

Apprentissage automatique

Julien Ah-Pine (julien.ah-pine@univ-lyon2.fr)

Université Lyon 2

M2 DM 2021/2022

Rappel du Sommaire

1 Introduction

- L'apprentissage automatique
- Quelques méthodes simples en guise d'illustration
- Différentes caractéristiques des méthodes d'apprentissage supervisé
- (Quelques) Problèmes théoriques en apprentissage automatique
- Evaluation et comparaison de modèles en apprentissage supervisé

Rappel du Sommaire

- 1 Introduction
- 2 Les méthodes linéaires et leurs pénalisations (ridge, lasso, ...)
- 3 Les machines à vecteurs supports ("Support Vector Machines")
- 4 Les arbres de décisions ("Decision Trees")
- 5 Décider en comité ("Ensemble Learning")

En quoi consiste l'apprentissage automatique ?

- De manière générale, un programme informatique tente de résoudre un problème pour lequel nous avons la solution. Par exemple : calculer la moyenne des étudiants, classer les étudiants selon leur moyenne. . .
- Pour certains problèmes, nous ne connaissons pas de solution exacte et donc nous ne pouvons pas écrire de programme informatique. Par exemple : reconnaître automatiquement des chiffres écrits à la main à partir d'une image scannée, déterminer automatiquement une typologie des clients d'une banque, jouer automatiquement aux échecs contre un humain ou un autre programme. . .
- En revanche, pour ces problèmes il est possible d'avoir une base de données regroupant de nombreuses instances du problème considéré.
- L'apprentissage automatique consiste alors à programmer des algorithmes permettant d'apprendre automatiquement de données et d'expériences passées, un algorithme cherchant à résoudre au mieux un problème considéré.

Un domaine pluri-disciplinaire

- L'apprentissage automatique (AA) (“**Machine Learning**”) est à la croisée de plusieurs disciplines :
 - ▶ Les **statistiques** : pour l'inférence de modèles à partir de données.
 - ▶ Les **probabilités** : pour modéliser l'aspect aléatoire inhérent aux données et au problème d'apprentissage.
 - ▶ L'**intelligence artificielle** : pour étudier les tâches simples de reconnaissance de formes que font les humains (comme la reconnaissance de chiffres par exemple), et parce qu'elle fonde une branche de l'AA dite symbolique qui repose sur la logique et la représentation des connaissances.
 - ▶ L'**optimisation** : pour optimiser un critère de performance afin, soit d'estimer des paramètres d'un modèle, soit de déterminer la meilleure décision à prendre étant donné une instance d'un problème.
 - ▶ L'**informatique** : puisqu'il s'agit de programmer des algorithmes et qu'en AA ceux-ci peuvent être de grande complexité et gourmands en termes de ressources de calcul et de mémoire.

AA et matières connexes (suite)

- Plus récemment :
 - ▶ La **science des données** (“Data science”) : approche(s) pluri-disciplinaire pour l'extraction de connaissances à partir de données hétérogènes [Cleveland, 2001, Abiteboul et al., 2014].
 - ▶ Les **données massives** (“Big data”) : mettant l'accent sur les problématiques “4V” (volume, variété, vélocité, véracité) et des éléments de solutions issus du stockage/calcul distribué [Leskovec et al., 2014].
 - ▶ Pour plus de ressources, consultez le site <http://www.kdnuggets.com>.

AA et matières connexes

- Quelques références et domaines d'application faisant intervenir l'AA :
 - ▶ Les **statistiques** (“Statistical Machine Learning”) : modèles d'AA traités sous l'angle des statistiques [Hastie et al., 2011, Dreyfus, 2008].
 - ▶ L'**intelligence artificielle** (“Artificial Intelligence”) : modèles d'AA mettant l'accent sur le raisonnement, l'inférence et la représentation des connaissances [Cornuéjols and Miclet, 2003, Mitchell, 1997, Alpaydin, 2010].
 - ▶ La **fouille de données** (“Data Mining”) : lorsque les objets étudiés sont stockés dans des bases de données volumineuses [Han and Kamber, 2006].
 - ▶ La **reconnaissance de formes** (“Pattern Recognition”) : lorsque les objets concernés sont de type “signal” comme les images, les vidéos ou le son [Bishop, 2006].
 - ▶ Le **traitement automatique du langage - TAL** (“Natural Language Processing” - NLP) : lorsque les problèmes concernent l'analyse linguistique de textes [Manning and Schütze, 1999, Clark et al., 2010].

Plusieurs types de problèmes en AA

- Apprentissage automatique :
 - ▶ **Supervisé** : on dispose d'un ensemble d'objets et pour chaque objet une valeur cible associée ; il faut apprendre un modèle capable de prédire la bonne valeur cible d'un objet nouveau.
 - ▶ **Non supervisé** : on dispose d'un ensemble d'objets sans aucune valeur cible associée ; il faut apprendre un modèle capable d'extraire les régularités présentes au sein des objets pour mieux visualiser ou appréhender la structure de l'ensemble des données.
 - ▶ **Par renforcement** : on dispose d'un ensemble de séquences de décisions (politiques ou stratégiques) dans un environnement dynamique, et pour chaque action de chaque séquence une valeur de récompense (la valeur de récompense de la séquence est alors la somme des valeurs des récompenses des actions qu'elle met en oeuvre) ; il faut apprendre un modèle capable de prédire la meilleure décision à prendre étant donné un état de l'environnement.

Plusieurs types de problèmes (suite)

- Apprentissage automatique (suite) :
 - ▶ **Semi-supervisé** : on dispose d'un petit ensemble d'objets avec pour chacun une valeur cible associée et d'un plus grand ensemble d'objets sans valeur cible ; il faut tirer profit à la fois des données avec et sans valeurs cibles pour résoudre des tâches d'apprentissage supervisé ou non supervisé.
 - ▶ **Actif** : on dispose d'un petit ensemble d'objets avec pour chacun une valeur cible associée ; il faut interagir avec l'utilisateur et lui demander de donner la valeur cible d'un nouvel objet afin de mieux apprendre le modèle de prédiction.
- Dans le cadre de ce cours, nous étudierons les problèmes d'**apprentissage supervisé** : il s'agit donc de définir et d'estimer des modèles de prédiction étant donné un ensemble d'objets et leurs valeurs cibles respectives. On parle également d'**algorithmes** d'apprentissage supervisé.

Apprentissage supervisé numérique

- Il existe deux types de sous-problèmes en apprentissage supervisé numérique :
 - ▶ **Régression** ("Regression") : lorsque la valeur cible à prédire est continue.
 - ▶ **Classement**, classification ou catégorisation ("Classification") : lorsque la valeur cible à prédire est discrète.
- Par ailleurs nous supposons également que les objets étudiés qui peuvent être complexes à l'origine (comme des données multimédia) sont représentés dans un format numérique structuré. En d'autres termes :
 - ▶ On représente un objet X_i par un vecteur noté \mathbf{x}_i défini dans un espace de description composé de plusieurs variables.
 - ▶ A chaque \mathbf{x}_i on lui associe une valeur cible notée y_i .

Apprentissage supervisé symbolique et numérique

- Deux familles en apprentissage supervisé [Cornuéjols and Miclet, 2003] :
 - ▶ Apprentissage supervisé **symbolique** : méthodes inspirées de l'intelligence artificielle et dont les fondements reposent beaucoup sur des modèles de logique, une représentation binaire des données (vrai/faux), et sur les méthodes de représentation des connaissances.
 - ▶ Apprentissage supervisé **numérique** : méthodes inspirées de la statistique, les données sont en général des vecteurs de réels, et les méthodes font intervenir des outils provenant des probabilités, de l'algèbre linéaire et de l'optimisation.
- Dans le cadre de ce cours, nous étudierons principalement les problèmes d'**apprentissage supervisé numérique** : le cours nécessite donc des prérequis de base dans les domaines sus-mentionnés.

Quelques exemples d'application

- Exemples de problèmes de régression :
 - ▶ Prédiction du montant des ventes d'une entreprise compte tenu du contexte économique.
 - ▶ Prédiction du prix de vente d'une maison en fonction de plusieurs critères.
 - ▶ Prédiction de la consommation électrique dans une ville étant donné des conditions météorologiques. . .
- Exemples de problèmes de catégorisation :
 - ▶ Prédiction de l'état sain/malade d'un patient par rapport à une maladie et compte tenu de différents facteurs.
 - ▶ Prédiction de l'accord ou du refus d'un crédit à un client d'une banque en fonction de ses caractéristiques.
 - ▶ Prédiction du chiffre correct à partir d'une image scannée d'un chiffre écrit à la main. . .

Motivations de ce cours

- Positionner le domaine de l'apprentissage automatique vis à vis des autres domaines scientifiques connexes (cf également mon cours de L3 CESTAT sur une brève rétrospective historique).
- Présenter quelques concepts importants du domaine :
 - ▶ Espaces d'hypothèses, fonctions objectif, méthodes d'inférence ou d'estimation ou d'optimisation.
 - ▶ Principe de généralisation, problèmes de sous et de sur-apprentissage, arbitrage biais-variance.
 - ▶ Données de grande dimension, expansion de bases.
- Présenter le protocole expérimental classique :
 - ▶ Ensembles d'apprentissage, de validation et de tests.
 - ▶ Estimation robuste de l'erreur en généralisation.
- Présenter des approches classiques et modernes d'apprentissage supervisé numérique.

Notations

- Comme données à notre disposition nous supposons que nous avons une table \mathbf{X} avec n lignes et p colonnes et un vecteur colonne (variable cible) \mathbf{y} de n éléments.

$$\mathbf{X} = \begin{pmatrix} x_{11} & x_{12} & \dots & x_{1p} \\ x_{21} & x_{22} & \dots & x_{2p} \\ \vdots & \dots & \dots & \vdots \\ x_{n1} & x_{n2} & \dots & x_{np} \end{pmatrix} \quad \text{et} \quad \mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}$$

- La ligne i de \mathbf{X} est associée à l'objet X_i et l'ensemble des objets $\{X_1, \dots, X_n\}$ sera noté \mathbb{O} .
- La colonne j de \mathbf{X} est associée à la variable ou attribut X^j et l'ensemble des variables $\{X^1, \dots, X^p\}$ sera noté \mathbb{A} .
- x_{ij} terme général de \mathbf{X} est la valeur de la variable X^j pour l'objet X_i .
- A chaque objet X_i est associée une valeur y_i de la variable $Y \in \mathbb{Y}$ où \mathbb{Y} est l'ensemble des valeurs que peut prendre Y .

Organisation de ce cours

- 12 séances de CM de 1h45 sur 3 semaines.
- Une feuille d'exercices (des exercices à faire à la maison chaque semaine - correction rapide en début ou fin de séance).
- Evaluation :
 - ▶ 1 projet (rapport + code R - coef. ~ 0.4).
 - ▶ 1 examen sur table individuel (2h - coef. ~ 0.6).
- Supports de cours sur la page moodle du cours.

Notations (suite)

- Chaque objet X_i est associé à un vecteur numérique \mathbf{x}_i appartenant à un espace de description \mathbb{X} .
- Sauf mention contraire, on supposera que \mathbb{X} est un espace vectoriel engendré par les variables $\{X^1, \dots, X^p\}$ (typiquement \mathbb{R}^p).
- Ainsi on notera par $\mathbf{x}_i = (x_{i1}, \dots, x_{ip})$ le vecteur colonne de taille $(p \times 1)$ des valeurs observées représentant X_i dans \mathbb{X} .
- On notera par $\mathbf{x}^j = (x_{1j}, \dots, x_{nj})$ le vecteur colonne de taille $(n \times 1)$ des valeurs observées sur \mathbb{O} pour la variable X^j .
- $\mathbf{y} = (y_1, \dots, y_n)$ est le vecteur colonne de taille $(n \times 1)$ des valeurs observées sur \mathbb{O} pour la variable cible Y .
- L'ensemble des couples observés $\mathbb{E} = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_i, y_i), \dots, (\mathbf{x}_n, y_n)\}$ est appelé **ensemble d'entraînement ou d'apprentissage** (ou ensemble des données annotées ou étiquetées).
- Nous dénoterons par X un objet quelconque, \mathbf{x} son vecteur représentant dans \mathbb{X} et y la valeur cible associée à \mathbf{x} .

Formalisation du problème

- Etant donné un ensemble d'entraînement \mathbb{E} , on cherche à déterminer $f : \mathbb{X} \rightarrow \mathbb{Y}$ une fonction **modélisant** la relation entre les X décrits dans l'espace de représentation \mathbb{X} et la variable cible Y :

$$f(X) = Y$$

- En revanche, ne connaissant pas la vraie nature de la relation entre X et Y et les données observées en $\{X^1, \dots, X^p\}$ étant soit bruitées, soit incomplètes ; il n'est pas raisonnable de supposer une relation déterministe. Aussi, il est davantage raisonnable de poser le problème en les termes suivants :

$$f(X) = Y + \epsilon$$

où ϵ est l'erreur ou le résidu.

- Autrement dit, il s'agit d'**approximer** f en commettant le **moins d'erreurs possibles** sur \mathbb{E} tout en faisant de **bonnes prédictions** pour des valeurs de \mathbb{X} non encore observées.

Schéma général (suite)

- Comme précisé précédemment, nous ne traitons pas dans ce cours du procédé permettant de représenter numériquement les données complexes telles que les images, vidéos, textes. . .
- Partant d'objets complexes comme une image par exemple, il s'agit d'extraire des variables permettant de représenter ceux-ci au travers d'un vecteur ("features extraction") par exemple. Plus généralement on parle d'apprentissage de représentation ("representation learning").
- Ces procédés sont à la base les champs d'expertises d'autres domaines que sont l'analyse d'images, le TAL. . . Il existe dans ce cas des outils pouvant être utilisés même par des non experts. Toutefois, les réseaux de neurones et le "deep learning" sont des méthodes assez génériques pouvant s'adapter à différents types de données.
- Notre point de départ sera nécessairement soit une matrice de données de type table comme présenté précédemment soit une matrice carrée de dissimilarités ou de similarités entre objets (que nous étudierons ultérieurement comme pour le cas des svm).

