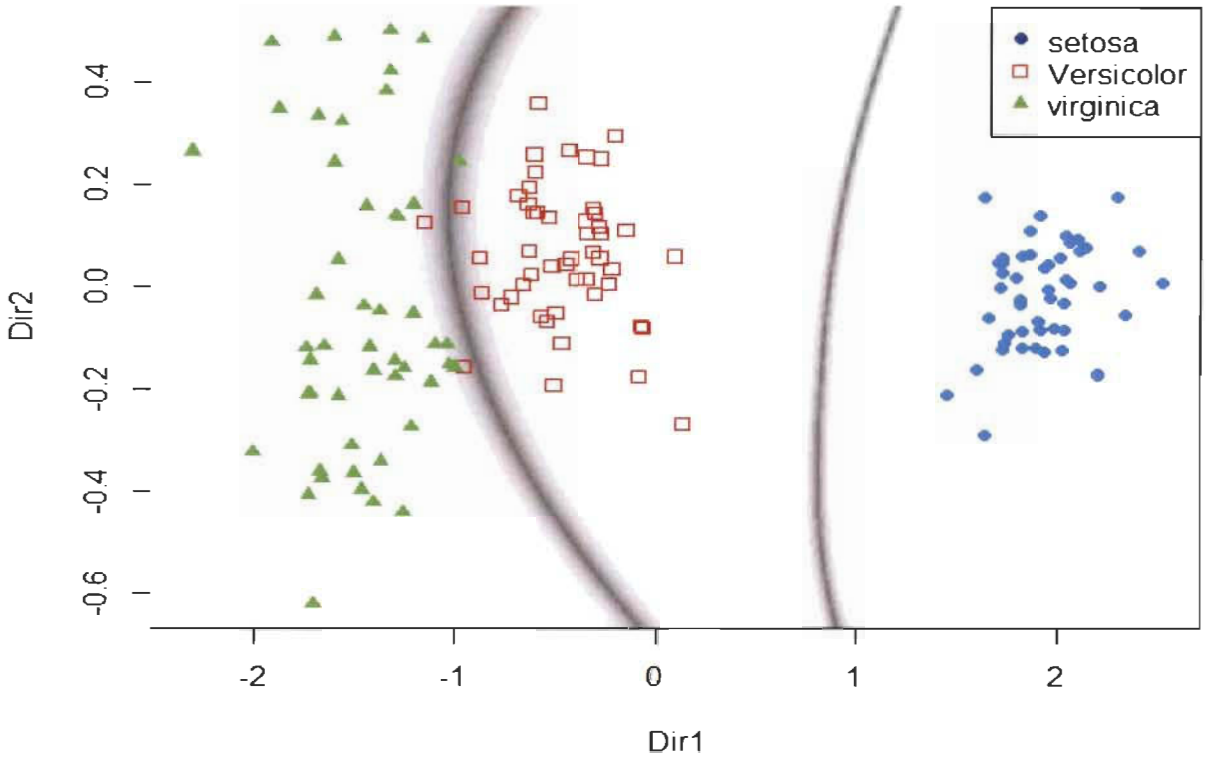


Graphique 1.8 – Évolution du la vraisemblance selon la partition des groupes.



En ce qui concerne *MclustDA*, la discrimination des classes reste relativement moins bonne que celle obtenue avec la fonction *mda*. En effet, avec un taux d'erreur de mauvaise classification de 2%. 2 espèces de type *versicolor* sont considérées comme *virginica* et une *virginica* retrouvée dans le groupe des *versicolor*. Ces résultats sont illustrés par le Graphique 1.9.

Graphique 1.9 – Séparation des classes avec la fonction MclustDA



Chapitre 2

Généralisation de l'analyse discriminante linéaire

Ce chapitre porte sur des variantes de l'ADL. Il s'agit en particulier de l'analyse discriminante flexible (ADF) et de l'analyse discriminante pénalisée (ADP). Elles sont bien développées par Hastie et al. (1995) [2] afin de pallier certaines limites de l'ADL. En ce qui concerne l'ADF, la méthode utilisé afin d'aboutir à des frontières non linéaires pour une meilleure séparation des groupes est la régression non paramétrique. Pour le cas de l'ADP, une matrice de pénalité est utilisée pour perturber les corrélations entre prédicteurs.

2.1 Quelques rappels sur la régression linéaire multiple

2.1.1 Description du modèle

La régression linéaire multiple est une généralisation de la régression linéaire simple. Elle cherche à répondre à la question : une série de p variables quantitatives X_j, $1 \leq j \leq p$ (explicatives dites exogènes) ont-elles une influence sur une variable quantitative Y (variable expliquée ou endogène) ? Dans ce cas, le modèle s'écrit :

$$Y = X\beta + \epsilon \; ; \tag{2.1}$$

$$y_i = \beta_0 + \mathrm{x}_{i1}\beta_1 + \mathrm{x}_{i2}\beta_2 + \cdots + \mathrm{x}_{ip}\beta_p + \epsilon_i \; ; \tag{2.2}$$

Avec X une matrice (n , p+1) de terme général x_{ij} avec $\mathrm{x}_{i0} = 1$, $\beta = (\beta_0, \cdots, \beta_p)'$ et $\epsilon = (\epsilon_1, \cdots, \epsilon_n)$

Hypothèses du modèle :

1. Les termes d'erreurs ϵ_i sont indépendantes et identiquement distribués i.e $E(\epsilon_i) = 0, \mathrm{Var}(\epsilon_i) = \sigma^2 I_n$; avec σ^2 la variance de la population.

2. L'erreur ϵ est indépendante de la distribution conjointe de $X_1, \cdots, X_p$:

$$E(Y/X_1, \cdots, X_P) = \beta_0 + \mathrm{x}_i^1\beta_1 + \mathrm{x}_i^2\beta_2 + \cdots + \mathrm{x}_i^p\beta_p; \tag{2.3}$$

$$\mathrm{Var}(Y/X_1, \cdots, X_p) = \sigma^2; \tag{2.4}$$

3. Les paramètres inconnus $\beta_0, \cdots, \beta_p$ du modèle sont supposés constants ;

4. Une quatrième hypothèse suppose une normalité de la loi des erreurs $\epsilon \sim N(0, \sigma^2 I_n)$.

2.1.2 Estimation des paramètres du modèle par les moindres carrés

Les valeurs des X_j étant connues, les paramètres β et σ sont estimés par la minimisation du critère des moindres carrés :

$$\sum_{i=1}^{n}(Y - X\beta)^2 = ||Y - X\beta||^2 \tag{2.5}$$

$$= (Y - X\beta)'(Y - X\beta) \tag{2.6}$$

$$= Y'Y - 2X'Y\beta' + \beta'X'X\beta \tag{2.7}$$

Par dérivée matricielle de l'équation (3.7) par rapport à β, on a :

$$X'Y - X'X\beta = 0 \tag{2.8}$$

En supposant la matrice $X'X$ inversible c'est à dire que X est de rang p+1 et que les X_j ne soient pas colinéaires. Alors l'estimation des β_j est donnée par :

$$\widehat{\beta} = (X'X)^{-1}X'Y \tag{2.9}$$

Ainsi les valeurs ajustées de Y sont données par l'expression :

$$\widehat{Y} = X\widehat{\beta} = X(X'X)^{-1}X'Y = HY \tag{2.10}$$

Avec $H = X(X'X)^{-1}X'$, appelée matrice chapeau ou «hat matrix» en anglais, la matrice de projection orthogonale dans R^n sur le sous-espace Vect(X) engendré par les X_j.

Les valeurs résiduelles sont obtenues par :

$$e = Y - \hat{Y} = (I_n - H)Y \tag{2.11}$$

e est la projection de Y sur le sous-espace engendré par $Vect(X)$ dans $\mathbb{R}''$.

2.2 L'analyse discriminante flexible (ADF)

Soit A la matrice (n, K) indicatrice des réponses ($A_{ik} = 1$ si l'individu $i \in G_k$ et 0 sinon). Plus explicitement, A représente une décomposition de la variable Y en k variables indicatrices. Ainsi, un nouvel individu peut être affecté au groupe G_k minimisant le critère des moindres carrées $\min||Y_k - \hat{Y}_k||^2$ après estimation des paramètres par une analyse canonique de X sur la matrice A. À la différence de la régression multivariée, l'analyse canonique effectue une régression d'une matrice de variables quantitatives X dites variables explicatives sur un ensemble de variables quantitatives Y à expliquées. Ainsi, elle peut être considérée comme une généralisation du modèle linéaire multivariée.

En régression linéaire multiple, un nouvel individu x_0 est affecté au groupe G_k ayant la plus grande valeur prédite. Selon Hastie et al. (1994) [1], cette règle d'affection bien que pouvant être assimilée à une affectation sur la base des probabilités a priori, a quelques limites, notamment, $\widehat{Y_k}$ n'est pas forcément dans [0,1], de même que la somme ne vaut pas toujours 1. L'affectation sur la base de la fonction *softmax* encore appelée la fonction exponentielle normalisée $\widehat{\sigma_k} = \frac{\exp(\widehat{Y_k})}{\sum_{j=1}^{K} \exp(\widehat{Y_j})}$, permet de régler ce problème de l'estimateur $\widehat{Y_k}$. Cependant, l'analyse discriminante flexible se base sur un algorithme substituant la régression linéaire multiple par une régression non paramétrique avec l'usage des scores optimaux pour passer des coefficients β aux scores optimaux $B\Theta$. Notons $\Theta : \mathcal{G} \longmapsto \mathbb{R}$ la fonction score de groupe. Les scores Θ_k sont obtenus en

résolvant le problème (Hastie et al. ,1994 [1]) :

$$\min||A\Theta - X\beta||^2; \tag{2.12}$$

Avec comme paramètres Θ et β ; Θ de moyenne nulle et de variance unitaire.
Les Θ_k sont mutuellement orthogonaux et normalisés pour éviter des solutions nulles triviales. Le choix optimal des coefficients β_k découle de la minimisation de la moyenne des carrés des résidus par la méthode des moindres carrés avec les Θ_k fixés.

$$MCR = \frac{1}{n}\sum_{k=1}^{K}\sum_{i=1}^{n}(\Theta_k(g_i) - \mathrm{x}_i'\beta_k)^2 \tag{2.13}$$

Avec g_i le centre de gravité du groupe G_i.
L'algorithme suivant montre comment l'analyse discriminante linéaire peut être réalisée en effectuant une régression linéaire multiple.
(1) Initialisation : Choisir la matrice indicatrice A_{nK} (voir Tableau 2.1) : $A_{ik} = 1$ si l'individu $i \in G_k$ et 0 sinon.

(2) Régression linéaire multiple : Obtenir $\widehat{A} = P_X A$ avec $P_{\mathrm{x}} = X(X'X)^{-1}X'$, l'opérateur de projection linéaire, et désignons par β, la matrice de coefficients : $\widehat{A} = X\beta$.

(3) Scores optimaux : Obtenir la matrice des vecteurs propres normalisés Θ de $A'P_X A$ telques : $\Theta' D_p \Theta = I_K$ avec $D_p = A'A/n$, la matrice de projection orthogonale sur R^P.

(4) Mise à jour des coefficients de la matrice β en tenant compte des scores optimaux : $\beta \longleftarrow \beta\Theta$.
La matrice des coefficients β de la régression est équivalente à une constante près, à la matrice des coefficients d'une ADL permettant de calculer les variables discriminantes. En effet, $U'\mathrm{x} = DX\beta_k$ où U est la matrice ayant en colonnes, les vecteurs résultant de la maximisation du problème de l'analyse discriminante linéaire de l'équation : $\max(a'Ba)$ sachant que $a'Wa = 1$ et D est une matrice diagonale composée

d'éléments $D^2_{kk} = \frac{1}{\alpha^2_k(1-\alpha^2_k)}$; avec α^2_k, la plus grande valeur propre obtenue à l'étape 3 de l'algorithme. D'où l'analyse discriminante flexible peut être conçue comme une analyse discriminante linéaire sur la matrice obtenue avec une régression non paramétrique linéaire.

Il existe d'autres formes de régression non paramétrique notamment la méthode (*MARS*) et *BRUTO* développées respectivement par Friedman (1991) [17] et par Hastie et al. (1990) [18] qui sont très utilisées en classification.

Tableau 2.1 – Exemple d'une matrice indicatrice A_{nK}

$$A_{nK} = \begin{array}{c} \\ i_1 \in G_2 \\ i_2 \in G_K \\ \cdots \\ \cdots \\ \cdots \\ \cdots \\ i_n \in G_k \end{array} \begin{array}{c} \begin{array}{cccccc} G_1 & G_2 & \cdots & G_k & \cdots & G_K \end{array} \\ \begin{pmatrix} 0 & 1 & \cdots & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 & \cdots & 1 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & 1 & \cdots & 0 \end{pmatrix} \end{array}$$

2.3 Description des méthodes MARS et BRUTO

2.3.1 Description de la méthode MARS

Introduite par Friedman (1991) [17], la méthode *MARS* (*Multivariate Adaptive Regression Spline*) est une forme de régression non paramétrique, extension du modèle linéaire. À la différence de la régression linéaire, elle ne fixe aucune hypothèse (linéarité ou non) de départ entre les variables explicatives et la variable dépendante. En outre, elle partitionne le domaine des variables explicatives en M sous-régions délimitées par des fonctions de base qui indiquent ses frontières. Chaque sous-région

est approximée par une fonction paramétrique simple (Hastie et al., 1990) [18]. La relation fonctionnelle donnant l'expression du modèle général est :

$$y = f(\mathrm{x}) = \beta_0 + \sum_{m=1}^{M} \beta_m \prod_{k=1}^{Lm} B(\mathrm{x}_{mk}) \tag{2.14}$$

Les termes β_m représentent les coefficients du modèle pour les variables explicatives x de la $m^{ième}$ sous-région et β_0, la constante. Les $B(\mathrm{x}_{mk})$ sont des fonctions de base qui s'appliquent aux variables explicatives x_k de la $m^{ième}$ sous-région. Lm représente le nombre de variables explicatives x_{mk} dans la sous-région m et M, le nombre de fonctions de base considérées ou de sous-régions.

$$B(\mathrm{x}_{mk}) = \max(0, \mathrm{x}_{mk} - c) \text{ ou } \max(0, c - \mathrm{x}_{mk}), \tag{2.15}$$

avec c une constante appelée nœud. Bref, il faut retenir que la méthode MARS procède par la détermination de l'ensemble des fonctions de base donnant un meilleur ajustement du modèle général par la maximisation du critère des moindres carrées (Hill et Lewicki, 2006) [19].