Schéma général

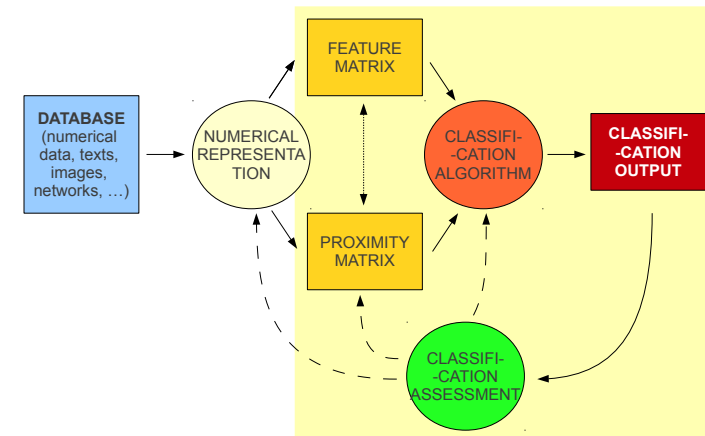


Schéma général (suite)

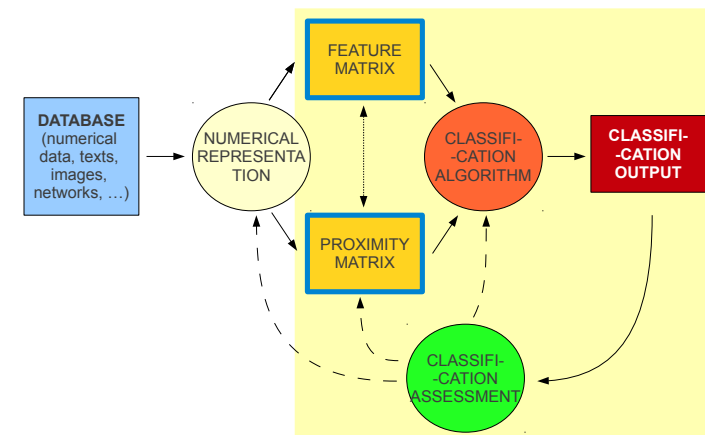


Schéma général (suite)

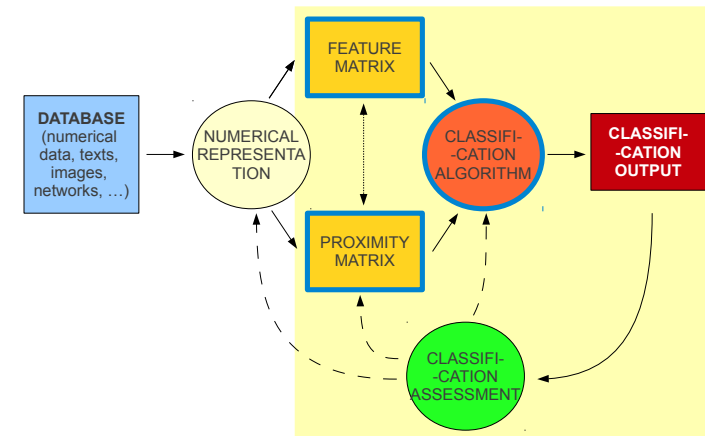
- Dans ce qui suit nous présentons des exemples simples de régression et de catégorisation qui sont à vocation pédagogique.
- Nous présentons également quelques méthodes relativement simples qui nous permettront de mettre en lumière certains concepts et sous-problèmes traités en apprentissage supervisé.

Rappel du Sommaire

1 Introduction

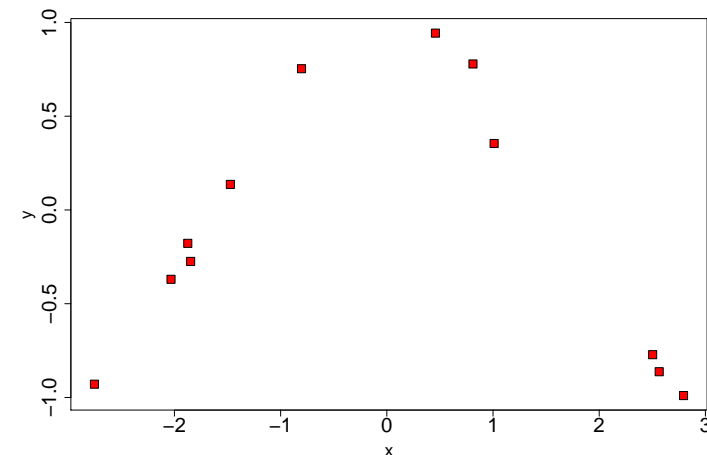
- L'apprentissage automatique
- Quelques méthodes simples en guise d'illustration
- Différentes caractéristiques des méthodes d'apprentissage supervisé
- (Quelques) Problèmes théoriques en apprentissage automatique
- Evaluation et comparaison de modèles en apprentissage supervisé

Schéma général (suite)



Exemple de problème de régression

- L'objectif est de déterminer une fonction f qui étant donné un nouveau $x \in \mathbb{R}$ prédise correctement $y \in \mathbb{R}$



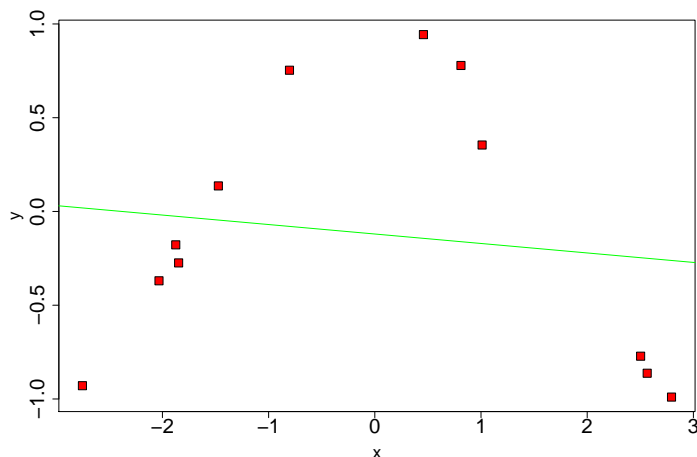
Régression linéaire simple

- Nous observons 12 couples de données avec en abscisse la variable X et en ordonnées la variable cible Y dont les éléments sont des réels.
- L'objectif est d'estimer une fonction $Y = f(X) + \epsilon$ qui représente la relation entre Y et X afin de prédire la valeur $\hat{y} = \hat{f}(\mathbf{x})$, $\forall \mathbf{x} \in \mathbb{X}$.
- Pour un problème de régression on parlera également de **prédicteur** pour la fonction \hat{f} .
- En statistique une méthode très classique est donnée par les **Moindres Carrés Ordinaires (MCO)** que l'on notera par $scr(f)$ (Somme des Carrés des Résidus ou "Residual Sum of Squares" ou "Sum of Squared Errors") :

$$\begin{aligned} scr(f) &= \sum_{i=1}^n (y_i - f(\mathbf{x}_i))^2 \\ &= \sum_{i=1}^n (\epsilon_i)^2 \end{aligned}$$

Régression linéaire simple (suite)

- Régression linéaire simple



Régression linéaire simple (suite)

- La régression linéaire simple consiste à prendre pour hypothèse que la relation f est un polynôme de degré 1 de X : $f(X) = a + bX$
- Ce qui nous donne :

$$scr(f) = scr(a, b) = \sum_{i=1}^n (y_i - (a + bx_i))^2$$
- $\mathbb{P} = \{a, b\}$ est l'ensemble des paramètres du modèle et on cherche les estimations \hat{a} et \hat{b} qui minimisent scr .
- Il faut déterminer les points critiques (ou stationnaires), solutions des équations normales (dérivées premières nulles). On obtient une solution analytique :

$$\hat{a} = \bar{y} - \hat{b}\bar{x} \text{ et } \hat{b} = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^n (x_i - \bar{x})^2}$$

où $\bar{y} = \frac{1}{n} \sum_{i=1}^n y_i$ est la moyenne empirique de Y .

- Le modèle de prédiction est alors donné par :

$$\hat{f}(\mathbf{x}) = \hat{a} + \hat{b}\mathbf{x}$$

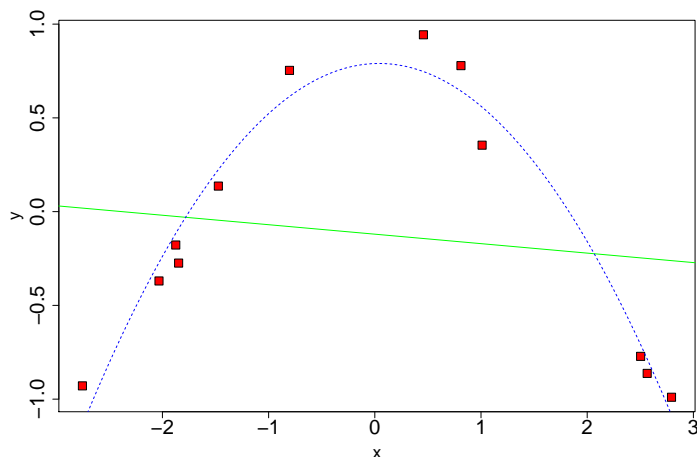
Régression linéaire multiple (polynôme de degré > 1)

- La régression linéaire simple fait l'hypothèse que la fonction f est un polynôme de degré 1 et clairement ceci n'est pas une hypothèse raisonnable pour l'exemple traité.
- Autre type d'hypothèse : f est un polynôme de degré 2 de X : $f(X) = a + bX + cX^2$
- Dans ce cas $\mathbb{P} = \{a, b, c\}$ et on cherche à minimiser :

$$scr(f) = scr(a, b, c) = \sum_{i=1}^n (y_i - (a + bx_i + cx_i^2))^2$$
- Remarque : on parle de modèle linéaire car f est une fonction linéaire des **paramètres** \mathbb{P} ! Les variables peuvent être tout type de fonction des variables initiales.

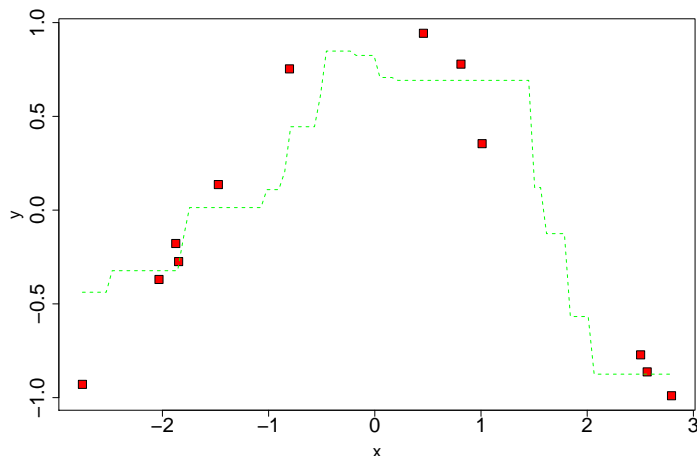
Régression linéaire multiple

- Régression linéaire multiple utilisant des fonctions de base polynômiales (jusqu'au degré 2).



Régression non paramétrique (suite)

- Avec $\lambda = 1$



Régression non paramétrique

- La régression linéaire est un **modèle paramétrique** : on choisit une famille de fonctions avec un nombre fini de paramètres (\mathbb{P}) et le problème revient alors à estimer les paramètres qui minimisent $scr(\mathbb{P})$.
- Il existe des **modèles non paramétriques** de régression. Dans ce cas une hypothèse courante est basée sur les **“plus proches voisins”** : “deux objets similaires doivent avoir deux valeurs cibles similaires”.
- La méthode la plus simple dans ce cas consiste à moyenner les y_i des \mathbf{x}_i proches de \mathbf{x} . Formellement il s'agit d'étendre la méthode des moyennes mobiles et on obtient l'estimateur suivant :

$$\hat{f}(\mathbf{x}) = \frac{\sum_{i=1}^n K_\lambda(\mathbf{x}, \mathbf{x}_i) y_i}{\sum_{i=1}^n K_\lambda(\mathbf{x}, \mathbf{x}_i)}$$

où, pour notre exemple ci-dessous, $K_\lambda(\mathbf{x}, \mathbf{x}_i)$ vaut 1 si $\|\mathbf{x} - \mathbf{x}_i\| < \lambda$ et 0 sinon (boule centrée en \mathbf{x} et de rayon λ).

Régression non paramétrique (suite)

- On peut réécrire \hat{f} de la façon équivalente suivante :

$$\hat{f}(\mathbf{x}) = \sum_{i=1}^n \left(\frac{K_\lambda(\mathbf{x}, \mathbf{x}_i)}{\sum_{i=1}^n K_\lambda(\mathbf{x}, \mathbf{x}_i)} \right) y_i$$

On donne un poids uniforme non nul pour tout y_i dont le \mathbf{x}_i appartient au voisinage de \mathbf{x} .

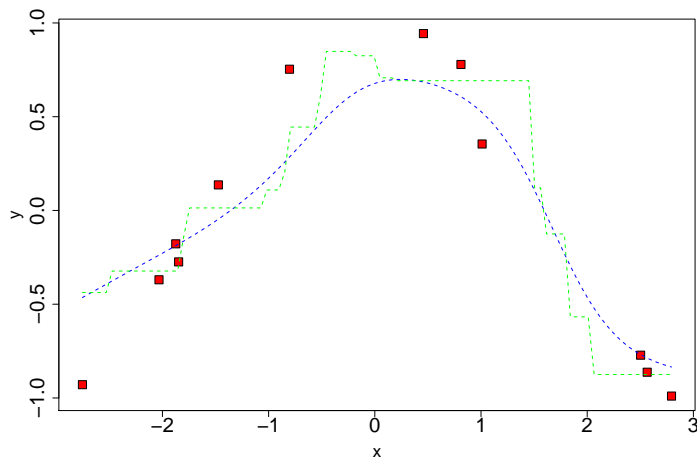
- Les **estimateurs à noyau** généralisent la méthode précédente en donnant des poids différents aux plus proches voisins \mathbf{x}_i de \mathbf{x} selon la distance entre \mathbf{x}_i et \mathbf{x} . La fonction K_λ est de manière générale appelée fonction noyau (ou noyau de Parzen).
- Exemple du noyau gaussien :

$$K_\lambda(\mathbf{x}, \mathbf{x}_i) = \frac{1}{\lambda\sqrt{2\pi}} \exp\left(-\frac{1}{2} \left(\frac{\mathbf{x} - \mathbf{x}_i}{\lambda}\right)^2\right)$$

- Pour toute fonction noyau, λ est un paramètre important qui permet de préciser la notion de voisinage autour de \mathbf{x} .

Régression non paramétrique (suite)

- Avec noyau gaussien et $\lambda = 1$.