2.3.2 Description de la méthode BRUTO

Comme MARS, BRUTO (*Adaptive Additive Modeling*) est une méthode de régression non paramétrique faisant intervenir les fonctions de base pour approximer le modèle général. Elle est très efficace dans des situations où le nombre de variables explicatives est très élevé. Cependant, elle est seulement additive, donc plus restrictive que MARS. Plus précisément, elle ne fait pas intervenir la partition du domaine des variables explicatives en sous-régions. La fonction de régression est approximée par $f(\mathrm{x}) = \beta_0 + \sum_{j=1}^{p} \beta_j B(\mathrm{x}_j)$.

En outre, dans l'ajustement du modèle, BRUTO utilise une version pénalisée des

moindres carrés.

$$\sum_{i=1}^{n}(y_i - \sum_{j=1}^{p}\beta_j B(\mathrm{x}_{ij}))^2 + \sum_{j=1}^{p}\lambda_j \beta_j' \Omega_j \beta_j. \tag{2.16}$$

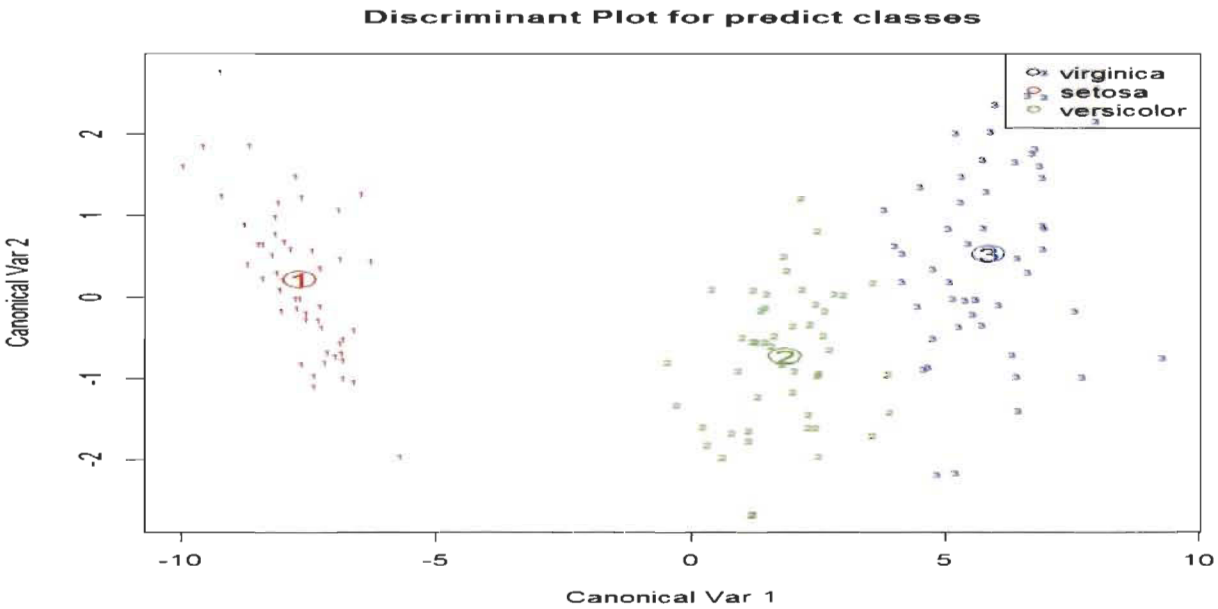
Avec Ω_j une matrice de pénalité et λ_j un vecteur de paramètre de lissage.

2.4 Application de la fonction fda aux données Iris de Fisher

Sous R, la fonction *fda* du package *Mass* permet d'effectuer une analyse discriminante flexible. Plusieurs méthodes peuvent s'appliquer avec la fonction *fda* notamment *polyreg*, *mars*, *bruto* et *gen.ridge*. Si aucune méthode n'est spécifiée, la méthode *polyreg* correspondant à une régression linéaire multiple est utilisée par défaut. La méthode *gen.ridge* est utilisée pour le cas d'une analyse discriminante pénalisée (voir section 2.5).

Chacune des fonctions de scores (probabilités a posteriori) à 5 degrés de liberté, la première explique 99,12% de l'inertie totale. le Graphique 2.1 représente la discrimination des trois types d'espèces des Iris sur la base de la fonction lda.

Graphique 2.1 – Discrimination des classes avec la fonction fda



Avec un taux de mauvaise classification de 2%, le modèle ADF est plus performant que l'ADL pour les Iris de Fisher(voir Tableau 2.2).

Tableau 2.2 – Matrice de confusion de l'ADF des Iris de Fisher

	setosa	versicolor	virginica
setosa	50	0	0
versicolor	0	48	1
virginica	0	2	49
Taux de mauvaise classification		2%	

2.5 L'analyse discriminante pénalisée (ADP)

Parmi les limites de l'ADL, on rappelle sa non robustesse pour le cas de variables explicatives fortement corrélées entre elles et si la distribution des variables explicatives n'est pas proche de la normalité. Ces situations s'opèrent souvent dans le cas de la reconnaissance d'images et de caractères manuscrits (Hastie et al., 1995) [2]. Pour pallier à cette situation, la méthode qui est souvent utilisée en analyse discriminante consiste à une pénalisation de la matrice de variance-covariances intra-groupe W, d'où le nom d'analyse discriminante pénalisée (ADP). De façon plus explicite, l'ADP fonctionne en remplaçant la matrice de variance intra-groupe W par une version plus régularisée $W + \lambda\Omega$ où Ω est une matrice symétrique (p x p) définie positive appelée matrice de pénalité permettant de perturber les corrélations entre variables explicatives X_j et λ, le coefficient de pénalité permettant de régler l'influence de Ω sur sur la matrice de variances-covariance.

Le problème à optimiser est le même que celui de l'ADL, on cherche à trouver des combinaisons linéaires des X_j maximisant la matrice de variance-covariances intergroupe B, sous la contrainte $a'(W + \lambda\Omega)a = 1$. Le problème d'optimisation s'écrit :

$$\max(a'Ba) \text{ sous la contrainte } a'(W + \lambda\Omega)a = 1 \tag{2.17}$$

Dans les travaux de Hastie et al. (1995) [2], la pénalisation de la matrice W a été introduite dans un cadre de traitement d'images. Ainsi, une ADL est appliquée avec comme métrique la distance de Mahalanobis :

$$D^2_{ADP} = (x - \bar{x}_k)'(W + \lambda\Omega)^{-1}(x - \bar{x}_k) \tag{2.18}$$

Avec la méthode des scores optimaux, nous considérons toujours la variable expliquée Y des réponses et un score Θ_k est attribué à chaque groupe G_k. Ainsi on effectuera une régression linéaire de la matrice X sur le vecteur $Y\Theta$.

Le critère d'optimisation des moindres carrés de ce cadre de l'ADP s'écrit :

$$\text{MCR} = \frac{1}{n}\sum_{k=1}^{K}\left[\sum_{i=1}^{n}(\Theta_k(g_i) - \eta_k(\mathrm{x}_i))^2 + \beta_k'\Omega\beta_k\right] \tag{2.19}$$

$$= \frac{1}{n}\Big[||Y\Theta - X\beta||^2 + \beta'\Omega\beta\Big], \tag{2.20}$$

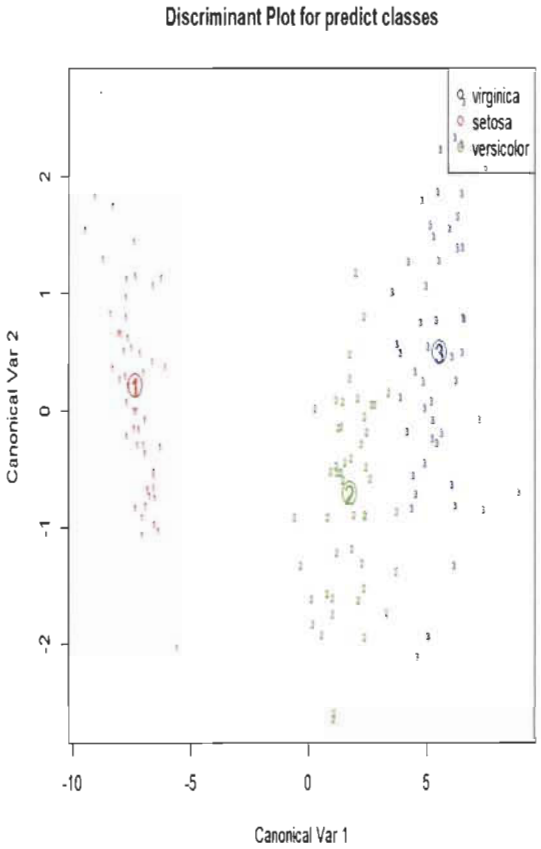
à minimiser sous la contrainte, $\frac{1}{n}||Y\Theta||^2 = 1$, η_k représente le modèle utilisé pour l'approximation de la fonction de régression, $\eta_k(X) = X\beta_k$ et g_i, le centre de gravité du groupe G_i de l'élément i.

Avec la fonction *fda* de R, il suffit d'utiliser la méthode *gen.ridge* pour effectuer une analyse discriminante pénalisée.

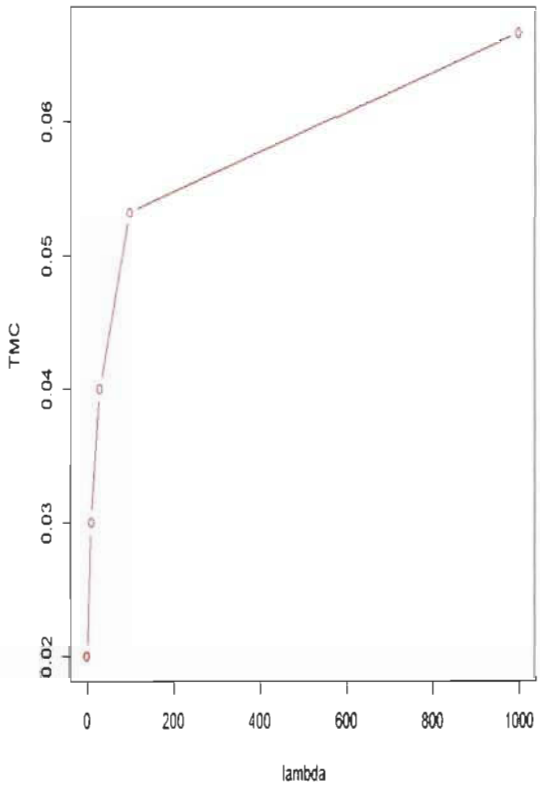
2.6 Application du modèle *gen.ridge* aux Iris de Fisher

La fonction fda appliquée à la méthode *gen.ridge* fournit les résultats d'une analyse discriminante pénalisée. Avec les Iris de Fisher, les résultats obtenus (graphique 2.2) sont proches de ceux de l'ADF (*polyreg*). Cela peut s'expliquer par l'idée selon laquelle les variables explicatives ne sont pas fortement corrélées entre elles à part *Petal.Length* et *Petal.Width*. Les trois types d'espèces sont séparés avec un taux d'erreur de 2%. Le coefficient de pénalité λ a une grande influence sur l'amélioration du taux d'erreur, mais également sur le nombre de degré de liberté (paramètres estimés). Il apparait également que de grandes valeurs de λ augmentent fortement le risque de mauvaise classification (voir Graphique 2.3). Pourvu que les variables explicatives ne soient pas fortement corrélées entre elles, un certain niveau de perturbation du lien existant entre les variables explicatives peut influencer négativement la discrimination. D'ailleurs, il n'est pas nécessaire de perturber la corrélation. D'où la non pertinence de l'analyse discriminante pénalisée pour une discrimination des Iris de Fisher.

Graphique 2.2 – Discrimination des groupes avec le modèle gen.ridge



Graphique 2.3 - Évolution du taux d'erreur de classification selon λ.