Régression linéaire multiple avec variables artificielles

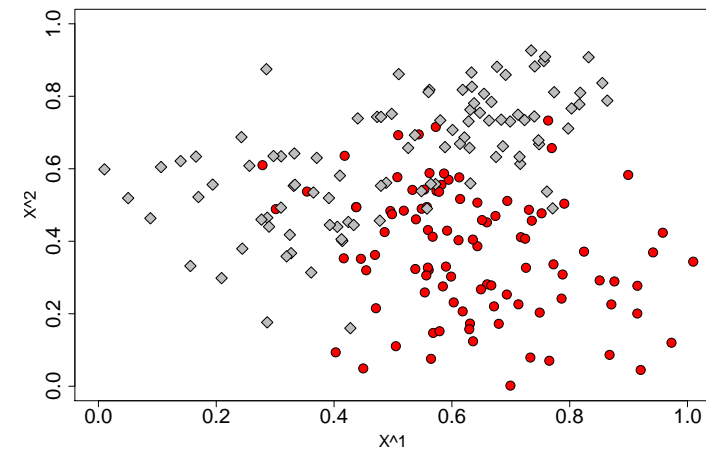
- On attribue des valeurs numériques à chacune des deux classes comme par exemple $C_1 \leftrightarrow 1$ et $C_2 \leftrightarrow -1$.
- On remplace Y variable discrète par Z une variable numérique remplie de -1 et 1 .
- On traite le problème comme une régression linéaire multiple : $Z = g(X)$.
- Pour un nouveau \mathbf{x} on applique la règle de décision suivante :

$$\hat{f}(\mathbf{x}) = \begin{cases} C_1 & \text{si } \hat{g}(\mathbf{x}) \geq 0 \\ C_2 & \text{si } \hat{g}(\mathbf{x}) < 0 \end{cases}$$

- La ligne de niveau $\{\mathbf{x} \in \mathbb{R}^2 : \hat{g}(\mathbf{x}) = 0\}$ est la **frontière de décision**.
- Pour un problème de catégorisation on parlera également de **classifieur** pour la fonction \hat{f} .

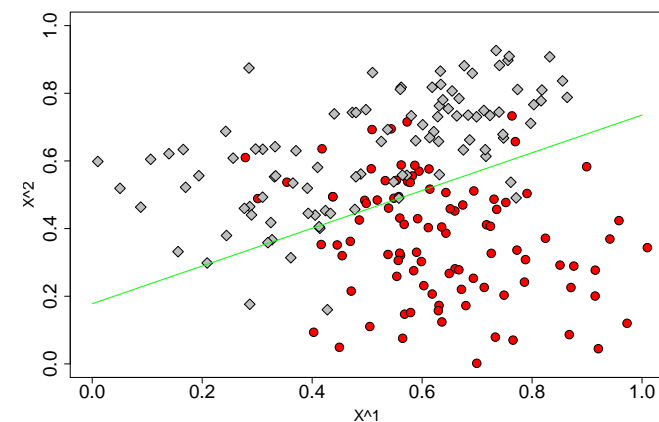
Exemple de problème de catégorisation

- L'objectif est de déterminer une fonction \hat{f} qui étant donné un nouveau $\mathbf{x} \in \mathbb{R}^2$ prédit correctement sa classe $y \in \{C_1, C_2\}$



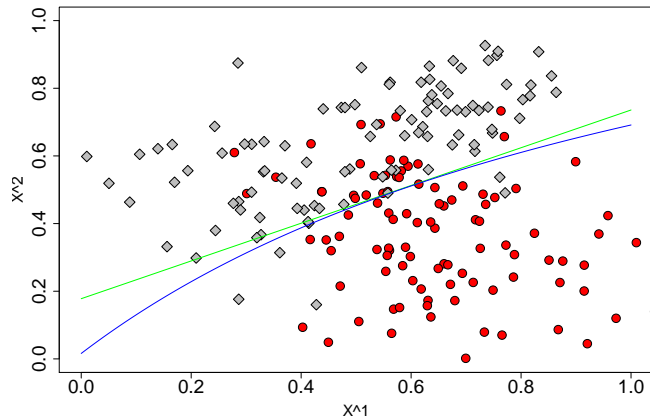
Régression linéaire multiple (suite)

- Hypothèse : $Z = g(X) = a + bX^1 + cX^2$ (polynôme de degré 1 des $\{X^j\}_{j=1}^2$)
- En vert on a tracé la frontière de décision.



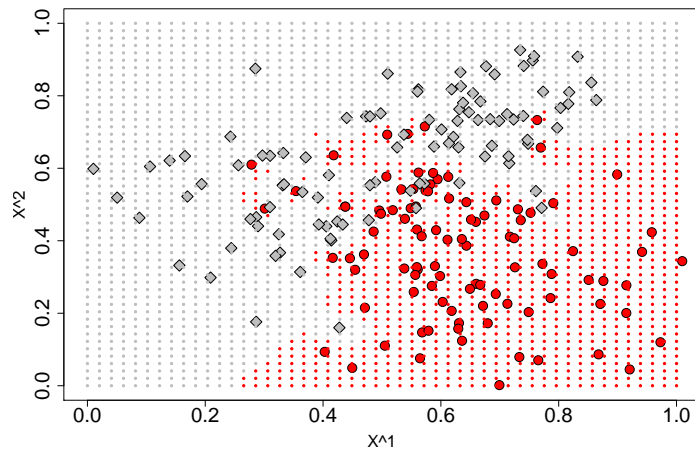
Régression linéaire multiple (suite)

- Hypothèse : $Z = g(X) = a + bX^1 + cX^2 + dX^1X^2$ (polynôme de degré 2 des $\{X^j\}_{j=1}^2$).
- En bleu on a tracé la frontière de décision.



Méthode des k -ppv (suite)

- $k=1$



Méthode des k plus proches voisins (k -ppv)

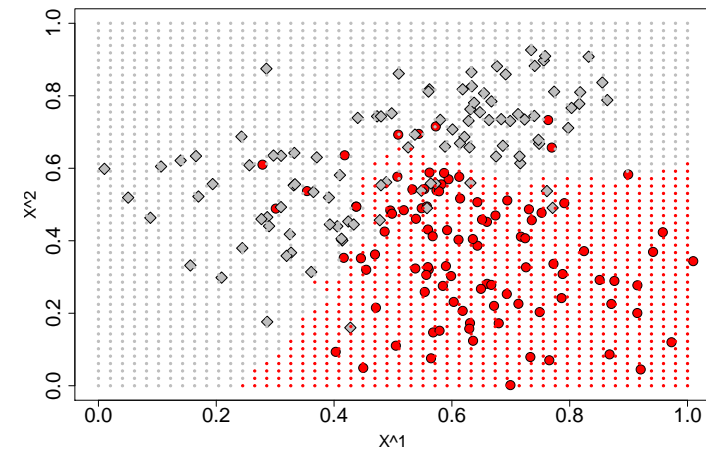
- Nous avons utilisé la régression linéaire associée à des variables artificielles pour un problème de catégorisation.
- Nous voyons une autre approche simple qui est un modèle non paramétrique : les k **plus proches voisins**.
- Etant donné un nouvel objet \mathbf{x} , la méthode consiste à déterminer les k plus proches objets (annotés) et d'effectuer un vote à la majorité relative afin de déterminer la classe de \mathbf{x} .
- Formellement nous avons la fonction de prédiction suivante :

$$\hat{f}(\mathbf{x}) = \arg \max_{C_i \in Y} \frac{|\{\mathbf{x}_i \in \mathbb{V}_k(\mathbf{x}) : y_i = C_i\}|}{k}$$

où $\mathbb{V}_k(\mathbf{x})$ est l'ensemble des k plus proches \mathbf{x}_i de \mathbf{x} et $\frac{|\{\mathbf{x}_i \in \mathbb{V}_k(\mathbf{x}) : y_i = C_i\}|}{k}$ est la proportion d'objets appartenant à la classe C_i parmi les k plus proches voisins.

Méthode des k -ppv (suite)

- $k=9$



Rappel du Sommaire

1 Introduction

- L'apprentissage automatique
- Quelques méthodes simples en guise d'illustration
- Différentes caractéristiques des méthodes d'apprentissage supervisé
- (Quelques) Problèmes théoriques en apprentissage automatique
- Evaluation et comparaison de modèles en apprentissage supervisé

Plusieurs modèles de prédiction

- Il existe plusieurs façons d'établir par apprentissage la relation entre les x_i et les y_i d'entraînement afin de prédire la valeur cible $\forall \mathbf{x} \in \mathbb{X}$. On peut distinguer :
 - ▶ Les méthodes de type **inductif** qui infèrent des données d'apprentissage une fonction de prédiction **globale** qui est estimée en optimisant un critère de performance. Les prédictions pour des nouvelles données sont déduites de la fonction de décision estimée. Il s'agit notamment des modèles paramétriques vus précédemment. Les dénominations anglo-saxonnes sont "inductive learning", "eager learning".
 - ▶ Les méthodes de type **transductif** qui ne passent pas par une phase d'inférence mais qui utilisent directement et **localement** les données d'apprentissage pour prédire la valeur cible pour les nouvelles données. Il s'agit notamment des modèles non paramétriques vus précédemment. Les dénominations anglo-saxonnes sont "transductive learning", "lazy learning" ou "instance-based learning".
- Nous verrons essentiellement des méthodes de type inductif et celles-ci se distinguent les unes des autres en considérant plusieurs aspects que nous voyons ci-après.

Choix de la méthode d'apprentissage

- Il existe plusieurs méthodes en apprentissage supervisé que ce soit pour la régression ou la catégorisation.
- Pour choisir une méthode, il y a deux approches complémentaires :
 - ▶ l'une, théorique, relève de la **bonne compréhension des fondements des méthodes** et de ce qui permet de les distinguer afin de déterminer les modèles qui traiteraient au mieux un cas d'étude donné,
 - ▶ l'autre, empirique, relève de l'**application de méthodes et critères d'évaluation et de sélection** permettant de sélectionner les algorithmes de catégorisation les plus performants pour le cas à l'étude.
- Nous aborderons respectivement ces deux approches en :
 - ▶ étudiant les différents outils et hypothèses mathématiques qui fondent chaque méthode,
 - ▶ étudiant les mesures d'évaluations des méthodes et le protocole expérimental conduisant à la sélection d'une bonne méthode.

Plusieurs types de fonction de prédiction

- Les méthodes de type inductif supposent que la fonction de prédiction f appartient à une famille de fonctions ou d'**hypothèses** \mathbb{H} dont les paramètres sont dénotés par \mathbb{P} .
- Par exemple la régression linéaire multiple :

$$\mathbb{H} = \{f : \mathbb{X} \rightarrow \mathbb{Y} : f(X) = a_0 + \sum_{j=1}^p a_j X^j\}$$
 où

$$\mathbb{P} = \{a_0, \dots, a_p\} \in \mathbb{R}^{p+1}.$$
- Le modèle linéaire dans sa forme générale peut s'écrire comme suit :

$$f(X) = a_0 + \sum_{m=1}^M a_m g_m(X)$$

- où $g_m : \mathbb{X} \rightarrow \mathbb{R}$ sont des fonctions quelconques à valeurs dans \mathbb{R} appelées **fonctions ou expansions de base** (par exemple : $f(X) = a_0 + a_1 X^1 + a_2 X^2 + a_3 X^1 X^2$).
- Il existe donc plusieurs familles d'hypothèses donnant autant de résultats de prédictions différents.

Biais inductif

- Le choix d'un espace d'hypothèses \mathbb{H} implique un **biais inductif** dans le processus d'apprentissage.
- En effet, il existe par exemple une infinité de façon de séparer les classes C_1 et C_2 dans l'exemple de catégorisation.
- Si on choisit la régression linéaire multiple, la forme de la fonction de prédiction est nécessairement un hyperplan.
- En d'autres termes, le biais inductif est l'ensemble des hypothèses implicites que l'on fait lorsque l'on utilise une méthode d'apprentissage supervisé pour résoudre un problème de régression ou de catégorisation.

Notations

Nous nous plaçons dans un **cadre probabiliste** et dans ce cas :

- Les variables X^1, \dots, X^p sont des variables aléatoires (v.a.) réelles.
- La variable Y est une v.a. à valeurs dans \mathbb{Y} .
- Les objets X_1, \dots, X_n et X sont des vecteurs aléatoires de dimension $(p \times 1)$, chaque dimension j étant associée à la variable X^j .
- On notera par $P(X)$ la fonction de densité de probabilité (multidimensionnelle) du vecteur aléatoire X .
- On notera par $P(Y|X)$ la probabilité conditionnelle de Y sachant X .
- On notera par $E_X(f(X))$ l'espérance de $f(X)$ par rapport à X .
- On notera par $E_{Y|X}(f(Y)|X)$ l'espérance conditionnelle de $f(Y)$ par rapport à Y sachant X .
- $\{X_i^j\}_{i=1}^n$ et $\{Y_i\}_{i=1}^n$ sont n variables aléatoires que l'on supposera i.i.d. (indépendantes et identiquement distribuées) selon les lois mères $P(X^j)$ et $P(Y)$ respectivement.

Plusieurs types de fonction de performance

- Etant donné un espace d'hypothèses \mathbb{H} , pour déterminer la fonction de prédiction (une instance de \mathbb{H}), il faut estimer les paramètres \mathbb{P} qui optimisent un critère de performance sur les données \mathbb{E} .
- Pour le problème de régression nous avons déjà évoqué les MCO :

$$scr(f) = \sum_{i=1}^n (y_i - f(\mathbf{x}_i))^2$$

- Lorsque les données n'ont pas toutes une importance uniforme, une façon de généraliser le scr est l'utilisation de concepts issus de la **décision statistique**.

Fonction de performance et décision statistique

- X est un vecteur aléatoire de taille $(p \times 1)$ à valeurs dans \mathbb{X} .
- Y est une variable aléatoire à valeurs dans \mathbb{Y} .
- Soient $P(X)$, $P(Y)$, les fonctions de probabilité de X et Y .
- Soit $P(X, Y)$ la fonction de probabilité jointe du couple (X, Y) .
- Soit $\ell(f(X), Y)$ une **fonction de perte** ("loss") mesurant le coût de la différence entre la prédiction du modèle et l'observation.

Définition. (Espérance de la fonction de perte)

On définit l'espérance de la fonction perte associée à f de la façon suivante :

$$E_{X,Y}(\ell(f(X), Y)) = E_{X,Y}(\ell) = \int_{\mathbb{X}} \int_{\mathbb{Y}} \ell(f(\mathbf{x}), y) P(\mathbf{x}, y) d\mathbf{x} dy$$

Fonction de performance et décision statistique (suite)

- **En théorie**, on cherche donc f qui minimise l'espérance de la fonction de perte ℓ (**en pratique**, nous ne connaissons ni $P(X, Y)$, ni $P(Y|X)$ et nous ne disposons que des observations dans \mathbb{E}).
- La régression linéaire multiple par MCO est un cas particulier puisqu'il s'agit de minimiser $E_{X,Y}(\ell)$ en prenant :
 - ▶ une fonction de perte quadratique : $\ell_2(f(X), Y) = (Y - f(X))^2$,
 - ▶ une distribution uniforme pour le couple (X, Y) ,
 - ▶ une fonction de prédiction polynomiale de degré 1 des $\{X^j\}_{j=1}^p$:

$$f(X) = a_0 + \sum_{j=1}^p a_j X^j$$

- On peut en théorie généraliser et utiliser d'autres types de fonction de perte ou en donnant différents poids selon $P(X, Y)$.