Chapitre 3

Le perceptron multicouche (PMC)

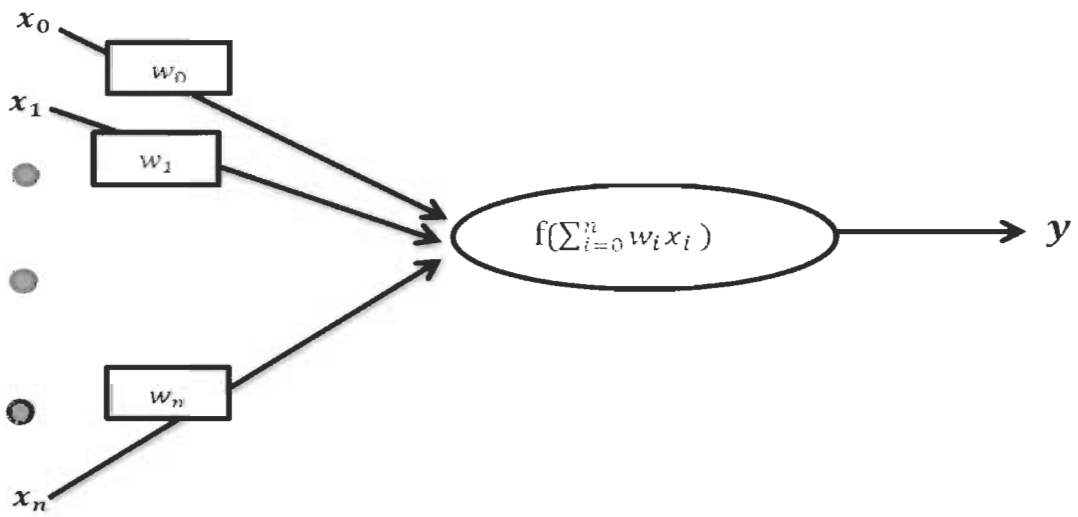
3.1 Historique du neurone artificiel

Le réseau de neurone artificiel, dont les travaux sont réalisés pour la première fois par les neurologues McCulloch et Pitts (1943) [3] dans la modélisation du cerveau humain, s'est inspiré du réseau de neurone physiologique (fonctionnement du cerveau humain). Ce concept fut développé par la suite par le psychologue américain Rosenblatt (1958) [4] dans un modèle de classifieur binaire afin d'étudier le comportement de l'oeil à capter et à reconnaitre une image. Ce dernier, connu sous le nom de perceptron simple permet de modéliser des échantillons linéairement séparables. Cependant, le plus souvent on ne se retrouve pas dans ces situations, d'où l'idée de penser à une combinaison de séparateurs linéaires afin d'obtenir un séparateur non-linéaire (Rumelhart et al., 1986) [20]. A l'instar des méthodes de régression (linéaire, logistique etc.), le perceptron multicouche est l'un des outils les plus utilisés en classification supervisée, surtout dans la modélisation de grandes bases de données.

3.2 Fonctionnement du perceptron multicouche

Les méthodes neuronales en général et le perceptron multicouche en particulier constituent des outils très bénéfiques à l'ère de l'intelligence artificielle en raison du coût faible en terme de temps de modélisation. Le PMC est un outil d'apprentissage supervisé permettant de modéliser et de prédire les classes d'éléments à partir de données disponibles appelées inputs. Il s'agit d'un modèle de réseau de neurones à une couche d'entrée, des couches cachées et d'une couche de sortie. Chaque couche est constituée de neurones permettant l'interconnexion entre deux couches consécutives. Comme le modèle d'analyse discriminante prédictif, le PMC permet de prédire une variable catégorielle explicative Y à K modalités à partir de n variables quantitatives prédictrices ($x_1, \cdots, x_n$) constituant les entrées. Les x_i sont pondérées par des coefficients de pondération appelés poids ($w_1, w_2, \cdots, w_n$).

Graphique 3.1 – Architecture du Modèle de Rosenblatt



Le neurone est un séparateur linéaire donnant une sortie y avec une fonction

d'activation f telle que :

$$y = f(a) = f(\sum_{i=1}^{n} \mathrm{w}_i \mathrm{x}_i +)\grave{a}\grave{a}textw_0) \tag{3.1}$$

Le Graphique 3.1 représente l'architecture du perceptron de Rosenblatt. Un poids w_0 appelé biais est appliqué à une entrée fictive $\mathrm{x}_0 = 1$. Plusieurs fonctions d'activation sont souvent utilisées notamment la fonction identité, la fonction sigmoïde, la fonction de marche, La fonction unité de rectification linéaire (ReLU) plus adaptée au modèle de Deep learning (voir Graphique 3.2 à 3.5).

Elles sont définies comme suites :

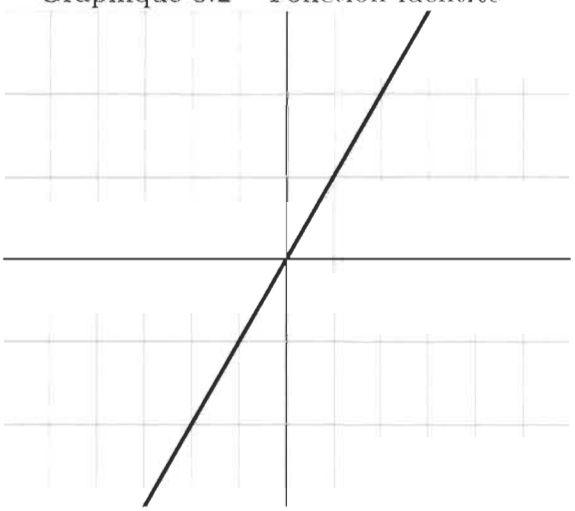
La fonction identité : $f(\mathrm{x}) = \mathrm{x}$ (Graphique 3.2) ;

La finction sigmoïde : $f(\mathrm{x}) = \frac{1}{1+\exp(-\mathrm{x})}$;

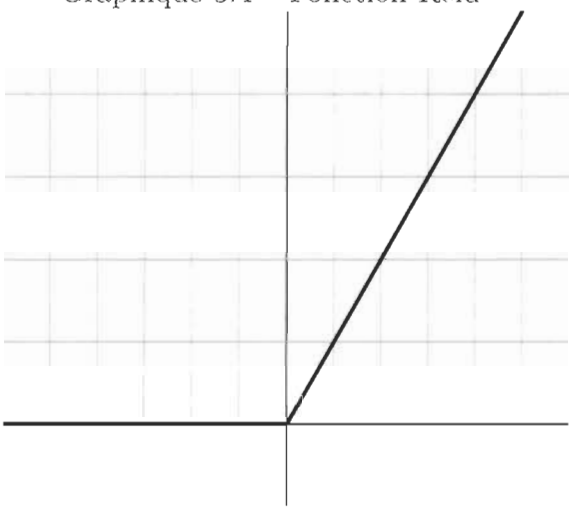
La fonction Relu : $f(\mathrm{x}) = 0$ pour $x < 0$; $f(\mathrm{x}) = x$ pour $x \geq 0$

La fonction de marche : $f(\mathrm{x}) = 0$ pour $x < 0$; $f(\mathrm{x}) = 1$ pour $\mathrm{x} \geq 0$.

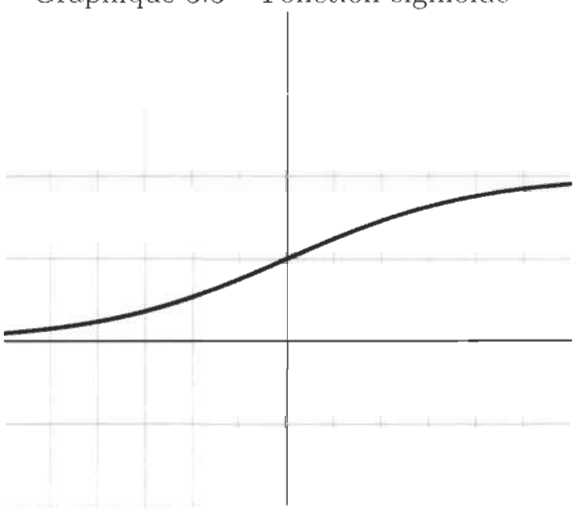
Graphique 3.2 - Fonction identité



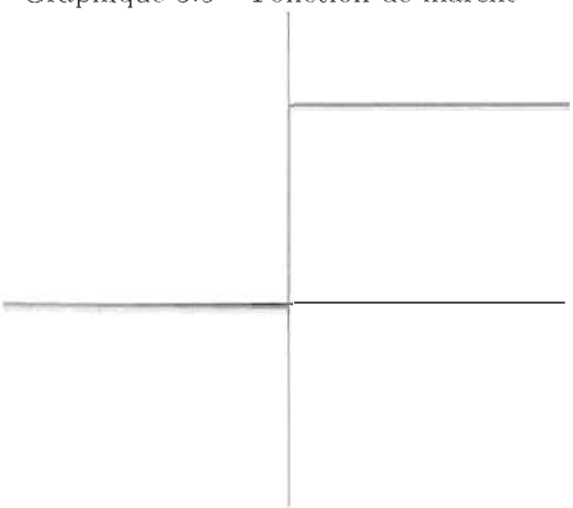
Graphique 3.4 - Fonction Relu



Graphique 3.3 - Fonction sigmoïde



Graphique 3.5 - Fonction de marche

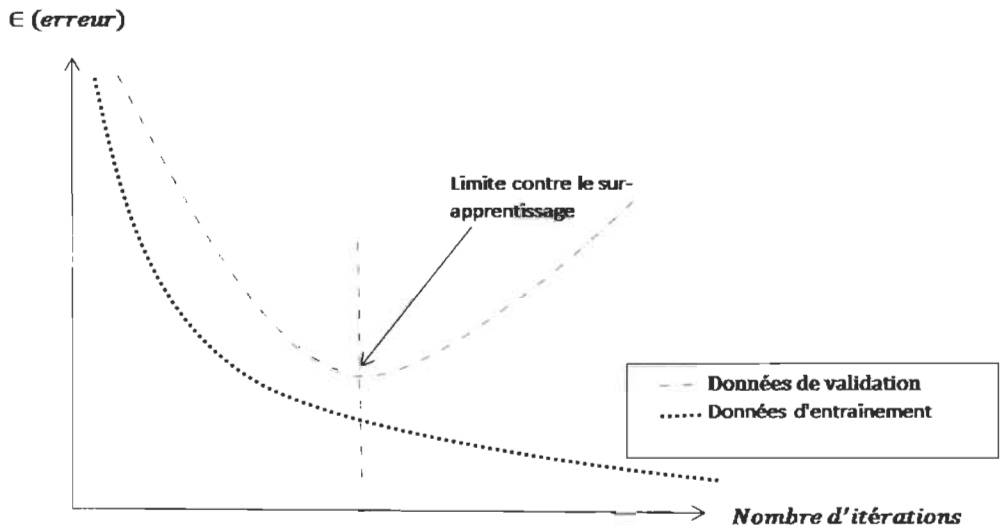


3.2.1 Modèle d'apprentissage du perceptron multicouche

Le jeu de données est divisé en deux ou trois sous-échantillons notamment en données d'entrainement, de validation et celles qui serviront de test. Les données d'entrainement permettent le paramétrage des poids w_i, i = 0,1, $\cdots$,n, celles de

validation permettent de contourner le sur-apprentissage. Il s'agit de la situation à laquelle, le modèle s'adapte aux données d'entrainement et reste sensible à une légère variation de ces dernières. Un modèle qui surapprend ne prédit pas bien les classes de nouvelles données non entrainées. Avec une validation croisée, le sur-apprentissage est noté dès que l'erreur des données de validation commence à croître (voir Graphique 3.6). En ce qui concerne les données test, elles permettent d'évaluer la performance du modèle mis en place avec les données d'entrainement.

Graphique 3.6 – Schéma de variation de l'erreur des données d'entrainement et de validation



A la différence du perceptron simple, le PMC se définit par :

— Son architecture composée de plusieurs couches cachées (voir Graphique 3.7) ;

— Les fonctions d'activation utilisées s'appliquant à chaque neurone afin de déterminer la sortie y ;

— La correction des erreurs par rétropropagation du gradient en ajustant les poids

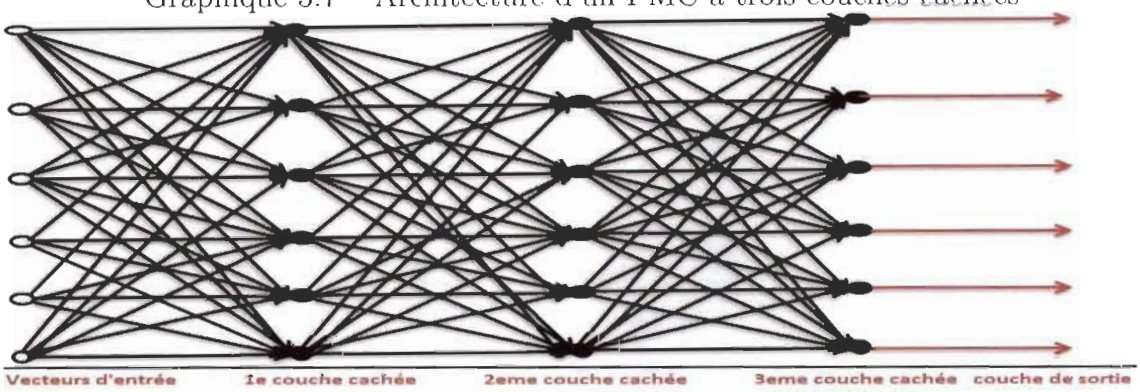
w_i.