Fonction de perte quadratique et fonction de régression (suite)

- $f(\mathbf{x})$ étant un élément de \mathbb{Y} on peut alors écrire :

$$\forall \mathbf{x} \in \mathbb{X} : \min_{z \in \mathbb{Y}} E_{Y|X}((z - y)^2 | X = \mathbf{x}) = \int_{\mathbb{Y}} (z - y)^2 P(y|X = \mathbf{x}) dy$$

- Pour résoudre ces problèmes, on cherche les points critiques :

$$\frac{\partial}{\partial z} E_{Y|X}((z - y)^2 | X = \mathbf{x}) = 0$$

- On obtient alors $\forall \mathbf{x} \in \mathbb{X}$ les équations normales suivantes :

$$\begin{aligned} 2 \int_{\mathbb{Y}} (z - y) P(y|X = \mathbf{x}) dy &= 0 \\ \Leftrightarrow \int_{\mathbb{Y}} z P(y|X = \mathbf{x}) dy &= \int_{\mathbb{Y}} y P(y|X = \mathbf{x}) dy \\ \Leftrightarrow z &= \int_{\mathbb{Y}} y P(y|X = \mathbf{x}) dy \\ \Leftrightarrow z &= E_{Y|X}(Y|X = \mathbf{x}) \end{aligned}$$

Fonction de perte quadratique et fonction de régression

- Etudions en **théorie** (car en pratique on ne connaît pas $P(X, Y)$) $E_{X,Y}(\ell)$ dans le cas de la fonction de perte quadratique :

$$\min_f E_{X,Y}(\ell_2) = \int_{\mathbb{X}} \int_{\mathbb{Y}} (f(\mathbf{x}) - y)^2 P(\mathbf{x}, y) d\mathbf{x} dy$$

- En écrivant $P(X, Y) = P(Y|X)P(X)$ on peut réécrire :

$$E_{X,Y}(\ell_2) = E_X(E_{Y|X}(\ell_2|X))$$

- On a le problème équivalent :

$$\min_f E_X(E_{Y|X}(\ell_2|X)) = \int_{\mathbb{X}} \left(\int_{\mathbb{Y}} (f(\mathbf{x}) - y)^2 P(y|\mathbf{x}) dy \right) P(\mathbf{x}) d\mathbf{x}$$

- Pour minimiser $E_{X,Y}(\ell_2)$ il suffit de résoudre le problème suivant **en chaque point $\mathbf{x} \in \mathbb{X}$** :

$$\min_f E_{Y|X}(\ell_2|X = \mathbf{x}) = \int_{\mathbb{Y}} (f(\mathbf{x}) - y)^2 P(y|\mathbf{x}) dy$$

Fonction de perte quadratique et fonction de régression (suite)

- En somme, nous obtenons la solution générique suivante que l'on appelle **fonction de régression** :

$$f^*(\mathbf{x}) = E_{Y|X}(Y|X = \mathbf{x})$$

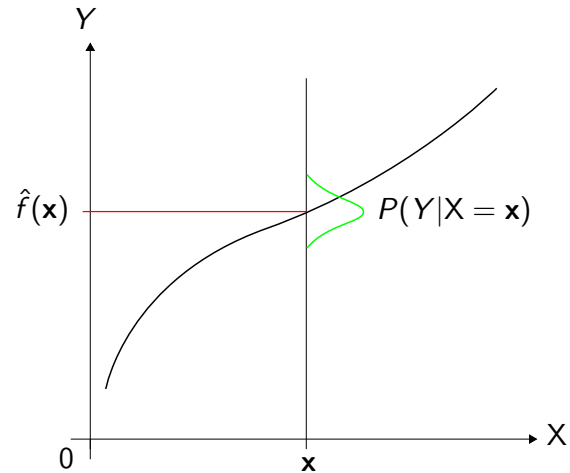
- Ainsi la fonction qui minimise en \mathbf{x} l'espérance de la fonction de perte quadratique, $E_{X,Y}(\ell_2)$, c'est l'espérance de Y sous la probabilité conditionnelle $P(Y|X = \mathbf{x})$.
- La fonction de perte quadratique est un sous-cas de la famille de fonction de perte suivante dite de Minkowski :

$$E_{X,Y}(\ell_r(f(X), Y)) = \int_{\mathbb{X}} \int_{\mathbb{Y}} |f(\mathbf{x}) - y|^r P(\mathbf{x}, y) d\mathbf{x} dy$$

- Le cas de ℓ_2 est souvent utilisé car elle conduit à la solution simple que nous venons de voir mais le principe d'espérance de fonction de perte permet d'avoir plusieurs types de fonction de performance.

Fonction de perte quadratique et fonction de régression (suite)

- Cas de la fonction quadratique et illustration de la fonction de régression.



Fonction de performance pour la catégorisation (suite)

- L'espérance de perte s'écrit alors :

$$E_{X,Y}(\mathbf{L}) = \int_{\mathbb{X}} \sum_{C_l \in \mathbb{Y}} \mathbf{L}(C_l, f(\mathbf{x})) P(\mathbf{x}, C_l) d\mathbf{x}$$

- Comme précédemment, on peut considérer l'espérance conditionnelle et obtenir l'écriture équivalente suivante :

$$E_X (E_{Y|X}(\mathbf{L}|X))$$

- Puis, pour minimiser l'espérance de la fonction de perte, il suffit de minimiser **en chaque point** $\mathbf{x} \in \mathbb{X}$ le problème suivant :

$$\min_f E_{Y|X}(\mathbf{L}|X = \mathbf{x})$$

- La solution est alors :

$$\forall \mathbf{x} \in \mathbb{X} : f^*(\mathbf{x}) = \arg \min_{C_{l'} \in \mathbb{Y}} \sum_{C_l \in \mathbb{Y}} \mathbf{L}(C_l, C_{l'}) P(C_l | X = \mathbf{x})$$

Fonction de performance pour la catégorisation

- Que se passe-t-il dans le cas où \mathbb{Y} est discret ?
- Le principe de minimisation de l'espérance de la fonction de perte est valide mais il faut adapter la fonction de perte au cas discret.
- Un coût de la fonction de perte intervient lorsqu'on attribue à un \mathbf{x} une classe qui n'est pas la bonne.
- Supposons que la classe de \mathbf{x} est C_l et qu'on lui attribue par erreur la classe $C_{l'}$. Pour chaque couple $(C_l, C_{l'})$ on a le coût $\mathbf{L}(C_l, C_{l'})$ associé à une mauvaise catégorisation.
- On a la donnée d'une matrice de perte \mathbf{L} de taille $(q \times q)$ (q étant le cardinal de \mathbb{Y} c-à-d le nombre de classes) dont le terme général est :

$$\mathbf{L}(C_l, C_{l'}) = \mathbf{L}_{ll'} = \text{Coût associée à une mauvaise affectation d'un objet de classe } C_l \text{ à une classe } C_{l'}$$

- \mathbf{L} est d'éléments positifs ou nuls et la diagonale est remplie de 0.

Fonction de perte binaire et classifieur bayésien

- Pour des données discrètes, la fonction de perte la plus simple est celle associée à la matrice de perte uniforme suivante :

$$\mathbf{L}_{ll'} = \begin{cases} 1 & \text{si } l \neq l' \\ 0 & \text{si } l = l' \end{cases}$$

- Dans ce cas, nous avons :

$$\begin{aligned} \forall \mathbf{x} \in \mathbb{X} : f^*(\mathbf{x}) &= \arg \min_{C_{l'} \in \mathbb{Y}} \sum_{C_l \in \mathbb{Y}} \mathbf{L}_{ll'} P(C_l | X = \mathbf{x}) \\ &= \arg \min_{C_{l'} \in \mathbb{Y}} (1 - P(C_{l'} | X = \mathbf{x})) \\ &= \arg \max_{C_{l'} \in \mathbb{Y}} P(C_{l'} | X = \mathbf{x}) \end{aligned}$$

- Ainsi, la fonction de prédiction est telle que $f^*(\mathbf{x}) = C_l$ ssi $P(C_l | X = \mathbf{x}) = \max_{C_{l'} \in \mathbb{Y}} P(C_{l'} | X = \mathbf{x})$. Cette approche est appelée **classifieur bayésien**.

Classifieur bayésien

- Si on suppose une matrice de perte uniforme et qu'on minimise l'espérance de la perte sous $P(X, Y)$ alors on obtient le classifieur bayésien qui repose sur la probabilité conditionnelle $P(Y|X)$.
- Si on applique le théorème de Bayes on a :

$$\underbrace{P(Y|X)}_{\text{posterior}} = \frac{\overbrace{P(Y)}^{\text{prior}} \overbrace{P(X|Y)}^{\text{likelihood}}}{\underbrace{P(X)}_{\text{evidence}}}$$

- Rappelons que $X = (X^1, \dots, X^p)$ est un vecteur aléatoire de dimension p . En utilisant successivement les probabilités conditionnelles ($P(A, B) = P(A|B)P(B)$) on a :

$$\begin{aligned} P(X|Y) &= P(X^1, \dots, X^p|Y) \\ &= P(X^1|Y, X^2, \dots, X^p)P(X^2, \dots, X^p|Y) \\ &= P(X^1|Y, X^2, \dots, X^p) \dots P(X^{p-1}|Y, X^p)P(X^p|Y) \end{aligned}$$

Classifieur bayésien naïf (suite)

- Il est "naïf" de supposer l'indépendance entre les variables mais ce modèle probabiliste est simple à estimer.
- On peut supposer d'autres modèles de dépendance pour $P(X^1, \dots, X^p|Y)$. Une approche consiste à modéliser les relations de dépendance par le biais de graphes. On parle alors de **modèles graphiques** (ou de **réseaux bayésiens**).

Classifieur bayésien naïf

- Si l'on suppose de plus que les v.a. X^1, \dots, X^p sont mutuellement indépendantes ($P(X^j|X^k) = P(X^j)$) on a :

$$P(X|Y) = P(X^1|Y)P(X^2|Y) \dots P(X^{p-1}|Y)P(X^p|Y)$$

- Il s'agit alors du **classifieur bayésien naïf**. Dans ce cas il suffit d'estimer à partir de \mathbb{E} les probabilités suivantes pour chaque $C_l \in \mathbb{Y}$:
 - ▶ La probabilité a priori : $P(C_l)$.
 - ▶ Les probabilités conditionnelles : $P(X^1|Y = C_l), \dots, P(X^p|Y = C_l)$.
- L'estimation de $P(X = \mathbf{x})$ n'est pas nécessaire car c'est un dénominateur commun aux probabilités a posteriori de chaque classe.
- La fonction de décision pour $\mathbf{x} = (x_1, \dots, x_p)$ est $f^*(\mathbf{x}) = C_l$ ssi $P(C_l|X = \mathbf{x}) = \max_{C_{l'} \in \mathbb{Y}} P(C_{l'}|X = \mathbf{x})$ où :

$$P(C_{l'}|X = \mathbf{x}) \propto P(C_{l'})P(X^1 = x_1|C_{l'}) \dots P(X^p = x_p|C_{l'})$$

Estimation, décision, zone de rejet

- Pour les méthodes de type inductif, il y a deux phases :
 - 1 une étape d'**inférence** ou d'estimation des paramètres du modèle \mathbb{P} ,
 - 2 une étape de **décision** qui permet d'aboutir à la prédiction $\hat{f}(X)$.
- Il existe plusieurs façons de définir théoriquement $f(X)$ (espaces d'hypothèses \mathbb{H}).
- Certains modèles sont simples et conduisent à des **solutions analytiques** comme la régression linéaire multiple par MCO.
- Pour des classes d'hypothèses plus complexes on a recours à des algorithmes d'**optimisation numérique**. Il existe également plusieurs façon d'estimer les paramètres de $f(X)$ étant donné \mathbb{E} .
- Certains modèles permettent d'appréhender une incertitude de la prédiction donnée par $\hat{f}(X)$. Dans ce cas, on peut définir une **zone de rejet** dans la prise de décision et faire intervenir l'humain. Par exemple, dans le cas des k -ppv et d'une catégorisation binaire, si la classe majoritaire ne dépasse pas 60% on peut avoir une zone de rejet.

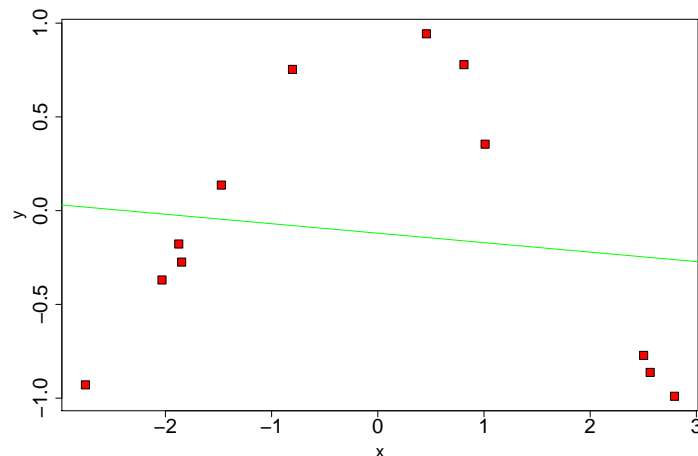
Rappel du Sommaire

1 Introduction

- L'apprentissage automatique
- Quelques méthodes simples en guise d'illustration
- Différentes caractéristiques des méthodes d'apprentissage supervisé
- (Quelques) Problèmes théoriques en apprentissage automatique
- Evaluation et comparaison de modèles en apprentissage supervisé

Généralisation, sous et sur-apprentissage (suite)

- Exemple de sous-apprentissage (\mathbb{H} =Ensemble des polynômes de degré 1 de X).