La sortie y_j du neurone j de la couche de sortie est donnée par :

$$y_j = (f(v_j)) = f(\sum_{i=0}^{n} w_j^i x_i) \tag{3.2}$$

$v_j = \sum_{i=0}^{n} w_j^i x_i$ est la somme pondérée des entrée du neurone j.
Si la fonction f est réduite à la fonction identité ($f(x) = x$), on se retrouve avec une analyse discriminante (Lebart et al., 1995 [16]).

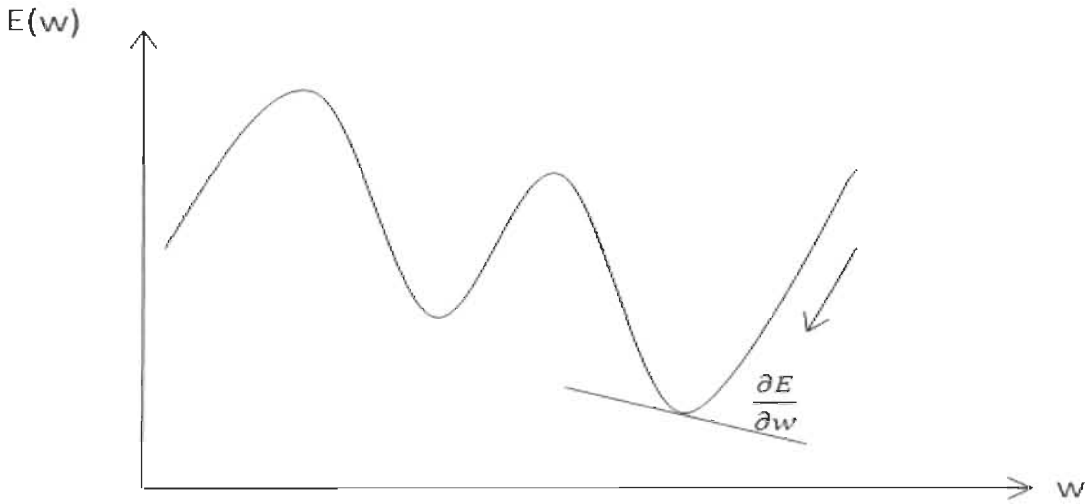
Graphique 3.7 - Architecture d'un PMC à trois couches cachées



3.2.2 Algorithme de rétropropagation

La rétropropagation est l'un des algorithmes les plus puissants dans la correction d'erreurs dans un modèle de PMC. Elle cherche à perfectionner le modèle par la mise à jour des poids ($w_1, \cdots, w_n$) afin de minimiser l'erreur quadratique moyenne E (Graphique 3.8). Il s'agit de la moyenne de la somme des carrés des écarts e_j entre la sortie observée et la sortie désirée au niveau de la couche de sortie du réseau.

Graphique 3.8 – La descente du gradient suivant W



— **Mise à jour des poids au niveau de la couche de sortie**

La rétropropagation consiste à la mise à jour des poids des neurones à partir de la couche de sortir vers la couche d'entrée afin de minimiser l'écart entre la sortie observée (y_j) et la sortie désirée ou la valeur réelle de y.

$$E = \frac{1}{2}\sum_{j=1}^{J}(e_j)^2 = \frac{1}{2}\sum_{j=1}^{J}(\mathrm{y}_j^d - \mathrm{y}_j)^2 \tag{3.3}$$

y_j : la sortie observée ;

y_j^d : la sortie attendue ;

e_j l'écart entre la sortie observée et la sortie désirée de la couche de sortie ;

J représente le nombre de neurones dans la couche de sortie.

La minimisation de E est basée sur la résolution de l'équation :

$$\frac{\partial E}{\partial \mathrm{w}_j^i} = 0 \tag{3.4}$$

w_j^i représente le poids de connexion entre le neurone i de la couche précédente et le

neurone j la couche en cours. Par la règle d'associativité des dérivées partielles, on a :

$$\frac{\partial \mathrm{f(y)}}{\partial \mathrm{x}} = \frac{\partial \mathrm{f(y)}}{\partial \mathrm{y}} \frac{\partial \mathrm{y}}{\partial \mathrm{x}};$$

l'équation (3.4) devient :

$$\frac{\partial E}{\partial \mathrm{w}_j^i} = \frac{\partial E}{\partial \mathrm{e}_j} \frac{\partial \mathrm{e}_j}{\partial \mathrm{y}_j} \frac{\partial \mathrm{y}_j}{\partial \mathrm{v}_j} \frac{\partial \mathrm{v}_j}{\partial \mathrm{w}_j^i}. \tag{3.5}$$

$$\frac{\partial E}{\partial \mathrm{e}_j} = \frac{\partial \left[\frac{1}{2}\sum_j e_j^2\right]}{\partial \mathrm{e}_j} = \mathrm{e}_j$$

$$\frac{\partial \mathrm{e}_j}{\partial \mathrm{y}_j} = \frac{\partial \left[\sum_j (\mathrm{y}_j^d - \mathrm{y}_j)\right]}{\partial \mathrm{y}_j} = -1$$

En considérant une fonction d'activation sigmoïde, $\mathrm{f}(\mathrm{v}_j) = \frac{1}{1+\exp(-\mathrm{v}_j)}$ et en appliquant cette fonction à l'expression $\frac{\partial \mathrm{y}_j}{\partial \mathrm{v}_j}$, on a :

$$\begin{aligned}
\frac{\partial \mathrm{y}_j}{\partial \mathrm{v}_j} &= \frac{\partial \mathrm{f}(\mathrm{v}_j)}{\partial \mathrm{v}_j} \\
&= \frac{\partial \left[\frac{1}{1+\exp(-\mathrm{v}_j)}\right]}{\partial \mathrm{v}_j} \\
&= \frac{\exp(-\mathrm{v}_j)}{[1+\exp(-\mathrm{v}_j)]^2} \\
&= \frac{1}{(1+\exp(-\mathrm{v}_j))} \frac{\exp(-\mathrm{v}_j)}{(1+\exp(-\mathrm{v}_j))} \\
&= \mathrm{y}_j \left[\frac{1+\exp(-\mathrm{v}_j)-1}{1+\exp(-\mathrm{v}_j)}\right] \\
&= \mathrm{y}_j(1-\mathrm{y}_j)
\end{aligned}$$

La dernière dérivée partielle de l'équation (3.5) donne :

$$\frac{\partial \mathrm{v}_j}{\partial \mathrm{w}_j^i} = \frac{\partial \sum_{k=1}^{n} \mathrm{w}_j^k \mathrm{x}_k}{\partial \mathrm{w}_j^i}$$
$$= \mathrm{x}_i$$

Ainsi, on obtient :

$$\frac{\partial E}{\partial \mathrm{w}_j^i} = -\mathrm{e}_j \mathrm{y}_j(1-\mathrm{y}_j)\mathrm{x}_i = -(\mathrm{y}_j^d - \mathrm{y}_j)\mathrm{y}_j(1-\mathrm{y}_j)\mathrm{x}_i.$$

La mise à jour des poids s'obtient avec la règle du delta :

$$\Delta_i^j = \eta \frac{\partial \mathrm{E}}{\partial \mathrm{w}_j^i}.$$

Avec η le taux d'apprentissage compris entre 0 et 1. L'équation de modification des poids par la méthode de rétropropagation de l'erreur est donnée par :

$$\mathrm{w}_j^i(t+1) = \mathrm{w}_j^i(t) + \Delta_j^i(t) = \mathrm{w}_{ij}(t) - \eta(\mathrm{y}_j^d - \mathrm{y}_j)\mathrm{y}_j(1-\mathrm{y}_j)\mathrm{x}_i \tag{3.6}$$

— **Algorithme**

1. Phase d'initialisation : Choix au hasard de la matrice des poids $\mathrm{w} = (\mathrm{w}_0, \cdots, \mathrm{w}_n)$;
2. Présenter les données d'entrainement au réseau par propagation pour obtenir les sorties ;
3. Calcul de l'erreur au niveau de chaque réseau de la couche de sortie $\mathrm{e}_j = (\mathrm{y}_j^d - \mathrm{y}_j)$;
4. Mise à jour des poids au niveau de la couche de sortie par la règle du delta $\mathrm{w}_j^i(t+1) = \mathrm{w}_j^i(t) + \Delta_i^j(t)$;
5. Propagation de l'erreur sur les poids de la couche de sortie vers les couches cachées ;

6. Itération de la mise à jour des poids jusqu'à convergence de l'erreur e_j vers un minimum global.

3.3 Application d'un PMC aux Iris de Fisher

Avec la fonction *nnet*, un réseau de perceptron multicouche (PMC) est entrainé avec 80% (données d'entrainement) de l'échantillon initial des Iris de Fisher. Les 20% ont servi d'échantillon de test afin de mesurer la performance du modèle sur la base du taux de mauvaise classification (TMC). Après une série de simulation en faisant varier le nombre de couches cachées, le meilleur TMC (2.2%) est obtenu avec un perceptron à une couche cachée à 2 neurones avec une fonction sigmoïde comme fonction d'activation. Le Tableau 3.1 représente la matrice de confusion de la prédiction de la base test.

Tableau 3.1 – Matrice de confusion de la prédiction des données test avec le PMC

	setosa	versicolor	virginica
setosa	15	0	0
versicolor	0	17	0
virginica	0	1	12
Taux de mauvaise classification		2.2%	

Chapitre 4

Application : Analyse comparative des résultats d'AD et du PMC

4.1 Définitions de quelques indicateurs de mesures de performances de modèles

4.1.1 La matrice de confusion

La matrice de confusion est un indicateur permettant de mesurer la qualité d'un modèle de prédiction. En colonne, il y a le nombre d'occurrences de chaque modalité de la variable prédite ($\hat{Y}$). Chaque ligne représente les valeurs réelles de chaque modalité de la variable expliquée Y (binaire $Y = 0/Y = 1$) voir tableau 4.1.

Tableau 4.1 – Exemple de matrice de confusion

	Valeurs réelles	
Valeurs prédites	Vrais Positifs (VP)	Faux Négatifs (FN)
	Faux Positifs (FP)	Vrais Négatifs (VN)

Les occurrences VP, FP, FN et VN sont définies par :

VP : nombre de vrais positifs, il représente le nombre d'observations appartenant à la *classe 0* bien classées par le modèle ;

FP : nombre de faux positifs, il représente le nombre d'observations de la *classe 0* mal classées par le modèle ;

FN : nombre de faux négatifs, il s'agit du nombre d'observations de la *classe 1* mal prédites par le modèle ;

VN : nombre de vrais négatifs, il représente le nombre d'entrées de la *classe 1* bien prédites.

4.1.2 Les taux de bonne classification (TBC) et de mauvaise classification (TMC)

Les TBC et TMC sont deux indicateurs souvent utilisés en classification supervisée afin de comparer les performances de modèles et éventuellement dans l'appréciation de la qualité d'un modèle. A des fins de prédiction, il est souvent préférable de prédire avec le modèle ayant le TBC le plus élevé ou avec celui ayant le plus faible TMC. Pour le cas d'une variable Y binaire, ces deux indicateurs se calculent comme suit :

$$TBC = \frac{VP + VN}{VP + VN + FP + FN} \text{x} 100 \tag{4.1}$$

$$TMC = \frac{FP + FN}{VP + VN + FP + FN} \text{x} 100 \tag{4.2}$$

Le nombre total d'entrées bien classées et le total d'entrées mal classées sont respectivement : $(VP + VN)$ et $(FP + FN)$.

4.1.3 La courbe ROC

La courbe ROC (de l'anglais *Receiver Operating Characteristic*) est un indicateur de performance de modèle de prédiction sur la base de l'aire sous la courbe (AUC). Elle représente le taux de vrais positifs en fonction du taux de faux positifs. La construction de la courbe est basée sur le taux de VP, du taux de FP et d'une valeur seuil s, définissant la règle de classification. En considérant la classe **1** comme la classe d'intérêt (généralement la classe ayant la plus grande proportion d'observation) et un seuil s fixé, d'après Delacour et al. 2005 [21], la règle de décision est donnée par la relation :

$$\widehat{Y} = \begin{cases} 1 \text{ si } S(\mathrm{x}_i) > s \\ 0 \text{ sinon} \end{cases}$$

Où S est la fonction de score (probabilités a posteriori). Les couples (VP, VF) formant la courbe sont déterminés en faisant varier la valeur du seuil s. Plus précisément à chaque valeur du seuil s, il y a un couple (VP, VF) correspondant.

Plus l'AUC est grande, plus le modèle est meilleur. Ainsi, la courbe ROC et l'AUC peuvent servir d'indicateurs de comparaison de modèles. Selon la situation des données à prédire, les cas suivants peuvent être envisagés.

— La courbe ROC est confondue avec la diagonale (AUC = 0.5), le modèle est considéré comme aléatoire.

— La courbe s'approche du coin supérieur gauche (AUC proche de 1), cas d'un meilleur modèle du fait que de fortes valeurs du taux de vrais positifs sont obtenues pour de faibles taux de faux positifs.

— Une AUC égale à 1 signifierait que le modèle permet de discriminer à 100% les deux groupes, d'où un modèle de prédiction parfait.

Dans des situations où les classes sont déséquilibrées (effectifs différents), la courbe ROC et l'AUC sont préférables aux indicateurs de performance tels

que le TBC et le TMC (Prevost et al., 1998)[22] . En effet, ces derniers sont trop sensibles à la disproportion des groupes. Ainsi, la courbe de ROC serait plus fiable comme indicatrice de performance ou de comparaison de modèles du fait qu'elle est construite sur la base de paramètres normalisés par l'effectif des classes (VP, FP) (Rioult, 2011) [23].

4.2 Application des méthodes d'analyse discriminante et du PMC aux données RFMT

4.2.1 Présentation des données RFMT et statistiques descriptives

Le premier jeu de données exploité dans cette partie provient du centre de services de transfusion sanguine (Yeh et al., 2009) [24]. Il est constitué de cinq (5) variables dont quatre (4) explicatives et une qualitative à expliquer et de 748 observations. Afin d'alimenter sa banque de sang et de rendre plus performant son modèle RFM [1], le centre passe tous les trois mois dans une des universités de Hsin-Chu afin de collecter du sang. Ainsi, 748 donneurs au hasard sont tirés de la base de données pour la mise en place du nouveau modèle.

Dans le cadre de ce projet, ces données sont utilisées afin de prédire si un donneur i pris au hasard dans la base de données a fait un don durant l'opération du 03 Mars 2007 ou non.

Les quatre(4) variables explicatives sont décrites comme suit :

R : Référence - mois depuis le dernier don ;

F : Fréquence - nombre total de dons ;

M : Quantité totale de sang donné en centimètre cube (cc) ;

T : Temps - mois depuis le premier don ;

Class : la variable catégorielle indiquant si le donneur i a fait un don durant

1. Il s'agit d'un modèle utilisant la séquence de Bernoulli en théorie des probabilités qui permet de découvrir les connaissances nécessaires à la sélection de cibles pour le marketing direct à partir d'une base de données [24].

l'opération du 03 Mars 2007 (*Class = 1*) ou non (*Class = 0*).

Le Tableau 4.2 donne des statistiques descriptives pour chacune des variables

Tableau 4.2 - Statistiques descriptives des donneurs de sang

	R	F	M	T	Class
Min.	0.0	1.0	250.0	2.0	0.0
1st Qu.	2.8	2.0	500.0	16.0	0.0
Median	7.0	4.0	1000.0	28.0	0.0
Mean	9.5	5.5	1379.0	34.3	0.2
3rd Q.	14.0	7.0	1750.0	50.0	0.0
Max.	74.0	50.0	12500.0	98.0	1.0

Ce dernier renseigne sur quelques caractéristiques de tendance centrale des variables notamment le min et le max, la médiane, la moyenne, le premier et troisième quartiles. Il apparaît également de façon implicite que les variables Fréquence (F) et Monétaire (M) sont colinéaires. Cela se visualise également au niveau du Tableau 4.3 donnant la matrice de corrélation entre variables. Les variables Fréquence (F) et Monétaire (M) jouent le même rôle dans tout

Tableau 4.3 - Matrice de corrélation des variables explicatives

	R	F	M	T
R	1	-0.183	-0.183	0.161
F	-0.183	1	1	0.635
M	-0.183	1	1	0.635
T	0.161	0.635	0.635	1

modèle. De ce fait, seule la variable Fréquence sera considérée dans la suite.

– Homoscédasticité et normalité des données

Les matrices de variance-covariances (voir Tableaux 4.4 et 4.5) aux cas où *Class* = 1 et *Class* = 0 ne sont pas identiques. Ainsi, l'hypothèse d'homoscédasticité n'est pas vérifiée. En ce qui concerne la normalité, nous nous basons sur le test de Shapiro-Wilks. Il teste l'hypothèse nulle selon laquelle, un échantillon

$(x_1, \cdots, x_n)$ suit une loi normale. Pour chacune des variables R, F et T, la *p-value* (2.2e-16) est hautement significative donc l'hypothèse nulle est rejetée i.e les données n'ont pas une distribution normale.

Tableau 4.4 – Matrice de variance-covariances de la classe 1

	R	F	T
R	26.735	-7.941	8.626
F	-7.941	64.592	140.576
T	8.626	140.576	558.350

Tableau 4.5 – Matrice de variance-covariances de la classe 0

	R	F	T
R	70.981	-5.073	36.329
F	-5.073	22.532	76.388
T	36.329	76.388	605.425

4.2.2 Comparaison des résultats des différents modèles sur la base du taux de bonne classification

Les résultats présentés dans le Tableau 4.6 représentent les taux de bonne classification des différents modèles d'analyse discriminante et du perceptron multicouche. Les données initiales sont partitionnées en données d'entrainement (80%) et de test (20%). Sur la base du taux de bonne classification (TBC), les modèles de prédiction les plus performants sont l'ADQ, l'ADF(BRUTO) et l'ADF(MARS) avec un TBC de 81% sur les données Test chacun. L'ADL fournit le modèle le moins performant avec un TBC relativement faible (65% des données d'entrainement et 67% des données de test). Ces résultats de l'ADL sont liés au fait que les hypothèses de normalité et d'homoscédasticité ne sont pas vérifiées. L'ADP n'a pas fourni un meilleur modèle avec 77% de TBC des données d'entrainement et 79% des données de test. Comme c'est bien décrit au niveau de la section 3.5, cette dernière méthode est recourue dans des situations où les variables explicatives sont fortement corrélées afin de perturber la

corrélation (tel n'est pas le cas pour les données utilisées). En ce qui concerne l'ADM, le maximum de la vraisemblance (700.92) est atteint avec une partition de 4 sous-groupes et un TBC de 78% des données d'entrainement et des données de test. Cependant, les modèles ADM à 3 sous-groupes et 5 sous-groupes permettent d'obtenir de meilleurs TBC mais le maximum de la vraisemblance pour l'estimation des paramètres n'est pas atteint d'où le choix de 4 sous-groupes comme meilleure partition. Le Perceptron multicouche est le meilleur modèle d'apprentissage (79%) et reste moins performant dans la prédiction des données de test (78% de TBC) comparé aux modèles ADQ, ADF (BRUTO) et ADF (MARS).

Tableau 4.6 – Taux de bonne classification des différents modèles

Modèles	*Train*	*Test*
ADL	0.65	0.67
ADQ	0.77	0.81
ADM(3 sous-groupes, dev = 686.001)	0.77	0.79
ADM(4 sous-groupes, dev = 700.92)	0.78	0.78
ADM(5 sous-groupes, dev = 625.813)	0.78	0.79
ADF (polyreg)	0.77	0.79
ADF(bruto)	0.77	0.81
ADF (bruto)	0.78	0.81
ADP (gen.ridge)	0.77	0.79
PMC	0.79	0.78

De façon générale, sur la base du taux de bonne classification, il est préférable de prédire avec des modèles d'analyse discriminante (ADQ, ADF, ADP, ADM) qu'un modèle de perceptron multicouche.

Une seconde application avec un plus grand jeu de données sera traitée afin d'étudier les performances des modèles sous l'effet de la taille des données.

4.3 Application des méthodes d'analyse discriminante et du PMC aux données PULSAR

4.3.1 Présentation des données HTRU2 et quelques statistiques descriptives

HTRU2 [2](relative aux données sur les Pulsar) est un ensemble de données qui décrit un échantillon de «candidats» pulsars recueillis au cours de l'enquête haute résolution sur l'univers. Les pulsars sont un type rare d'étoile à neutrons qui produisent des émissions radio détectables ici sur Terre. Ils présentent un intérêt scientifique considérable en tant que sondes de l'espace-temps, du milieu inter-stellaire et des états de la matière. Lorsque les pulsars tournent, leur faisceau d'émission balaie le ciel, lorsqu'il traverse notre ligne de mire, produit un motif détectable d'émission radio à large bande. Comme les pulsars tournent rapidement, ce motif se répète périodiquement. Ainsi, la recherche de pulsars implique la recherche de signaux radio-périodiques avec de grands radiotélescopes.

Chaque pulsar produit un motif d'émission légèrement différent, qui varie légèrement avec chaque rotation. Ainsi, une détection de signal de potentiel connue sous le nom de «candidat» est moyennée sur de nombreuses rotations du pulsar, comme déterminé par la longueur d'une observation. En l'absence d'informations supplémentaires, chaque «candidat» pourrait potentiellement décrire un vrai pulsar. Cependant, dans la pratique, presque toutes les détections sont provoquées par des interférences de radio-fréquences (RFI) et du bruit, rendant les signaux définissants de vrais pulsars difficiles à trouver.

Les outils d'apprentissage automatique sont maintenant utilisés pour étiqueter automatiquement les «candidats» pulsars afin de faciliter l'analyse rapide. Les systèmes de classification, en particulier, sont largement adoptés et traitent les ensembles de

2. https ://www.kaggle.com/pavanraj159/predicting-pulsar-star-in-the-universe/data

données «candidats» comme des problèmes de classification binaires. Ici, les exemples de vrais pulsars sont une classe positive minoritaire et les exemples de faux pulsars, la classe négative majoritaire.

L'ensemble de données contient 17898 observations dont 16259 observations de faux pulsars causés par RFI/bruit et 1639 observations de pulsars réels. Ces exemples ont tous été vérifiés par des annotateurs humains. Les étiquettes de classe utilisées sont 0 (négatif) et 1 (positif).

Chaque «candidat» est décrit par 8 variables continues explicatives et une variable classe à expliquer. Les quatre premières sont des statistiques simples obtenues à partir du profil d'impulsion intégré (profil plié). Il s'agit d'un tableau de variables continues décrivant une version du signal résolue en longitude dont la moyenne a été calculée en temps et en fréquence. Les quatre variables restantes sont obtenues de la même manière à partir de la courbe DM-SNR *(Dispersion measure - Signal-to-noise ratio).* Celles-ci sont résumées ci-dessous :

V1 : Moyenne du profil intégré ;

V2 : Écart type du profil intégré ;

V3 : Coefficient d'aplatissement en excès du profil intégré ;

V4 : Coefficient d'asymétrie du profil intégré ;

V5 : Moyenne de la courbe DM-SNR ;

V6 : Ecart type de la courbe DM-SNR ;

V7 : Coefficient d'aplatissement excédentaire de la courbe DM-SNR ;

V8 : Asymétrie de la courbe DM-SNR ;

class : La variable classe vaut 1 = pulsar réel/positif et 0 = faux pulsar /négatif.

D'après le tableau 4.7, certaines variables sont fortement corrélées notamment V3 et V4 (cor = 0.946), V7 et V8 (cor = 0.924) de même que les variables V1 et V3 (cor = -0.87). Par contre, la plupart des variables ne sont pas corrélées entre elles. Ainsi, le recours de l'ADP pour palier à un problème lié à une forte corrélation des variables explicatives peut ne pas être nécessaire dans cette situation. De ce fait, on peut se fier à l'ADL et ne pas penser à utiliser l'ADP.

Les matrices de variance-covariances représentant respectivement le Tableau 4.9 et

Tableau 4.7 – Matrice de corrélation des données PULSAR

	V1	V2	V3	V4	V5	V6	V7	V8
V1	1	0.547	-0.874	-0.739	-0.299	-0.307	0.234	0.144
V2	0.547	1	-0.521	-0.540	0.007	-0.048	0.029	0.028
V3	-0.874	-0.521	1	0.946	0.414	0.433	-0.341	-0.214
V4	-0.739	-0.540	0.946	1	0.412	0.415	-0.329	-0.205
V5	-0.299	0.007	0.414	0.412	1	0.797	-0.616	-0.354
V6	-0.307	-0.048	0.433	0.415	0.797	1	-0.810	-0.576
V7	0.234	0.029	-0.341	-0.329	-0.616	-0.810	1	0.924
V8	0.144	0.028	-0.214	-0.205	-0.354	-0.576	0.924	1

le Tableau 4.10 en Annexe ne sont pas identiques. Cela prouve qu'on est dans une situation d'hétéroscédasticité où l'ADQ pourrait fournir des résultats de prédiction plus intéressants comparée à l'ADL.