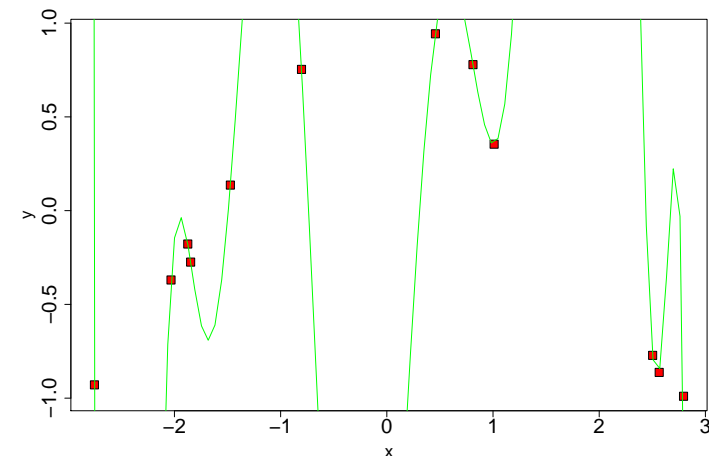


Généralisation, sous et sur-apprentissage

- Le choix d'une méthode revient à choisir un espace d'hypothèses, une fonction de perte et une technique d'inférence.
- Ce qu'on attend d'une bonne méthode n'est pas tant sa capacité à reproduire à l'identique le résultat des données d'entraînement mais de produire les résultats corrects sur des données de test c'est-à-dire non observées : c'est le principe de **généralisation**.
- Dans cette perspective il faut une bonne adéquation entre la complexité de la classe d'hypothèse choisie \mathbb{H} et la véritable relation entre X et Y . Si la complexité de \mathbb{H} n'est pas assez suffisante on parle de **sous-apprentissage**.
- Quand au contraire, la complexité de \mathbb{H} est trop grande, il arrive que l'erreur sur \mathbb{E} est proche de zéro mais l'erreur sur les données de test est grande. Dans ce cas on parle de **sur-apprentissage**.

Généralisation, sous et sur-apprentissage (suite)

- Exemple de sur-apprentissage (\mathbb{H} =Ensemble des polynôme de degré 12 de X).



Complexité des modèles de régression linéaire

- Dans l'exemple de régression, la complexité d'un modèle ou d'une classe d'hypothèses \mathbb{H} est l'ordre du polynôme.
- Si l'ordre est trop petit, il y a sous-apprentissage et s'il est trop grand il y a sur-apprentissage.
- Pour le polynôme de degré 1 :
 - ▶ la complexité des données et celle du modèle ne coïncident pas,
 - ▶ l'erreur mesurée sur les données \mathbb{E} est très grande,
 - ▶ mais la fonction de prédiction étant une droite la variance du modèle est faible (si on change \mathbb{E} la "droite changera peu").
- Pour le polynôme de degré 12 :
 - ▶ la complexité des données et celle du modèle ne coïncident pas,
 - ▶ l'erreur mesurée sur les données \mathbb{E} est très faible,
 - ▶ mais la fonction de prédiction est instable et donc la variance du modèle est très grande (si on change \mathbb{E} la "courbe changera beaucoup").

Arbitrage biais-variance

- Etant donné X , l'espérance de l'erreur de prédiction avec une **fonction de perte quadratique** peut se décomposer comme suit :

$$\begin{aligned}
 \mathbb{E}_{Y|X}((f(X) - Y)^2|X) &= \mathbb{E}_{Y|X}((f(X) - \mathbb{E}_{Y|X}(Y|X) + \mathbb{E}_{Y|X}(Y|X) - Y)^2|X) \\
 &= \mathbb{E}_{Y|X}(\underbrace{[f(X) - \mathbb{E}_{Y|X}(Y|X)]}_{\text{Erreur quadratique}} + \underbrace{[\mathbb{E}_{Y|X}(Y|X) - Y]}_{\text{Bruit irréductible}})^2|X) \\
 &= \mathbb{E}_{Y|X}(\underbrace{[f(X) - \mathbb{E}_{Y|X}(Y|X)]^2}_{\text{Erreur quadratique}} + \underbrace{[\mathbb{E}_{Y|X}(Y|X) - Y]^2}_{\text{Bruit irréductible}} \\
 &\quad + 2 \underbrace{[f(X) - \mathbb{E}_{Y|X}(Y|X)]}_{\text{Erreur quadratique}} \underbrace{[\mathbb{E}_{Y|X}(Y|X) - Y]}_{\text{Bruit irréductible}}|X) \\
 &= \mathbb{E}_{Y|X}([f(X) - \mathbb{E}_{Y|X}(Y|X)]^2|X) + \mathbb{E}_{Y|X}([\mathbb{E}_{Y|X}(Y|X) - Y]^2|X) \\
 &\quad + 2\mathbb{E}_{Y|X}([f(X) - \mathbb{E}_{Y|X}(Y|X)] [\mathbb{E}_{Y|X}(Y|X) - Y]|X) \\
 &= \mathbb{E}_{Y|X}([f(X) - \mathbb{E}_{Y|X}(Y|X)]^2|X) + \mathbb{E}_{Y|X}([\mathbb{E}_{Y|X}(Y|X) - Y]^2|X)
 \end{aligned}$$

- Car en effet, la double somme vaut 0 :

$$\begin{aligned}
 \mathbb{E}_{Y|X}([f(X) - \mathbb{E}_{Y|X}(Y|X)] [\mathbb{E}_{Y|X}(Y|X) - Y]|X) \\
 &= \mathbb{E}_{Y|X}(f(X)\mathbb{E}_{Y|X}(Y|X) - f(X)Y - \mathbb{E}_{Y|X}(Y|X)^2 + \mathbb{E}_{Y|X}(Y|X)Y|X) \\
 &= f(X)\mathbb{E}_{Y|X}(Y|X) - f(X)\mathbb{E}_{Y|X}(Y|X) - \mathbb{E}_{Y|X}(Y|X)^2 + \mathbb{E}_{Y|X}(Y|X)\mathbb{E}_{Y|X}(Y|X) \\
 &= 0
 \end{aligned}$$

Rappels de probabilités

- Dans ce qui suit, nous étudions des propriétés qui font appel à des outils probabilistes. Nous rappelons quelques propriétés de linéarité de l'opérateur espérance \mathbb{E} .
- Supposons que X soit une variable aléatoire et que b et a soient deux réels alors :

$$\begin{aligned}
 \mathbb{E}_X(aX + b) &= a\mathbb{E}_X(X) + \mathbb{E}_X(b) \\
 &= a\mathbb{E}_X(X) + b
 \end{aligned}$$

- Supposons que X soit un vecteur aléatoire et que \mathbf{b} et \mathbf{A} soient respectivement un vecteur et une matrice carrée qui nous sont donnés alors :

$$\mathbb{E}_X(\mathbf{A}X + \mathbf{b}) = \mathbf{A}\mathbb{E}_X(X) + \mathbf{b}$$

Arbitrage biais-variance (suite)

Définition. (Décomposition Erreur quadratique - Bruit)

$$\begin{aligned}
 &\underbrace{\mathbb{E}_{Y|X}((f(X) - Y)^2|X)}_{\mathbb{E}_{Y|X}(\ell_2|X)} \\
 &= \underbrace{\mathbb{E}_{Y|X}([f(X) - \mathbb{E}_{Y|X}(Y|X)]^2|X)}_{\text{Erreur quadratique}} + \underbrace{\mathbb{E}_{Y|X}([\mathbb{E}_{Y|X}(Y|X) - Y]^2|X)}_{\text{Bruit irréductible}}
 \end{aligned}$$

- Le **bruit irréductible** est intrinsèque aux données (les erreurs de mesure par exemple) et le terme associé représente l'erreur minimale que l'on peut commettre en moyenne.
- L'**erreur quadratique** est l'espérance de l'erreur entre f et la fonction de regression (que l'on a vue être la fonction optimale pour la minimisation de $\mathbb{E}_{Y|X}(\ell_2|X)$ qui solutionne la minimisation de $\mathbb{E}_{X,Y}(\ell_2)$ pour tout $\mathbf{x} \in \mathbb{X}$).

Arbitrage biais-variance (suite)

- Dans ce contexte, les méthodes d'apprentissage consistent donc à approximer $E_{Y|X}(Y|X)$. Etant donné les données d'apprentissage à disposition, \mathbb{E} , on infère une fonction de prédiction $\hat{f}_{\mathbb{E}}(X) \in \mathbb{H}$.
- Si l'on **change de données d'apprentissage** on obtient une autre fonction de prédiction de \mathbb{H} . Ainsi, on peut voir les données d'entraînement comme la réalisation d'un processus aléatoire et on définit $E_{\mathbb{E}}(\hat{f}_{\mathbb{E}}(X))$: l'espérance de la fonction de prédiction dépendant du processus aléatoire générant les données \mathbb{E} .
- Etant donné un ensemble \mathbb{E} et une fonction de prédiction induite $\hat{f}_{\mathbb{E}}(X)$, on peut alors décomposer l'erreur quadratique comme suit :

$$E_{\mathbb{E}} \left(\left[\hat{f}_{\mathbb{E}}(X) - E_{Y|X}(Y|X) \right]^2 \right) = E_{\mathbb{E}} \left(\left[\hat{f}_{\mathbb{E}}(X) - E_{\mathbb{E}}(\hat{f}_{\mathbb{E}}(X)) \right]^2 \right) + E_{\mathbb{E}} \left(\left[E_{\mathbb{E}}(\hat{f}_{\mathbb{E}}(X)) - E_{Y|X}(Y|X) \right]^2 \right)$$
 Puisque comme précédemment la double somme s'annule.

Arbitrage biais-variance (suite)

- A erreur quadratique constante, on voit qu'il y a un **arbitrage entre biais et variance**.
- Exemple de fort biais et faible variance : régression linéaire avec polynôme de degré 1.
- Exemple de faible biais et forte variance : régression linéaire avec polynôme de degré 12.
- L'idéal est d'avoir un faible biais et une faible variance pour une meilleure généralisation mais plus facile à dire qu'à faire !
- Plus la complexité d'un modèle augmente plus le biais mesuré sur des données \mathbb{E} diminue. Mais la variance augmentant également, le bon comportement du modèle estimé sur des données non observées n'est alors plus garanti.

Arbitrage biais-variance (suite)

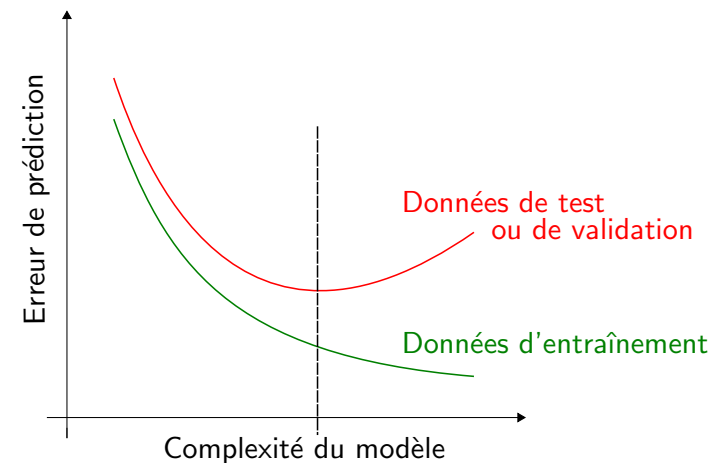
Définition. (Décomposition Biais - Variance)

$$\underbrace{E_{\mathbb{E}} \left(\left[\hat{f}_{\mathbb{E}}(X) - E_{Y|X}(Y|X) \right]^2 \right)}_{\text{Erreur quadratique}} = \underbrace{E_{\mathbb{E}} \left(\left[\hat{f}_{\mathbb{E}}(X) - E_{\mathbb{E}}(\hat{f}_{\mathbb{E}}(X)) \right]^2 \right)}_{\text{Variance}} + \underbrace{\left[E_{\mathbb{E}}(\hat{f}_{\mathbb{E}}(X)) - E_{Y|X}(Y|X) \right]^2}_{\text{Biais}^2}$$

- Le **biais** indique, l'écart entre la fonction de prédiction moyenne apprise sur plusieurs jeux de données et la fonction de régression.
- La **variance** représente en moyenne, l'écart quadratique entre une fonction de prédiction apprise sur un jeu de données et la fonction de prédiction moyenne apprise sur plusieurs jeux de données.

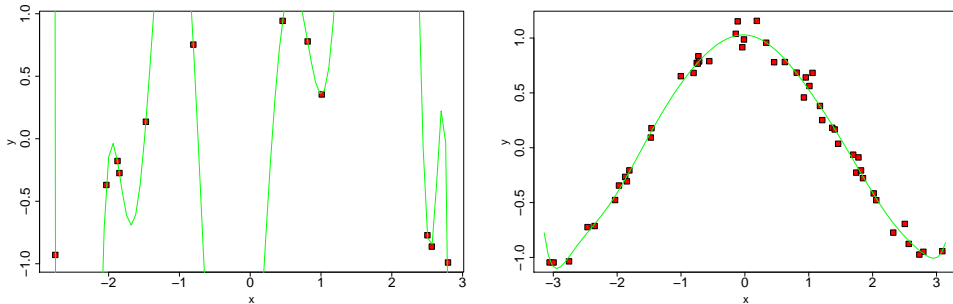
Arbitrage biais-variance (suite)

- Illustration du problème de l'arbitrage biais-variance



Impact de la taille de l'échantillon d'entraînement \mathbb{E}

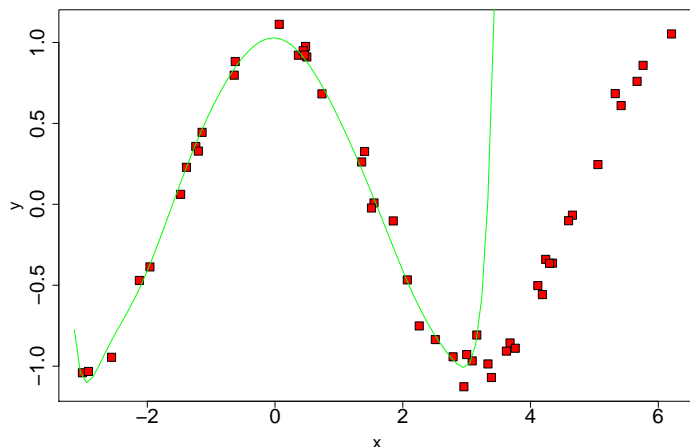
- Même tâche que précédemment : les données sont générées par la fonction \cos sur $[-\pi, \pi]$ à laquelle on ajoute un bruit $\epsilon \sim \mathcal{N}(0, 0.08)$.
- \mathbb{H} = Ensemble des polynômes de X de degré 12.
- Estimation sur deux échantillons de tailles $n = 12$ et $n = 50$.



- Bien sûr plus on a de données meilleure est l'estimation !

Arbitrage entre Complexité et Données d'entraînement (suite)

- Extrapolation du modèle appris précédemment jusqu'à 2π .



Arbitrage entre Complexité et Données d'entraînement

- Ainsi en pratique, étant donné un ensemble d'entraînement, il y a un arbitrage entre deux facteurs pour assurer une bonne généralisation de la fonction de prédiction sur des données de test :
 - ▶ La complexité de l'espace des hypothèses choisis \mathbb{H} .
 - ▶ La quantité de données d'entraînement n .
- Une grande complexité de \mathbb{H} permet une meilleure flexibilité du modèle et implique une meilleure généralisation.
- Mais une trop grande complexité donne parfois trop de flexibilité : sur \mathbb{E} l'erreur diminue mais la variance du modèle sera plus forte. Ainsi, si les données de test sortent de la région des données \mathbb{E} , le comportement de la fonction de prédiction risque d'être chaotique.
- Ce problème est moins fort lorsque n est grand comme on vient de le voir mais jusqu'à un certain point.

Dimension de Vapnik-Chervonenkis

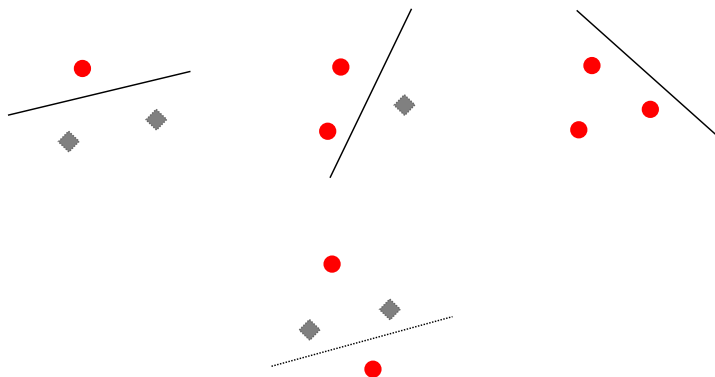
- On a parlé de complexité des modèles linéaires. De manière plus générale, la notion de complexité d'un espace \mathbb{H} peut être appréhendée par le concept de **dimension de Vapnik-Chervonenkis**.
- Considérons un problème de catégorisation binaire. Soit \mathbb{E} un ensemble d'apprentissage de n points. Il y a 2^n façons différentes d'attribuer l'une ou l'autre classe à ces n points.
- Si pour chacune de ces 2^n configurations, il existe $h \in \mathbb{H}$ qui permet de réaliser cette dichotomie par une fonction indicatrice alors on dit que \mathbb{H} **pulvérise** l'ensemble de points.
- Pour rappel, une fonction indicatrice ind est telle que $ind(A) = 1$ si la proposition A est vraie ; $ind(A) = 0$ sinon.

Définition. (Dimension VC)

La dimension VC d'un espace d'hypothèses \mathbb{H} , notée $vc(\mathbb{H})$, est le cardinal du plus grand ensemble de points de \mathbb{X} que \mathbb{H} peut pulvériser.

Dimension de Vapnik-Chervonenkis (suite)

- Exemple : la dimension VC de $\mathbb{H} = \{f : \mathbb{R}^2 \rightarrow \mathbb{R} : f(X) = a_0 + a_1X^1 + a_2X^2\}$ (polynôme de degré 1) est $vc(\mathbb{H}) = 3$



Dimension de Vapnik-Chervonenkis (suite)

- Plus la dimension VC est grande plus \mathbb{H} est complexe. On dit également que \mathbb{H} a plus de capacité ou est plus flexible.
- Intuitivement, on voit que plus la dimension VC est grande, plus la forme de la frontière de décision est onduleuse ce qui permet de pulvériser plus de points (en opposition aux hyperplans).
- Autre exemple, l'espace d'hypothèses $\mathbb{H} = \{f : \mathbb{R}^p \rightarrow \{0, 1\} : f(\mathbf{x}) = \text{ind}(\sin(\alpha\mathbf{x}) > \nu), \alpha \in \mathbb{R}\}$ est tel que $vc(\mathbb{H}) = \infty$.

Dimension de Vapnik-Chervonenkis (suite)

- Soit $u_1, \dots, u_p, v \in \mathbb{R}$ où au moins un des u_i est non nul. Rappelons que l'ensemble des points $\mathbf{x} = (x_1, \dots, x_p) \in \mathbb{R}^p$ qui satisfait l'équation linéaire suivante est appelé un **hyperplan** de \mathbb{R}^p :

$$u_1x_1 + \dots + u_nx_n = v \text{ ce qui est équivalent à } \langle \mathbf{u}, \mathbf{x} \rangle = v$$

où $\langle \mathbf{u}, \mathbf{v} \rangle = \mathbf{u}^\top \mathbf{v}$ est le produit scalaire canonique de \mathbb{R}^p .

- Les hyperplans généralisent dans \mathbb{R}^p , le point dans \mathbb{R} , la droite dans \mathbb{R}^2 et le plan dans \mathbb{R}^3 .
- Nous pouvons alors généraliser la dimension VC de l'exemple précédent :

Propriété. (Dimension VC des hyperplans)

L'espace d'hypothèses

$\mathbb{H} = \{f : \mathbb{R}^p \rightarrow \{0, 1\} : f(\mathbf{x}) = \text{ind}(\langle \mathbf{x}, \mathbf{u} \rangle > \nu), \mathbf{u} \in \mathbb{R}^p, \nu \in \mathbb{R}\}$ est tel que $vc(\mathbb{H}) = p + 1$.

Apprentissage PAC

- L'apprentissage "**Probably Approximately Correct**" (PAC) de Valiant¹ [Valiant, 1984], est un sous-domaine théorique de l'AA qui s'intéresse aux propriétés de généralisation des méthodes.
- Soit C une classe de \mathbb{Y} et soient $\mathbb{E} = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_n, y_n)\}$ des exemples générés par une fonction de probabilité inconnue $P(X, Y)$.
- La question que traite l'apprentissage PAC est la suivante : combien d'exemples n faut-il pour qu'avec une probabilité $1 - \delta$, une hypothèse $\hat{f}_{\mathbb{E}} \in \mathbb{H}$ inférée d'un ensemble \mathbb{E} généré par $P(X, Y)$, commet en moyenne un taux d'erreur d'au plus ε ?
- Formellement on a :

$$P(\mathbb{E}_{\mathbb{E}}(\ell(\hat{f}_{\mathbb{E}}(X), Y)) < \varepsilon) \geq 1 - \delta$$

- Autrement dit, "avec probabilité plus grande que $1 - \delta$, on a un taux d'erreur plus petit que ε ".

1. Prix Nevanlinna 1986, Prix Knuth 1997, Prix Turing 2010

Apprentissage PAC et dimension VC

- L'un des résultats théoriques majeurs développés par Vapnik et Chervonenkis est d'avoir pu déterminer une borne de la probabilité précédente qui dépend de la dimension VC de la méthode utilisée.
- Pour alléger les formules, on introduit les notations suivantes :
 - ▶ $risk(f) = E_{X,Y}(\ell(f(X), Y))$ est le **risque théorique**.
 - ▶ $risk_{emp}(\hat{f}_{\mathbb{E}}) = \frac{\sum_{i=1}^n \ell(\hat{f}_{\mathbb{E}}(x_i), y_i)}{n}$ est le **risque empirique** étant donné \mathbb{E} .

Propriété. (Borne PAC et dim. VC pour un pb de classement binaire)

Si on estime une fonction de prédiction $\hat{f}_{\mathbb{E}} \in \mathbb{H}$ à partir de \mathbb{E} alors avec une probabilité au moins égale à $1 - \delta$ on a :

$$risk(f) \leq risk_{emp}(\hat{f}_{\mathbb{E}}) + \frac{\eta}{2} \left(1 + \sqrt{1 + \frac{4risk_{emp}(\hat{f}_{\mathbb{E}})}{\eta}} \right)$$

$$\text{où } \eta = \alpha \frac{vc(\mathbb{H}) \left(\log\left(\beta \frac{n}{vc(\mathbb{H})}\right) + 1 \right) - \log\left(\frac{\delta}{4}\right)}{n}, \quad 0 \leq \alpha \leq 4 \text{ et } 0 \leq \beta \leq 2$$

Apprentissage PAC et dimension VC (suite)

- La borne PAC précédente est un résultat théorique intéressant car elle est valable pour n'importe quelle probabilité jointe $P(X, Y)$ et qu'elle ne dépend que de la dimension de VC de la classe d'hypothèses.
- Toutefois :
 - ▶ L'absence de précision sur la forme de $P(X, Y)$ rend la borne peu efficace et on obtient un nombre n qui est surestimé.
 - ▶ Par ailleurs, $vc(\mathbb{H})$ est en général très difficile à calculer.
- En pratique, il est donc difficile d'utiliser ce résultat.
- Il n'empêche que ces questions de nature théorique restent importantes.

Malédiction de la dimensionalité

- Dans les exemples précédents, les problèmes étaient de faibles dimensions avec $|\mathbb{A}|$ ou encore $dim(\mathbb{X})$ petits.
- Dans beaucoup de problèmes pratiques, les vecteurs x_i appartiennent en fait à un espace de très grande dimension.
- C'est le cas notamment lorsque les objets sont des textes ou des images. Par exemple, l'espace de description d'un texte est potentiellement l'ensemble du vocabulaire de la langue considérée (soit plus de 60000 descripteurs pour le français).
- Il arrive parfois que $|\mathbb{A}| = p$ est plus grand que $|\mathbb{E}| = n$.
- Par ailleurs, on pourrait penser, comme suggérer précédemment, que si n est très grand alors on est toujours capable d'avoir de bons résultats en généralisation.
- Dans le cas des données de grande dimension ceci n'est malheureusement pas vrai notamment pour les méthodes locales (telles que les k -ppv ou les méthodes à noyau).

Malédiction de la dimensionalité (suite)

- On a l'habitude d'évoluer dans un espace à 3 dimensions au sein duquel, les distances entre objets nous paraissent "claires". En revanche, les espaces de grande dimension sont beaucoup plus "vastes" et les mesures de distances ne s'appréhendent pas de la même manière qu'en 3 dimensions.
- Exemple [Hastie et al., 2011] : considérons une hypersphère de rayon unitaire centrée à l'origine d'un espace de dimension p ; considérons également un ensemble de n points générés aléatoirement selon une loi uniforme à l'intérieur de cette hypersphère.
- On considère les plus proches voisins de l'origine de l'hypersphère et on montre que la distance médiane du plus proche voisin de l'origine est donnée par la formule suivante :

$$d(n, p) = \left(1 - \left(\frac{1}{2}\right)^{1/n} \right)^{1/p}$$

Malédiction de la dimensionalité (suite)

- Pour $p = 1$ (intervalle $[-1, 1]$), on a :
 - ▶ Pour $n = 20$: $d(20, 1) \approx 0.03$.
 - ▶ Pour $n = 500$: $d(500, 1) \approx 0.001$.
 - ▶ Pour $n = 20000$: $d(20000, 1) \approx 3 \times 10^{-5}$.
- Pour $p = 3$ (boule de rayon 1), on a :
 - ▶ Pour $n = 20$: $d(20, 3) \approx 0.32$.
 - ▶ Pour $n = 500$: $d(500, 3) \approx 0.11$.
 - ▶ Pour $n = 20000$: $d(20000, 3) \approx 0.03$.
- Pour $p = 10$ (hypersphère de rayon 1), on a :
 - ▶ Pour $n = 20$: $d(20, 10) \approx 0.71$.
 - ▶ Pour $n = 500$: $d(500, 10) \approx 0.52$.
 - ▶ Pour $n = 20000$: $d(20000, 10) \approx 0.36$.
 - ▶ Pour $n = 2 \times 10^{14}$: $d(2 \times 10^{14}, 10) \approx 0.03$.
- Ainsi pour avoir la même couverture de l'espace entre un espace de petite dimension et un espace de plus grande dimension, le nombre de points nécessaires croît de façon exponentielle !

Rappel du Sommaire

1 Introduction

- L'apprentissage automatique
- Quelques méthodes simples en guise d'illustration
- Différentes caractéristiques des méthodes d'apprentissage supervisé
- (Quelques) Problèmes théoriques en apprentissage automatique
- Evaluation et comparaison de modèles en apprentissage supervisé

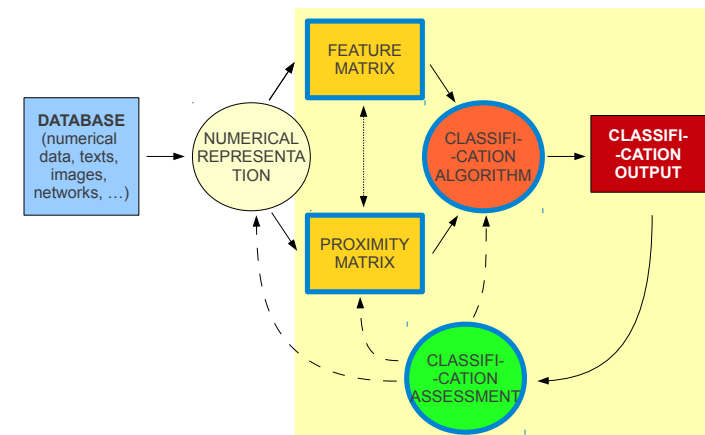
Malédiction de la dimensionalité (suite)

- Un autre problème associé à la dimensionalité est le suivant : à mesure que p augmente, les mesures de distances sont de moins en moins "significatives". En fait, la mesure de distance séparant les deux points les plus éloignés ($dist_{max}$) et celle séparant les points les plus proches ($dist_{min}$) sont de plus en plus comparables. Dans [Beyer et al., 1999], on montre dans un cadre assez courant que :

$$\forall \varepsilon > 0, \lim_{p \rightarrow \infty} P \left(\left| \frac{dist_{max}}{dist_{min}} - 1 \right| \leq \varepsilon \right) = 1$$

- Le traitement des données de grandes dimensions forme un sous-domaine particulier de l'AA puisque les méthodes développées dans le cas des données de faibles dimensions ne sont pas efficaces.
- Dans ce cas, une façon de procéder est d'appréhender les variables discriminantes des données en déterminant les sous-espaces de dimensions plus faibles au sein desquels les distances redeviennent significatives.