4.3.2 Résultats des différents modèles

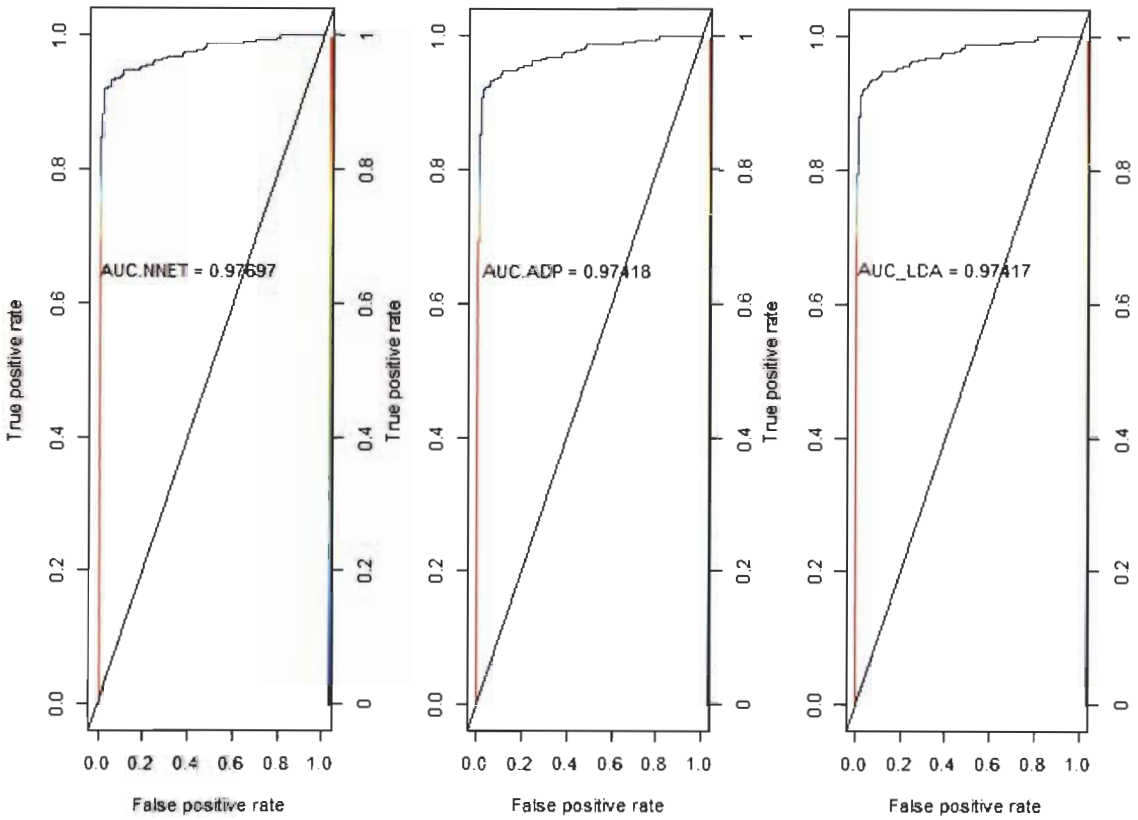
En se basant sur les taux de bonne classification (Tableau 4.8), le perceptron multicouche est le meilleur modèle d'apprentissage et de prédiction. En effet, les TBC des données d'entrainement et de test représentent respectivement 99.995% et 100%. Parmi les modèles d'analyse discriminante, l'ADM à deux sous-groupes fournit le meilleur modèle d'apprentissage (TBC = 97.81%), tandis que ceux qui prédissent le mieux sont l'ADQ et l'ADF (MARS) avec un TBC de 97.98%. L'ADF est le modèle d'apprentissage le moins performant (TBC = 97.46%). A part la méthode MARS, les méthodes de discrimination BRUTO, POLYREG et GEN.RIDGE de l'ADF ont les taux de bonne classification les moins élevés (97.62%). Une remarque importante à signaler concerne la quasi-similarité des modèles ADL et ADP dans l'apprentissage comme dans la prédiction qui est une conséquence de la relation existante entre les variables explicatives (pas de fortes corrélations de façon générale).

Tableau 4.8 – Taux de bonne classification des différents modèles

Modèles	*Train*	*Test*
ADL	97.47	97.62
ADQ	96.69	96.98
ADM(2 sous-groupes, dev = 4450.119)	97.81	97.90
ADM(3 sous-groupes, dev = 2977.757)	97.69	97.74
ADM(4 sous-groupes, dev = 3446.379)	97.58	97.93
ADM(5 sous-groupes, dev = 3788.525)	97.71	97.91
ADM(6 sous-groupes, dev = 3978.7)	97.44	97.79
ADM(7 sous-groupes, dev = 4097.147)	97.49	97.74
ADF (polyreg)	97.46	97.62
ADF (bruto)	97.46	97.62
ADF (mars)	97.46	97.98
ADP (gen.ridge)	97.46	97.62
PMC	99.995	100

Sur la base de la courbe de ROC (Graphique 4.1), le perceptron multicouche (PMC) reste toujours le meilleur modèle avec une AUC = 0.98. Ainsi, la nature (vrai ou faux pulsar) d'un nouveau «candidat» pourrait être identifiée avec un risque d'erreur quasiment nul par le modèle PMC(nnet). Ce dernier est suivi de l'ADL et de l'ADP ayant presque les mêmes valeurs de l'AUC (0.97418 et 0,97417 respectivement). Ces résultats pouvaient être prévus du fait de la qualité des données. En effet, d'après le tableau 4.7, les variables explicatives ne sont pas fortement corrélées à part quelques unes (voir la sous-section 4.3.1). Cependant, malgré ses avantages par rapport aux TMC et TBC, la courbe ROC reste sensible à l'échantillonnage des données d'entrainement et de test. Plus précisément, chaque fois que de nouveaux échantillons (entrainement et test) sont générés, une légère variation de l'AUC peut survenir.

Graphique 4.1 - Courbes de ROC des trois meilleurs modèles selon l'AUC



Les résultats obtenus avec les données *PULSAR* comparés à ceux des données RFMT de la première application prouvent que la prédiction avec un perceptron multicouche pourraient être préférable aux méthodes d'analyse discriminante dans des cas où la taille de la population est trop élevée. En effet, les méthodes classiques exigent souvent certaines particularités des données notamment l'absence de valeurs manquantes, pas de fortes corrélations entre prédicteurs et la normalité des données. Ainsi, surmonter ces limites qui sont souvent fréquentes dans de grandes bases de données a un coût temporel et financier. Par contre, les méthodes neuronales peuvent s'appliquer sans penser à un traitement des données aux préalables (Virole, 2001) [25]. Néanmoins, il faut préciser qu'un travail de traitement du jeu de données pourrait permettre d'obtenir de meilleurs résultats.

Conclusion et perspectives

Dans le but de faire le lien entre les méthodes statistiques classiques et neuronales à des fins de classification et de prédiction, nous avons étudié dans ce document quelques méthodes de classification supervisée. L'analyse discriminante linéaire qui est une des méthodes classiques est décrite avec ses différentes variantes notamment la quadratique (ADQ), la mixte (ADM), la flexible (ADF) et la pénalisée (ADP). Le perceptron multicouche, une des méthodes d'apprentissage automatique est également décrite. Toutes ces méthodes sont utilisées à des fins de prédiction d'une variable à expliquer (catégorielle) sur la base de variables explicatives (quantitatives). Cependant, d'une part elles n'ont pas les mêmes bases théoriques, d'autre part, elles n'ont pas les mêmes performances dans l'apprentissage et dans la prédiction.

La description théorique de chacune des méthodes a permis de mieux comprendre leur principe. De plus, du point de vue pratique, des modèles basés sur les techniques susmentionnées ont été effectués, afin de mesurer la qualité d'apprentissage, mais également de comparer leurs performances. Ainsi, les méthodes d'analyse discriminante restent sensibles aux types de données étudiées. En d'autres termes, l'ADL exige l'hypothèse d'homoscédasticité et de normalité des données, mais également une corrélation entre prédicteurs pas assez forte. Au cas où une de ces hypothèses n'est pas vérifiée, il serait préférable de recourir aux autres techniques d'analyse discriminante dont l'ADQ, l'ADM, l'ADF et l'ADP. En ce qui concerne le perceptron multicouche, il peut fournir de bons résultats peu importe la situation des données. Cependant, les simulations des différentes techniques avec les Iris de Fisher et les applications sur les

données *RFMT* et *PULSAR* indiquent que le PMC pourraient fournir de meilleurs résultats dans l'apprentissage comme dans la prédiction (d'après le TBC, le TMC et l'AUC) dans des situations où les données sont de grande taille.

Comme perspectives, il serait idéale de penser à rendre les méthodes classiques plus efficaces dans l'exploitation de grandes bases de données. De ce fait, quelque soit la taille des données à modéliser, les méthodes d'analyse discriminante pourraient être beaucoup plus utiles que le PMC au cas où certaines hypothèses sont vérifiées notamment l'homoscédasticité et la normalité des données (ADL), l'hétéroscédasticité (ADQ), le nombre de variables explicatives trop élevé (ADF) et une forte corrélation de ces dernières (ADF).

Bibliographie

[1] T. Hastie, R. Tibshirani et A. Buja, « Flexible discriminant analysis by optimal scoring », *Journal of the American Statistical Association*, vol. 89, no. 428, p. 1255–1270, 1994.

[2] T. Hastie, A. Buja et R. Tibshirani, « Penalized discriminant analysis », *Ann. Statist.*, vol. 23, p. 73–102, 02 1995.

[3] W. S. McCulloch et W. Pitts, « A logical calculus of the ideas immanent in nervous activity », *The bulletin of mathematical biophysics*, vol. 5, p. 115–133, Dec 1943.

[4] F. Rosenblatt, « The perceptron : A probabilistic model for information storage and organization in the brain », *Psychological Review*, vol. 65, p. 386–408, 1958.

[5] D. O. Hebb, *The organization of behavior :*. New York : Wiley.

[6] M. L. Minsky et S. Papert, *Perceptrons : An Introduction to Computational Geometry*. first éd., 1969.

[7] J. Boucher, « L'intelligence artificielle : histoire et notions de base (partie 1 de 2) », *Hinnovic.org*, 2018.

[8] G. Mclachlan, « Mahalanobis distance », *Resonance*, vol. 4, p. 20–26, 06 1999.

[9] P. Casin, « Categorical multiblock linear discriminant analysis », *Journal of Applied Statistics*, vol. 45, p. 1–14, 09 2017.

[10] J. Ulmo, « Différents aspects de l'analyse discriminante », *Revue de Statistique Appliquée*, vol. 21, no. 2, p. 17–55, 1973.

[11] B. Ghojogh et M. Crowley, « Linear and quadratic discriminant analysis : Tutorial », *arXiv e-prints*, p. arXiv :1906.02590, Jun 2019.

[12] S. Bose, A. Pal, R. SahaRay et J. Nayak, « Generalized quadratic discriminant analysis », *Pattern Recognition*, vol. 48, p. 2676–2684, 08 2015.

[13] T. Hastie, R. Tibshirani et J. Friedman, *The Elements of Statistical Learning*. Springer Series in Statistics, New York, NY, USA : Springer New York Inc., 2001.

[14] A. P. Dempster, N. M. Laird et D. B. Rubin, « Maximum likelihood from incomplete data via the em algorithm », *Journal of the royal statistical society, series B*, vol. 39, no. 1, p. 1–38, 1977.

[15] T. Hastie et R. Tibshirani, « Discriminant analysis by gaussian mixtures », *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 58, no. 1, p. 155–176, 1996.

[16] L. Lebart, A. Morineau et M. Piron, « Statistique exploratoire multidimensionelle », 01 1995.

[17] J. H. Friedman, « Multivariate adaptive regression splines », *The Annals of Statistics*. vol. 19, p. 1–67, 03 1991.

[18] T. Hastie et R. Tibshirani, *Generalized additive models*. London : Chapman & Hall, 1990.

[19] T. Hill et P. Lewicki, *Statistics : Methods and Applications : A Comprehensive Reference for Science, Industry, and Data Mining*. 01 2006.

[20] D. Rumelhart, G. Hinton et R. Williams, « Learning representations by back-propagating errors », *Nature*, vol. 323, no. 6088, p. 533–536, 1986.

[21] H. Delacour, A. Servonnet, A. Perrot, V. Jf et J. Ramírez, « La courbe roc (receiver operating characteristic) : principes et principales applications en biologie clinique », 2005.

[22] F. J. PROVOST, T. FAWCETT et R. KOHAVI, « The case against accuracy estimation for comparing induction algorithms », *in Proceedings of the Fifteenth International Conference on Machine Learning*, ICML '98, (San Francisco, CA, USA), p. 445–453, Morgan Kaufmann Publishers Inc., 1998.

[23] F. RIOULT, « Interprétation graphique de la courbe roc », p. 301-302, 01 2011.

[24] I.-C. YEH, K.-J. YANG et T.-M. TING, « Knowledge discovery on rfm model using bernoulli sequence », *Expert Syst. Appl.*, vol. 36, p. 5866–5871, apr 2009.

[25] B. VIROLE, *Réseaux de neurones et psychométrie*. Editions du Centre de Psychologie Appliquée - ECPA, 06 2001.

Annexes

Annexe 1

Tableau 4.9 – Matrice de variance-covariances des variables pour la classe 0

	V1	V2	v3	v4	V5	V6	V7	V8
V1	305.408	44.509	-4.971	-10.193	50.056	38.961	-8.775	-146.063
V2	44.509	38.229	-0.813	-4.716	35.519	21.244	-4.291	-61.096
v3	-4.971	-0.813	0.112	0.219	-0.852	-0.591	0.131	2.026
v4	-10.193	-4.716	0.219	1.056	-1.166	-1.044	0.272	4.571
V5	50.056	35.519	-0.852	-1.166	595.917	315.515	-57.185	-782.840
V6	38.961	21.244	-0.591	-1.044	315.515	277.270	-54.345	-963.199
V7	-8.775	-4.291	0.131	0.272	-57.185	-54.345	17.966	420.798
V8	-146.063	-61.096	2.026	4.571	-782.840	-963.199	420.798	11,389.570

Tableau 4.10 – Matrice de variance-covariances des variables pour la classe 1

	V1	V2	v3	v4	V5	V6	V7	V8
V1	900.462	138.050	-52.360	-350.278	-736.222	-256.409	50.773	542.187
V2	138.050	64.539	-10.514	-83.116	-16.106	-19.934	4.618	76.394
v3	-52.360	-10.514	3.508	25.372	41.708	15.778	-2.963	-31.715
v4	-350.278	-83.116	25.372	195.922	265.058	97.467	-18.353	-195.627
V5	-736.222	-16.106	41.708	265.058	2,050.997	627.279	-98.891	-818.551
V6	-256.409	-19.934	15.778	97.467	627.279	389.316	-52.725	-601.143
V7	50.773	4.618	-2.963	-18.353	-98.891	-52.725	9.647	136.549
V8	542.187	76.394	-31.715	-195.627	-818.551	-601.143	136.549	2,590.430

Fonctions R utilisées pour les modèles d'analyse discriminante[3]

3. R-manual

Definitions des fonctions R utilisées (R-Manual)

— **lda** (package : MASS) :

lda(formula, data, $\cdots$, subset, na.action)

formula : groupe $\sim x_1 + x_2 + \cdots$ (groupe représente la variable cible) ;

data : Le jeux de données à partir de la quelle les variables x_1, x_2, $\cdots$ sont extraites de préférences ;

subset : Un vecteur index spécifiant les cas à utiliser dans l'échantillon d'apprentissage ;

na.action : Cette option spécifie la façon de traiter les valeurs manquantes. Par défaut, ces dernières rentre dans la procédure. Par contre, "na.omit" permet d'ignorer les observations ayant des valeurs manquantes.