Schéma général



Protocol expérimental en apprentissage supervisée

- Etant donné une tâche d'apprentissage supervisé, le but est donc d'**estimer plusieurs modèles afin de prédire au mieux la variable cible pour des données futures**. Pour sélectionner le modèle, il faut procéder en distinguant au moins deux ensembles de données.
 - 1 Un ensemble des **données d'apprentissage ou d'entraînement** \mathbb{E} à partir duquel on estime une ou plusieurs fonctions de prédiction appartenant à un ou plusieurs espaces d'hypothèses.
 - 2 Un ensemble de **données de validation** noté \mathbb{V} qui n'est pas utilisé lors de l'estimation des modèles et qui sert à mesurer l'erreur de prédiction des différents modèles appris.
- C'est l'erreur de prédiction mesurée sur \mathbb{V} qui permet en pratique de sélectionner le meilleur modèle \hat{f}^* .
- 3 En revanche, si l'on souhaite avoir une estimation de l'erreur en généralisation de \hat{f}^* alors on ne peut pas utiliser celle mesurée à l'aide de \mathbb{V} . On a recourt à un troisième jeu de données appelé ensemble de **données de test** et noté \mathbb{T} .

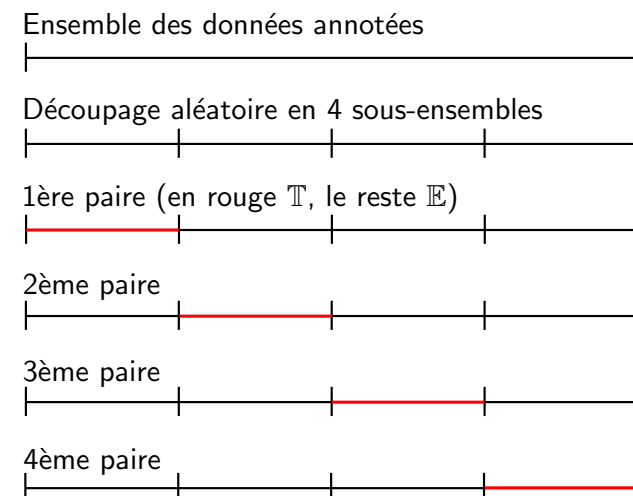
Validation croisée

- Précédemment on a supposé les données annotées séparées en \mathbb{E} et \mathbb{T} . Mais l'estimation de l'erreur de prédiction est plus précise si on avait à disposition **plusieurs ensembles** \mathbb{E} et \mathbb{T} .
- La **validation croisée** consiste à :
 - ▶ Séparer aléatoirement l'ensemble des données annotées en k sous-ensembles.
 - ▶ Utiliser un sous-ensemble comme ensemble de test \mathbb{T} .
 - ▶ Utiliser l'union des $k - 1$ sous-ensembles restants comme ensemble d'entraînement \mathbb{E} .
- En changeant chaque fois l'ensemble de validation, on voit qu'une **validation croisée** permet d'avoir k paires d'échantillons (\mathbb{E}, \mathbb{T}) et ainsi k estimations de l'erreur de prédiction.
- On moyenne l'ensemble des k mesures d'erreurs afin d'avoir une estimation plus robuste de l'erreur de prédiction.
- Si $k = n$ on parle de "**leave one out cross validation (LOOCV)**". On apprend sur $n - 1$ individus et on teste sur 1 individu (n fois).

Protocol expérimental en apprentissage supervisée (suite)

- En général on prend 50% des données annotées pour \mathbb{E} , 25% pour \mathbb{V} et 25% pour \mathbb{T} . Mais il n'y a pas en théorie de découpage optimal.
- Dans certaines situations, on utilisera uniquement un ensemble de données d'**entraînement** \mathbb{E} et un ensemble de données de **test** \mathbb{T} :
 - ▶ Lorsque nous voulons tester un seul modèle et non plusieurs. Dans ce cas, l'ensemble de données de validation n'est pas nécessaire.
 - ▶ Lorsque l'ensemble des données annotées n'est pas grand (n relativement petit). Dans ce cas, il devient difficile de découper en trois l'ensemble des données annotées et d'obtenir un bon apprentissage.
- Le second cas est souvent rencontré en pratique. En effet, il est en général difficile d'avoir une grande quantité de données annotées car cela nécessite l'intervention humaine et la tâche d'annotation est fastidieuse.
- Nous présentons dans la suite des méthodes permettant d'avoir une bonne estimation de l'erreur en généralisation.

Validation croisée (suite)



Bootstrap

- Une alternative à la validation croisée, qui est notamment utilisée lorsque l'ensemble des données annotées est de taille très réduite est la méthode de rééchantillonnage dite du **"bootstrap"**.
- La méthode consiste à générer de nouveaux échantillons à partir de l'échantillon initial :
 - ▶ On tire aléatoirement avec remise n objets de \mathbb{E} et on obtient ainsi \mathbb{E}' .
 - ▶ On infère de \mathbb{E}' une fonction de prédiction.
- On répète le processus et on crée ainsi k échantillons bootstrap permettant d'inférer k fonctions de prédiction.
- L'idée est ensuite de moyennner l'erreur de prédiction donnée par ces k fonctions de prédiction ce qui permet d'avoir une estimation plus robuste. Mais, il faut faire attention de bien définir pour chaque fonction estimée un ensemble de test adéquat.

Mesures d'évaluation

- Précédemment, nous avons vu des fonctions de performances pour la régression et la catégorisation que l'on cherche à optimiser en utilisant les données \mathbb{E} afin d'inférer des fonctions de prédiction appartenant à une ou plusieurs classes d'hypothèses.
- Pour la sélection des modèles on peut avoir recours à d'autres types de critères d'évaluation mesuré sur les données \mathbb{V} et/ou \mathbb{T} , indiquant la plus ou moins bonne performance d'une fonction de prédiction. Ces différentes mesures permettent de mieux comparer les modèles entre eux.
- En ce qui concerne la régression, les critères courants sont :
 - ▶ La somme des carrés des résidus (*scr* ou "Residual Sum of Squares").
 - ▶ La moyenne des carrés des résidus ("Mean Squared Error").
 - ▶ La moyenne des résidus en valeurs absolues ("Mean Absolute Error")
- Pour ce qui est du problème de catégorisation :
 - ▶ Le taux d'erreur.
 - ▶ La précision.
 - ▶ Le rappel.

Bootstrap (suite)

- Si les tirages sont mutuellement indépendants, la probabilité pour qu'un objet ne soit pas tiré après n tirages est environ de 37%. Donc environ 63% des objets de \mathbb{E} serviront à l'estimation du modèle et 37% des objets restants peuvent servir au test.
- Ainsi, pour chaque fonction de prédiction apprise sur un échantillon bootstrap \mathbb{E}' on garde en mémoire les objets de test $\mathbb{T}' = \mathbb{E} \setminus \mathbb{E}'$ à partir desquels on estime l'erreur de prédiction.
- L'estimation de l'erreur de prédiction basée sur le **"leave one out bootstrap"** consiste alors à :
 - ▶ Générer k échantillons bootstrap $(\mathbb{E}', \mathbb{T}')$.
 - ▶ Apprendre k fonctions de prédiction à partir des k échantillons bootstrap.
 - ▶ Evaluer l'erreur de prédiction moyenne de chaque objet X_i de \mathbb{E} mais en n'utilisant que les fonctions dont X_i n'a pas été un objet d'entraînement.
 - ▶ Evaluer l'erreur de prédiction en moyennant sur chaque objet X_i de \mathbb{E} son erreur de prédiction moyenne.

Mesures d'évaluation pour le problème de régression

- La Somme des carrés des résidus ou les Moindres Carrés Ordinaires ("Residual Sum of Square") :

$$scr(f) = \sum_{i=1}^n (y_i - f(\mathbf{x}_i))^2$$

- La Moyennes des carrés des résidus ("Mean Squared Error") :

$$mse(f) = \frac{1}{n} \sum_{i=1}^n (y_i - f(\mathbf{x}_i))^2$$

- Contrairement au *scr*, le *mse* permet de comparer les erreurs de prédiction mesurés sur des ensembles de données de tailles différentes
- La moyenne des résidus en valeurs absolues ("Mean Absolute Error") :

$$mae(f) = \frac{1}{n} \sum_{i=1}^n |y_i - f(\mathbf{x}_i)|$$

Mesures d'évaluation pour le problème de régression (code R)

```
> lm_fit_deg_1$residuals
      1      2      3      4      5      6      7
0.1820817 0.8329108 1.0864414 -0.6129203 -0.1524638 -0.5252052 -0.7285628
      8      9     10     11     12
0.5256503 0.9397729 -0.9480325 -0.2474956 -0.3521770
>
> scr=sum(lm_fit_deg_1$residuals^2)
> mse=sum(lm_fit_deg_1$residuals^2)/length(lm_fit_deg_1$residuals)
> mae=sum(abs(lm_fit_deg_1$residuals))/length(lm_fit_deg_1$residuals)
>
> scr
[1] 5.35634
> mse
[1] 0.4463616
> mae
[1] 0.5944762
```

Mesures d'évaluation pour le problème de catégorisation binaire (suite)

- En statistique on interprète souvent une classe comme étant la classe "positive" (C_1 par exemple) et l'autre classe comme étant la classe "négative" (resp. C_2). Par exemple C_1 = "Malade" et C_2 = "Sain". Dans ce cas, les différentes valeurs du tableau de contingence sont aussi connues sous les vocables suivants :

		$\hat{f}(x)$		Total
		C_1	C_2	
y	C_1	a	b	$a + b$
	C_2	c	d	$c + d$
Total		$a + c$	$b + d$	$a + b + c + d = n$

- a = Vrais positifs ("True Positive", tp)
- b = Faux négatifs ("False Negative", fn)
- c = Faux positifs ("False Positive", fp)
- d = Vrais négatifs ("True Negative", tn)

Mesures d'évaluation pour le problème de catégorisation binaire

- Quand il y a uniquement deux classes $\mathbb{Y} = \{C_1, C_2\}$, beaucoup de mesures de performance sont décrites par le biais du tableau de contingence suivant appelé **matrice de confusion** :

		$\hat{f}(x)$		Total
		C_1	C_2	
y	C_1	a	b	$a + b$
	C_2	c	d	$c + d$
Total		$a + c$	$b + d$	$a + b + c + d = n$

- a = Nb d'objets C_1 correctement catégorisés
- b = Nb d'objets C_1 catégorisés en C_2
- c = Nb d'objets C_2 catégorisés en C_1
- d = Nb d'objets C_2 correctement catégorisés

Mesures d'évaluation pour le problème de catégorisation binaire (suite)

		$\hat{f}(x)$		Total
		C_1	C_2	
y	C_1	a	b	$a + b$
	C_2	c	d	$c + d$
Total		$a + c$	$b + d$	$a + b + c + d = n$

- Taux d'erreur ("Error rate" ou "Misclassification Rate") :

$$err(\hat{f}) = \frac{b + c}{n}$$

- Taux de réussite ou de reconnaissance ("Accuracy Rate") :

$$acc(\hat{f}) = \frac{a + d}{n} = 1 - err(\hat{f})$$

Mesures d'évaluation pour le problème de catégorisation binaire (suite)

		$\hat{f}(\mathbf{x})$		Total
		C_1	C_2	
y	C_1	a	b	$a + b$
	C_2	c	d	$c + d$
Total		$a + c$	$b + d$	$a + b + c + d = n$

- Taux de vrais positifs ("True positive rate") et taux de faux positifs ("False positive rate" ou "False alarm rate") :

$$tp(\hat{f}) = \frac{a}{a+b} \text{ et } fp(\hat{f}) = \frac{c}{c+d}$$

- Sensitivité ("sensitivity") et spécificité ("specificity") :

$$sen(\hat{f}) = tp(\hat{f}) = \frac{a}{a+b} \text{ et } spe(\hat{f}) = 1 - \frac{c}{c+d}$$

Courbe ROC

- Toujours dans le cas binaire, supposons une fonction de prédiction qui soit dépendante d'un seuil :

$$\hat{f}(\mathbf{x}) = \begin{cases} C_1 & \text{si } \hat{g}(\mathbf{x}) \geq \delta \text{ (classe "positive")} \\ C_2 & \text{si } \hat{g}(\mathbf{x}) < \delta \end{cases}$$

- A titre illustratif et pour fixer les idées, on pourra interpréter $\hat{g}(\mathbf{x})$ comme étant le score obtenu par \mathbf{x} pour la fonction discriminante (cas de la régression linéaire avec variables artificielles $C_1 \leftrightarrow 1$ et $C_2 \leftrightarrow -1$) associée à C_1 et δ le seuil au-dessus duquel on considère que \mathbf{x} est dans C_1 .
- Dans ce contexte, on s'intéresse typiquement aux mesures tp et fp d'un modèle pour son évaluation (toutefois d'autres mesures peuvent être utilisées comme précision et rappel).

Mesures d'évaluation pour le problème de catégorisation binaire (code R)

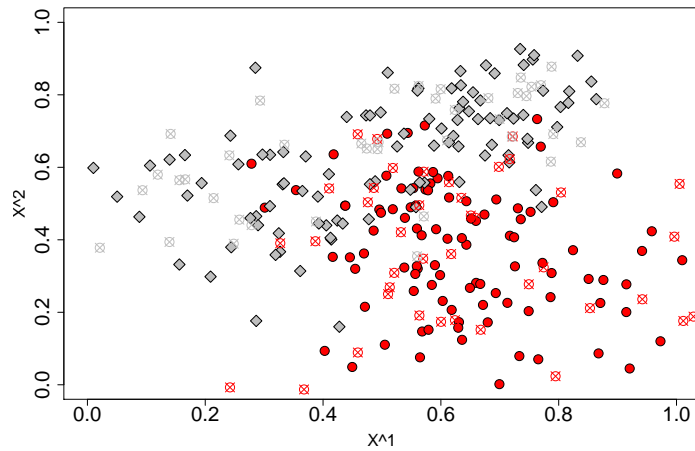
```
> lm_pred=ifelse(predict(lm_fit,X)>=0,1,-1)
> conf_mat=table(c,lm_pred)
> conf_mat
      lm_pred
c      -1  1
-1  74 26
 1   7 93
> a=conf_mat[1,1];b=conf_mat[1,2];c=conf_mat[2,1];d=conf_mat[2,2];n=sum(sum(conf_mat))
> err=(b+c)/n;acc=1-err;tp=a/(a+b);fp=c/(c+d)
> err
[1] 0.165
> acc
[1] 0.835
> tp
[1] 0.74
> fp
[1] 0.07
```

Courbe ROC (suite)

- Le seuil δ est dans ce cas un paramètre à déterminer et on voit qu'en fonction de sa valeur, les mesures d'erreurs fluctuent. Par exemple, si le classifieur est basé sur une fonction discriminante comme précédemment alors si δ est proche de 1, il sera très difficile d'affecter des objets de \mathbb{T} dans la classe C_1 et dans ce cas $fp(\hat{f})$ mais aussi $tp(\hat{f})$ auront tendance à être faibles.
- Pour différentes valeurs de δ , on obtient plusieurs valeurs pour la paire $(fp(\hat{f}), tp(\hat{f}))$.
- Le graphe de ces différents points dans le repère fp en abscisse et tp en ordonnée est appelée **courbe ROC "Receiver Operating Characteristics"**.
- Idéalement on aimerait trouver δ tel que $tp(\hat{f}) = 1$ et $fp(\hat{f}) = 0$ mais plus facile à dire qu'à faire !
- Ainsi, les modèles \hat{f} relatifs aux δ dont les points de coordonnées $(fp(\hat{f}), tp(\hat{f}))$ sont proches du coin supérieur gauche sont meilleurs que les autres.