— **qda** (package : MASS) : Les mêmes arguments que lda sont utilisés dans qda

qda(formula, data, $\cdots$, subset, na.action)

— **mda** (package : mda) :

mda(formula, data, subclasses, sub.df, tot.df, dimension, eps, iter, weights, method, keep.fitted, trace, $\cdots$)

subclasses : Permet de spécifier le nombre de sous-groupes à considérer pour chaque groupe. Par défaut, chaque groupe est divisé en trois (3) sous groupes ;

sub.df : Permet de spécifier les degrés de liberté effectifs des centroïdes par classe.

tot.df : Permet de spécifier le nombre total de degré de liberté de tous les centroïdes plutôt que de les traiter groupe par groupe.

dimension : Cette option permet de fixer la dimension du modèle final ;

EPS : Un seuil numérique pour tronquer automatiquement la dimension ; *iter* : Elle permet de fixer le nombre limite d'itération dans *EM*. Par défaut cette valeur est 5 ;

poids : C'est une structure de pondération spéciale qui attribue à chaque observation de cette classe une pondération (probabilité antérieure) appartenant à l'une des sous-classes.

— **MclustDA** (Package : mclust)

MclustDA(data, class, G = NULL, modelNames = NULL, modelType = c("MclustDA", "EDDA"), prior = NULL $\cdots$)

class : Spécifie la variable cible à expliquer ;

G : un vecteur entier définissant le nombre de composants de mélange. Par defaut G = 1 :5 ;

modelNames : Un vecteur de chaînes de caractères indiquant les modèles à adapter par *EM* dans chaque classe ;

modelType : Une chaîne de caractères spécifiant si les modèles indiqués dans *modelNames* doit correspondre à un nombre différent de composants de mélange et de structures de covariance pour chaque classe ("MclustDA"valeur par défaut) ou doit être contrainte à avoir un seul composant pour chaque classe avec la même structure de covariance parmi les classes ("EDDA").

— **fda** : (Package : fda)

fda(formula, data, weights, theta, dimension, eps, method, keep.fitted, ···) ;

weights : Un vecteur facultatif de poids d'observation.

theta : une matrice facultative de scores de classe, avec moins de J-1colonnes en général.

eps : Un seuil pour les petites valeurs singulières pour exclure les variables discriminantes ;

methode : méthode de régression utilisée dans la mise à l'échelle optimale. La régression linéaire par défaut via la fonction *polyreg* entraîne une analyse discriminante linéaire. D'autres options peuvent être utilisées notamment *mars* et *bruto*. Pour les discriminants pénalisés, une analyse *gen.ridge* est appropriée. *keep.fitted* : une variable logique, qui détermine si le composant *fitted.values* du fit composant de l'objet *fda* renvoyé doit être conservé. La valeur par défaut est *TRUE* si *n * dimension < 5000*.

— **PenalizedLDA** (Package : penalizedLDA) :

PenalizedLDA(x, y, xte=NULL, type = "standard", lambda, K = 2,···) ;

x,y : x est le jeu de donnée, une matrice nxp ; y est la variable cible à expliquer ;

xte : représente le jeux donnée test ;

type : Il peut être "standard" ou "order". Le type est *standard* si le pénalisation *lasso* est utilisée. Le type *order* si les entités sont ordonnées ;

lambda : Si type = Il est "standard" s'il s'agit du paramètre de réglage de la pénalité

de lasso. Le type est *ordonné* s'il s'agit du paramètre de réglage pour la pénalité de lasso fusionné ;

K : Le nombre de vecteurs discriminants souhaité, il doit être inférieur ou égal au *nombre de classes - 1* ;

Annexe 2

```
\textbf{programme R Application Iris}
####### Programme l'ADL ADM ********************
data(iris)
if (!require("pacman")) install.packages("pacman")
pacman::p_load(tidyverse, data.table,lubridate, lattice, MASS,mda,
caret,
mlbench,
lattice,
klaR,
mclust,
tidyverse,
klaR,
ggplot2,
quantreg,
caret,
MASS,
car,
combinat,
```

```
dplyr,
mclust)



#data(BreastCancer)
attach(iris)
iris
str(iris)
scatterplotMatrix(iris[,-5])
res.man <- manova(cbind(Sepal.Length, Petal.Length,
Sepal.Width, Petal.Width) ~
 Species, data = iris)
summary(res.man, "Wilk")
summary(Manova, "Wilks")
#pairs(iris[,-5],upper.panel=panel.cor)
#install.packages("mlbench")
index = sample(1:nrow(iris), size=0.7*nrow(iris))
train = iris[index,]
test = iris[-index,]
lda.train <- lda(iris$Species~., iris, prior = c(1,1,1)/3)
plot(lda.train)
lda.train
res<-lda(Species~., iris)
cor(iris[,-5])
mu.k <- lda.train$means

###############""Partimat#################"""
mu <- colMeans(mu.k)
dscores <- scale(iris[,1:4], center=mu) %*% lda.train$scaling
species <- factor(rep(c("s","c","v"), each=50))
```

```
partimat(x=dscores[,2:1], grouping=species, method="lda",
 col.correct='blue', col.wrong='red')
#legend("topright", legend=c("setosa","versicolor", "virginica"),
col=1:3, pch=1.5)
#partimat(Species ~. , data = iris, method = "lda",
 plot.matrix = FALSE, col.correct='blue', col.wrong='red')

#####Prédiction###
pred <- predict(lda.train,iris)$class
%table(Predicted=pred, train$Species)
table(Predicted=pred, iris$Species);mean(pred == iris$Species)
table(Predicted=pred, train$Species);mean(pred == train$Species)

str(pred)
plot(lda.train)############### Nuage de points des deux fonctions
 discriminante ###############
plot(lda.train, dimen = 0, train$Species) #Histogramme des valeurs

empilées fournies par LDA 1
##plot limite##
####  QDA ##########################""
qda.train <-qda(Species~., data = iris, graph = TRUE)
qda.train
pred.qda <-predict(qda.train, iris)
table(pred.qda$class, iris$Species); mean(pred.qda$class == iris$Species)
### plot limites partimat####
mu.k <- qda.train$means
mu <- colMeans(mu.k)
#dscores <- scale(iris[,1:4], center=mu) %*% lda.train$scaling
Species <- factor(rep(c("s","c","v"), each=50))
```

```
partimat(x=iris[,-5], grouping=Species, method="qda", col.correct='blue',
 col.wrong='red')
#partimat(Species ~ ., data = iris, method = "qda", plot.matrix = FALSE,
 col.correct='green', col.wrong='red')
str(train)

##################### mda ###################################
mda.train <- mda(iris$Species~., iris, subclass = 2, main=FALSE)
plot(mda.train)

layout(matrix(1:2,nrow=1))##Afiche tes graphiques cote a cote
for (j in 4:5){
# On trace un nuage de points qui correspond a l'ensemble des données
# (hommes et femmes)
plot(mda.train <- mda(iris$Species~., iris, subclass = j),
main=c(NULL,NULL))

legend("top", legend=c("virginica", "setosa", "versicolor"),
col=c(4,2,3), pch=1)
}

for (j in 1:16){
for (i in 1:16) {
x[i] <- mda(iris$Species~., iris, subclass = j)

}
}

Nombre_sous_groupes <- c(2,3,4,5,6,7,8,9,10,11,12,13,14)
TMC<- c(0.02,0.013,0.013,0.00667,0.013,0.013,0.00667,
```

```
0.013,0.013,0.00667,0.00667,0.00667,0)
deviance<- c(  12.27,13.3,10.93,11.893,8.724,11.486,
6.864,8.143,7.955,6.499,3.855,4.187,1.85)

layout(matrix(1:2,nrow=2))##Afiche tes graphiques cote a cote
plot(Nombre_sous_groupes, TMC, col = "red", type = "b")
plot(Nombre_sous_groupes, deviance, col = "blue", type = "b", add = TRUE)

pred.mda <- predict(mda.train, test)
table(pred.mda, test$Species); mean(pred.mda == test$Species)

plot(mda.train)
legend("topright", legend=c("virginica", "setosa", "versicolor"),
col=1:3, pch=1.5)

for(i in 2:16)
s =  i+1
seuil=16
s<-1
while (s <= seuil) {
mda <- mda(iris$Species~., iris, subclasses = s)
plot(mda)}

s<-16
mda <- mda(iris$Species~., iris, subclasses = s)

###################  FDA ######################
###################  fda MARS ############
fda.train <- fda(iris$Species~., data = iris, method = polyreg,
 cv = T) # TBC 93,3%
```

```
fda.train
plot(fda.train)
legend("topright", legend=c("virginica", "setosa", "versicolor"),
col=1:3, pch=1.5)

summary(fda.train)
confusion(fda.train)
pred.fda <- predict(fda.train, test, cv = T)
table(pred.fda, test$Species); mean(pred.fda == test$Species)

############# fda bruto ##########################
fda.train <- fda(train$Species~., data = train, method = bruto,
 cv = T) # TBC 93,3%
fda.train

summary(fda.train)
confusion(fda.train)
pred.fda <- predict(fda.train, test, cv = T)
table(pred.fda, test$Species); mean(pred.fda == test$Species)

pda.train <- fda(train$Species~., data = train, method = gen.ridge,
 cv = T)  # TBC 96,6%
pda.train

pred.pda <- predict(pda.train, test, cv = T)
table(pred.pda, test$Species); mean(pred.pda == test$Species)

##################" fda polyreg #########
fda.train <- fda(iris$Species~., method = polyreg,data = iris,
graph = TRUE, cv = T) ## TBC 96,6%
```

```
fda.train

summary(fda.train)
confusion(fda.train)
pred.fda <- predict(fda.train, test, cv = T)
table(pred.fda, test$Species); mean(pred.fda == test$Species)

plot(fda.train)
system.time(fda.train <-fda(Species~., data = train)) ### Temps d'excussion

##################" fda gen.ridge #########
pda.train <- fda(iris$Species~., method = gen.ridge,
data = iris,graph = TRUE, cv = T) ## TBC 96,6%
pda.train
lambda<-c(0.5, 2,10,30,100,1000)
TMC<-c(0.02, 0.02,0.03,0.04,0.05333,0.06667)
plot(lambda,TMC, col = "red", type = "b")
plot(pda.train)
legend("bottom", legend=c("virginica", "setosa", "versicolor"),
col=1:3, pch=1.5)

fda.train <- fda(iris$Species~., data = iris, methode = "gen.ridge",
 lambda = 0.5, cv = T) # TBC 93,3%
fda.train

################## nnet ####################
library(neuralnet)
n <- names(train)
f <- as.formula(paste("Species ~", paste(n[!n %in% "Species"],
collapse = " + ")))
```

```
set.seed(2)
nn = neuralnet(Species ~ Sepal.Length+Sepal.Width+Petal.Length +
 Petal.Width,
 train, hidden = 1,act.fct = "logistic"
, linear.output = T)

library(nnet)
nnet.train <- nnet(Species~., data=train, hidden = 1, size=3, decay=0.1,
maxit=150)
predictions <- predict(nnet.train, test, type="class")
table(predictions, test$Species); mean(predictions != test$Species)
plot(nnet)
#################### lIMITE ZONE MclustDA ###
class<-iris$Species
mod<-MclustDA(iris[,-5], class)
summary(mod)
plot(mod)
plot(mod,"error")
mod1dr <- MclustDR(mod, lambda=-300)
plot(mod1dr, what = "density",ngrid = 400,  type="persp")#### very Good
plot(mod1dr, what = "boundaries",ngrid = 700)
legend("topright", legend=c("setosa", "Versicolor", "virginica"),
col=c(4,2,3),
pch=c(16,22,17))
legend = rownames(train$chd)
legend = rownames(train$boundaries)
plot(mod1dr, what = "boundaries",ngrid = 800,
bg=c("red","green3","blue")[train[,5]],
pch=c(21,25,24)[iris[,5]],
main="Iris de Fisher")
```

```
### Applicaion sur les données PULSAR

setwd("C:/Users/dell/Dropbox/UQTR 2018/Version 3 memoire/Version 3_2
/Application/AP2")
data<-read.csv2("pulsor.csv", dec = ".")
data<-rename(data, "class" = "target_class" )
data<-rename (data,"V1"
= "Mean.of.the.integrated.profile", "V2"="Standard.deviation.of.
the.integrated.profile",
"v3"="Excess.kurtosis.of.the.integrated.profile", "v4" =
"Skewness.of.the.integrated.profile",
"V5"= "Mean.of.the.DM.SNR.curve","V6"="Standard.deviation.
of.the.DM.SNR.curve", "V7"="Excess.kurtosis.of.
the.DM.SNR.curve" ,"V8"="Skewness.of.the.DM.SNR.curve")

#############DATA PARTITION ##############################

index = sample(1:nrow(data), size=0.8*nrow(data))
train = data[index,]
test = data[-index,]
#################   Stats desc ###################
stargazer(cor(data))
data1<-data[which(data$class==0), ]
data2<-data[ which(data$class==1), ]
stargazer(cov(data1[,-9]))
stargazer(cov(data1[,-9]))
variable.names(data)
shapiro.test(data[,-9])########### Test de normalité#####"
################################### LDA ####################
```

```
lda<-lda(train$class~., train)
lda
plot(lda.train)
pred.lda <- predict(lda,test)$class
table(pred.lda, test$class); mean(pred.lda == test$class)

################################## QDA ######################
qda.train <- qda(train$class~., data = train)
qda
pred.qda <- predict(qda.train, train)$class
table(pred.qda, train$class); mean(pred.qda == train$class)
plot(qda.transf)
predict<-pred.qda -1;
actual<- test$Class
rmse(predict, actual)
error<- predict - (actual)
rmse.qda<-rmse(error); rmse.qda
mae.lda<-mae(error);mae.lda
rmse(pred.qda)
pred.qda <- predict(qda.transf, test)
library(ROCR)
pred.qda
#pred<-ifelse(pred.qda < 0.6, 0,1)
perf=performance(pred.qda,"tpr", "fpr")
plot(perf,colorize = TRUE)
################################ mda ####################
mda.train <- mda(train$class~., data = train, subclasses = 7,
lambda = 1)
mda.train
plot(mda.train)
```

```
pred.mda <- predict(mda.train, test)
table(pred.mda, test$class)
mean(pred.mda == test$class)
predict<-c(pred.mda) -1;
actual<- test$Class
error<- predict - (actual)
rmse.qda<-rmse(error); rmse.qda
mae.lda<-mae(error);mae.lda

library(pROC)
score <- predict(mda.train,test,type="class")
score
roc_obj <- roc(test$class score)
################################ fda polyreg#####################
fda.train <- fda(train$class~.,data = train,method = polyreg, plot=T)
fda.train
pred.fda <- predict(fda.train, test)
table(pred.fda, test$class)
mean(pred.fda == test$class)
predict<-c(pred.fda) -1;
actual<- test$Class
error<- predict - (actual)
rmse.qda<-rmse(error); rmse.qda
mae.lda<-mae(error);mae.lda
pred<-ifelse(pred.fda<0.6,0,1)
library(ROCR)
p=predict(fda.transf,type="Class")
```

```
roclogit=predict(pred,newdata=test)
predlogit=prediction(pred,test$Class)
perflogit=performance(pred.fda, "tpr","fpr")
plot(perflogit,col=1)



plot(perf,colorize = TRUE)
pred.fda
################################ fda Bruto ####################
fda.train <- fda(train$class~., method = bruto, lambda = 0.1,
data = train)
fda.train
pred.fda <- predict(fda.train, test)
table(pred.fda, test$class)
mean(pred.fda == test$class)
predict<-c(pred.fda) -1;
actual<- test$Class
error<- predict - (actual)
rmse.qda<-rmse(error); rmse.qda
mae.lda<-mae(error);mae.lda

################################ FDA mars ####################
fda.transf <- fda(train$class~., method = mars, lambda = 0.1,
data = train)
fda.train
pred.fda <- predict(fda.transf, test)
table(pred.fda, test$class)
mean(pred.fda == test$class)
predict<-c(pred.fda) -1;
```

```
actual<- test$Class
error<- predict - (actual)
rmse.qda<-rmse(error); rmse.qda
mae.lda<-mae(error);mae.lda

############################## fda gen.ridge ####################
fda.train <- fda(train$class~., method = gen.ridge ,data = train,
plot = T)
fda.train
pred.fda <- predict(fda.transf, test)
table(pred.fda, test$class)
mean(pred.fda == test$class)
predict<-c(pred.fda) -1;
actual<- test$Class
error<- predict - (actual)
rmse.qda<-rmse(error); rmse.qda
mae.lda<-mae(error);mae.lda
plot(fda.transf)
######################### nnet  ############"
################ schema du réseau ##########"
library(neuralnet)
n <- names(train)
f <- as.formula(paste("class ~", paste(n[!n %in% "Class"],
collapse = " + ")))
nn <- neuralnet(f,data=train,hidden=c(3,1))
pred.nn <- predict(nn, test, type="class")
pred.nn<-ifelse(pred.nn  < 0.6, 0,1)
table(pred.nn, test$class)
mean(pred.nn == test$class)
plot(nn)
```

```
nn <- neuralnet(f,data=train,hidden=c(3,1))
pred.nn <- predict(nn, train, type="class")
pred.nn<-ifelse(pred.nn  < 0.6, 0,1)
table(pred.nn, train$Class)
mean(pred.nn == train$Class)

"
nnet.train <- nnet(Species~., data=train, size=8, decay=0.0001, maxit=500)
summary(nnet.train)
confusion(nnet.train)
predictions <- predict(nnet.train, iris, type="class")
table(predictions, iris$Species); mean(predictions != iris$Species)



library(nnet)
nnet.train <- nnet(Class~., data=train, size=10, decay=0.0001,
maxit=500, type="class")
summary(nnet.train)
predictions <- predict(nnet.train, test)
pred<-ifelse(predictions<0.5,0,1)
table(pred, test$Class); mean(pred == test$Class)

"

# Erreur quadratique moyenne de prévision
sqrt(sum((pred.nnet-test[,"Class"])^2)/nrow(test))
# Matrice de confusion pour la prévision du
# dépassement de seuil (régression)
%table(pred.nnet,test[,"Class"])
```

```
# Même chose pour la discrimination
table(pred.nnet>0.5,test[,"Class"])
rmse(pred.nnet)
table(test$Class)

########################################"
"
############MclustDA#####"
class<-train$class
mod<-MclustDA(train[,-9], class)
plot(mod)
summary(mod)
pred.nnet=predict(mod,newdata=test, type = "Class")
mod1dr <- MclustDR(mod)
plot(mod1dr, what = "class",ngrid = 100,  type="persp")
legend("topright", legend=c("class 0", "class 1" ),col=1:2, pch=(4))

#########################################################3

Modèles<- c("ADL","ADQ","MDA( 2 sous groupes, dev = 4450.119)",
"MDA( 3 sous groupes, dev = 2977.757)
","MDA( 4 sous groupes, dev = 3446.379 )","MDA( 5 sous groupes,
dev = 3788.525 )","MDA( 6 sous groupes, dev = 3978.7  )",
"MDA( 7 sous groupes, dev =  4097.147  )",
"ADF(bruto)","ADF (mars)","ADP (gen.ridge)","PMC")
Train<- c(0.65,0.77,0.77,0.78,0.78,0.77,0.77,
0.78,0.77,0.78)
Test<- c(0.67,0.81,0.79,0.77,0.79,0.79,0.81,0.81,
0.79,0.81)
table(Modèles, Train, Test)
```

```
resultats<-read.csv2("C:/Users/dell/Dropbox/UQTR 2018/
Version 4 memoire/Version 3_2/Application/AP2/resultats.csv")
table<-cbind(resultats)
stargazer(table)
table
###
f <- paste(names(train)[9], "~", paste(names(train)[-9], collapse=" + "))
lda2 <- lda(as.formula(paste(f)), data = train)
lda.predict <- predict(lda2, newdata = test)
wdbc_raw.lda.predict.posteriors <- as.data.frame(lda.predict$posterior)
pred <- prediction(wdbc_raw.lda.predict.posteriors[,2], test$class)
roc.perf1 = performance(pred, measure = "tpr", x.measure = "fpr")
auc.train <- performance(pred, measure = "auc")
auc.train <- auc.train@y.values
plot(roc.perf1, colorize = TRUE)
abline(a=0, b= 1)
text(x = .40, y = .65 ,paste("AUC_LDA = ",
round(auc.train[[1]],5), sep = ""))
################################ROC_QDA#########
f <- paste(names(train)[9], "~", paste(names(train)[-9], collapse=" + "))
qda <- qda(as.formula(paste(f)), data = train)
qda.predict <- predict(qda, newdata = test)
wdbc_raw.qda.predict.posteriors <- as.data.frame(qda.predict$posterior)
pred <- prediction(wdbc_raw.qda.predict.posteriors[,2], test$class)
roc.perf2 = performance(pred, measure = "tpr", x.measure = "fpr")
auc.train <- performance(pred, measure = "auc")
auc.train <- auc.train@y.values
plot(roc.perf2, colorize = TRUE)
abline(a=0, b= 1)
text(x = .40, y = .65 ,paste("AUC.QDA = ", round(auc.train[[1]],5), sep = ""))
```

```
############################ mda_2sous_groupes #################
f <- paste(names(train)[9], "~", paste(names(train)[-9], collapse=" + "))
mda <- mda(as.formula(paste(f)), data = train, subclasses = 2)
mda.predict <- predict(mda, newdata = test, "posterior")
wdbc_raw.mda.predict.posteriors <- as.data.frame(mda.predict)
pred <- prediction(wdbc_raw.mda.predict.posteriors[,2], test$class)
roc.perf3 = performance(pred, measure = "tpr", x.measure = "fpr")
auc.train <- performance(pred, measure = "auc")
auc.train <- auc.train@y.values
plot(roc.perf3,colorize = TRUE)
abline(a=0, b= 1)
text(x = .40, y = .65 ,paste("AUC.MDA = ", round(auc.train[[1]],5), sep = ""))
############################fda#################
f <- paste(names(train)[9], "~", paste(names(train)[-9], collapse=" + "))
fda <- fda(as.formula(paste(f)), data = train, method = mars)
fda.predict <- predict(mda, newdata = test, "posterior")
wdbc_raw.mda.predict.posteriors <- as.data.frame(mda.predict)
pred <- prediction(wdbc_raw.mda.predict.posteriors[,2], test$class)
roc.perf4 = performance(pred, measure = "tpr", x.measure = "fpr")
auc.train <- performance(pred, measure = "auc")
auc.train <- auc.train@y.values
plot(roc.perf4, colorize = TRUE)
abline(a=0, b= 1)
text(x = .40, y = .65 ,paste("AUC.MARS = ", round(auc.train[[1]],5), sep = ""))
############################pda#################
f <- paste(names(train)[9], "~", paste(names(train)[-9], collapse=" + "))
pda <- fda(as.formula(paste(f)), data = train, method = gen.ridge)
pda.predict <- predict(pda, newdata = test, "posterior")
wdbc_raw.mda.predict.posteriors <- as.data.frame(pda.predict)
pred <- prediction(wdbc_raw.mda.predict.posteriors[,2], test$class)
```

```
roc.perf5 = performance(pred, measure = "tpr", x.measure = "fpr")
auc.train <- performance(pred, measure = "auc")
auc.train <- auc.train@y.values
plot(roc.perf5, colorize = TRUE)
abline(a=0, b= 1)
text(x = .40, y = .65 ,paste("AUC.ADP = ", round(auc.train[[1]],5), sep = ""))
############################nnet#################
f <- paste(names(train)[9], "~", paste(names(train)[-9],
collapse=" + "))
nnet <- nnet(as.formula(paste(f)), data = train, size=6,
decay=0.001, maxit=500 )
nnet.predict <- predict(nnet, newdata = test)
wdbc_raw.mda.predict.posteriors <- as.data.frame(nnet.predict)
pred <- prediction(wdbc_raw.mda.predict.posteriors, test$class)
roc.perf6 = performance(pred, measure = "tpr", x.measure = "fpr")
auc.train <- performance(pred, measure = "auc")
auc.train <- auc.train@y.values
plot(roc.perf5, colorize = TRUE)
plot(roc.perf6, col="blue", type = "l")
plot(roc.perf5 ,col="red", add=TRUE)
plot(roc.perf4, col="green", add=TRUE)
plot(roc.perf3, col="brown", add=TRUE)
abline(a=0, b= 1)
text(x = .40, y = .65 ,paste("AUC.NNET = ", round(auc.train[[1]],5),
sep = ""))
legend("topright", names(a)[-1] , lty=1, col=c("red", "blue", "green",
"brown"), bty="n", cex=.75)

legend(-6,-1, yjust=0, c("NNET", "LDA"),
lwd=3, lty=1, col=c(par('fg'), 'red'),
```

```
)

a<-c("nnet","adp","fda(MARS)","mda")
legend("bottomright",legend=c("nnet","adp","fda(MARS)","mda"),
col=c("red", "green", "blue","brown"))

legend(x=-3,y=7,c("nnet","adp"),cex=.8,col=c("red","blue"),
pch=c(1,2))
plot(roc.perf6, )
lines(roc.perf4, type="l", col="green")
lines(roc.perf3, type="l", col="blue")
lines(roc.perf2, type="l", col="brown")
lines(roc.perf1, type="l", col="black")

nnet.predict <- predict(nnet, newdata = test, type = "response")
roc(test$class,nnet.predict, plot=TRUE)
roc(test$class,fda.predict,plot=TRUE,col="red",add=TRUE)
roc(test$class,pda.predict,plot=TRUE,col="blue",add=TRUE)
roc(test$class,mda.predict,plot=TRUE,col="blue",add=TRUE)
roc(test$class,lda.predict,plot=TRUE,col="blue",add=TRUE)
roc(test$class,qda.predict,plot=TRUE,col="blue",add=TRUE)
legend("bottomright",legend=c("nnet","fda","pda","mda","lda","qda")
col=c("green","red","blue", "black", "yellow","rainbow(5)"),lty=1,lwd=2)
```