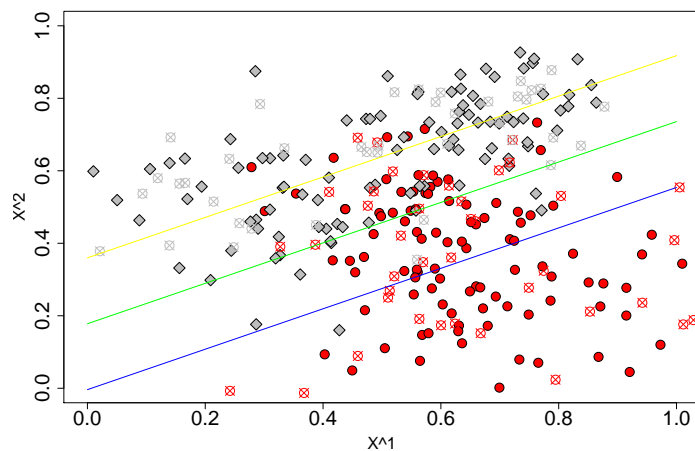
Courbe ROC (suite)

- Reprenons l'exemple de catégorisation auquel on a ajouté les données de test \mathbb{T} .



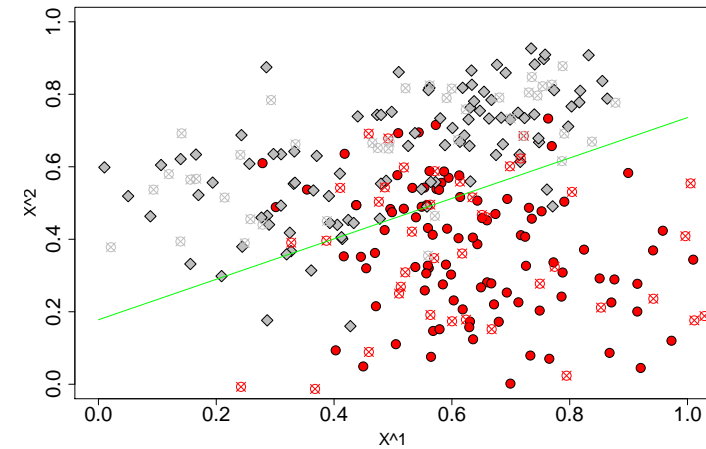
Courbe ROC (suite)

- Régression linéaire simple sur variables artificielles et frontières de décision $\hat{g}(\mathbf{x}) = \delta$ avec $\delta = 0.5, 0, -0.5$.



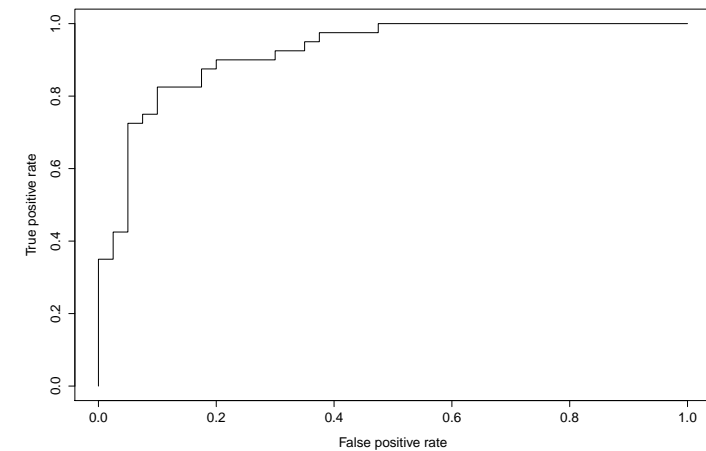
Courbe ROC (suite)

- Régression linéaire simple sur variables artificielles et frontière de décision $\hat{g}(\mathbf{x}) = 0$.



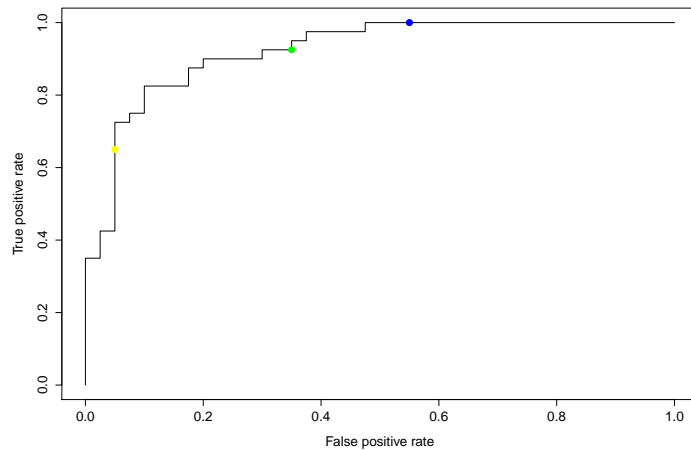
Courbe ROC (suite)

- Pour l'exemple précédent, on obtient la courbe ROC suivante :



Courbe ROC (suite)

- Les points correspondent aux valeurs pour $\delta = 0.5, 0, -0.5$.

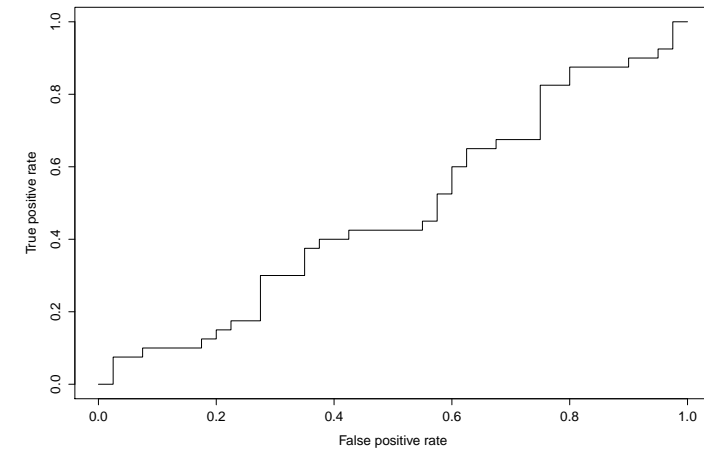


Courbe ROC et AUC "Area Under the Curve"

- Pour qu'un modèle soit intéressant il faut qu'il soit meilleur qu'un classifieur aléatoire : la courbe ROC du modèle doit pour cela être au-dessus de la diagonale.
- On peut comparer deux types de fonction de prédiction en utilisant les courbes ROC : le modèle dont la courbe est au-dessus de l'autre est le meilleur.
- La courbe ROC permet une évaluation graphique des performances d'un classifieur. On peut par aussi résumer le graphique par un nombre appelé "Area Under the Curve" qui est l'indice *auc*. $auc(\hat{f})$ est la mesure de la surface sous la courbe ROC.
- Idéalement on souhaite déterminer un modèle \hat{f} tel que $auc(\hat{f}) = 1$.
- Le modèle \hat{f} est meilleur que \hat{f}' si $auc(\hat{f}) > auc(\hat{f}')$.
- Le modèle \hat{f} est meilleur que le classifieur aléatoire si $auc(\hat{f}) > 0.5$.

Courbe ROC (suite)

- Courbe ROC pour un classifieur aléatoire (on tire au hasard dans $\{C_1, C_2\}$ pour chaque point) :



Mesures d'évaluation pour le problème de catégorisation multiclasse

- Quand $\mathbb{Y} = \{C_1, C_2, \dots, C_q\}$ avec $q > 2$, on parle d'un problème de catégorisation multiclasse.
- La **matrice de confusion** est alors une matrice carrée \mathbf{N} d'ordre q .
- Le terme $\mathbf{N}(l, l') = \mathbf{N}_{ll'}$ indique le nombre d'objets \mathbf{x} de \mathbb{T} appartenant à la classe C_l et ayant été affecté à la classe $C_{l'}$ par $\hat{f}(\mathbf{x})$.
- Idéalement, il faudrait que les termes hors diagonale ne contiennent que des 0 ce qui conduirait à un taux d'erreur nul.
- Le taux de reconnaissance est la somme des termes de la diagonale divisée par le cardinal de \mathbb{T} .
- L'analyse de la matrice de confusion permet de déterminer les paires de classes les plus difficiles à séparer.
- Des tests statistiques permettent également de comparer les résultats de plusieurs modèles et sur plusieurs bases de données [Alpaydin, 2010, Cornuéjols and Miclet, 2003].

Mesures d'évaluation pour le problème de catégorisation multiclasse (suite)

- Dans le cas multiclassés on a la matrice de confusion \mathbf{N} de taille $(q \times q)$:

$$\mathbf{N} = \begin{array}{c|cc|cc} & & \hat{f}(\mathbf{x}) & & \\ & & C_1 & \dots & C_q \\ \hline y & C_1 & & & \\ & \vdots & & & \\ & C_q & & & \end{array}$$

- On généralise au cas multiclassés (avec un coût uniforme) le taux d'erreur ("Error rate" ou "Misclassification Rate") et le taux de reconnaissance ("Accuracy rate") :

$$err(\hat{f}) = \frac{\sum_{l \neq l'} \mathbf{N}_{ll'}}{\sum_{l, l'} \mathbf{N}_{ll'}} \quad \text{et} \quad acc(\hat{f}) = 1 - err(\hat{f})$$

Rappel du Sommaire

- 1 Introduction
- 2 Les méthodes linéaires et leurs pénalisations (ridge, lasso, ...)
- 3 Les machines à vecteurs supports ("Support Vector Machines")
- 4 Les arbres de décisions ("Decision Trees")
- 5 Décider en comité ("Ensemble Learning")

Autres critères pour comparer deux modèles

- Au-delà des critères de performances de type erreur de prédiction ou en généralisation, il faut également tenir compte de plusieurs autres critères lorsque l'on compare des algorithmes d'apprentissage supervisé :
 - ▶ La **complexité en temps de traitement et en espace mémoire** : on parle d'algorithmes ou de modèles scalables ou non.
 - ▶ L'**interprétabilité du modèle estimé** : au-delà d'une simple prédiction de valeurs ou de classe, est-ce que le modèle estimé permet une meilleure connaissance sur le processus génératif qui engendre les observations (X, Y) ou s'agit-il d'une "boîte noire" ?
 - ▶ La capacité des modèles à s'adapter à des **données** qui peuvent être **hétérogènes et/ou manquantes et/ou aberrantes et/ou non pertinentes** vis à vis du problème considéré.
- Principe de simplicité dit du **rasoir d'Occam** : à erreur de prédiction comparable, on préférera le modèle de complexité la moindre permettant l'interprétation la plus simple du phénomène étudié.

Introduction

- Les méthodes de régression linéaire supposent que la fonction de régression $E(Y|X)$ est une fonction linéaire des paramètres \mathbb{P} .
- Ce sont des méthodes développées depuis le XVIIIème siècle en statistiques et qui sont encore de nos jours très utilisées car elles sont simples et permettent une bonne interprétation de l'influence des variables explicatives sur la variable à expliquer :

$$f(X) = \sum_j a_j X^j \Rightarrow \frac{\partial f}{\partial X^j}(X) = a_j$$

- Des développements récents sont également proposés permettant d'enrichir la panoplie de ce type de méthodes. En particulier, certaines méthodes aboutissant à des frontières de décision non linéaires sont en fait des généralisations des méthodes linéaires (au sens d'un polynôme de degré 1 des paramètres \mathbb{P}).
- On étudiera pour les problèmes de régression et de catégorisation, les fondements et la mise en oeuvre de méthodes de base et avancées.

Rappel du Sommaire

- 2 Les méthodes linéaires et leurs pénalisations (ridge, lasso, ...)
 - Méthodes linéaires pour la régression
 - Méthodes linéaires pour la catégorisation

Régression linéaire multiple et MCO (suite)

- L'étape d'induction consiste à estimer les paramètres \mathbb{P} étant données les données d'entraînement \mathbb{E} .
- La méthode classique est les **Moindres Carrés Ordinaires** (MCO) :

$$\begin{aligned} scr(f) &= \sum_{i=1}^n (y_i - f(\mathbf{x}_i))^2 \\ &= \sum_{i=1}^n (y_i - (a_0 + \sum_{j=1}^p a_j x_{ij}))^2 \end{aligned}$$

- Du point de vue statistique, l'utilisation de ce modèle suppose que les observations y_i sont des réalisations de v.a. Y_i i.i.d..
- Introduisons les notations suivantes :
 - ▶ \mathbf{a} , le vecteur colonne de taille $p + 1$ contenant les paramètres.
 - ▶ \mathbf{X} , la matrice des données de taille $(n \times (p + 1))$ à laquelle on a ajouté une 1ère colonne remplie de 1.

Régression linéaire multiple et MCO

- Rappelons que nous souhaitons déterminer une fonction f modélisant la relation entre la variable cible Y et les variables explicatives $\{X^1, X^2, \dots, X^p\}$ qui constituent l'espace de description des objets \mathbb{X} .
- Le modèle de régression linéaire est le suivant :

$$Y = f(X^1, \dots, X^p) + \epsilon = a_0 + \sum_{j=1}^p a_j X^j + \epsilon$$

- On a donc $\mathbb{H} = \{f : \mathbb{R}^p \rightarrow \mathbb{R} : f(\mathbf{X}) = a_0 + \sum_{j=1}^p a_j X^j\}$.
- Les variables explicatives peuvent être :
 - ▶ Les variables initiales.
 - ▶ Des transformations des variables initiales.
 - ▶ Des expansions de bases des variables initiales [Hastie et al., 2011].
- Le modèle reste une fonction linéaire des paramètres $\mathbb{P} = \{a_j\}_{j=0}^p$.

Régression linéaire multiple et MCO (suite)

- Nous avons :

$$\mathbf{a} = \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_p \end{pmatrix} ; \quad \mathbf{X} = \begin{pmatrix} 1 & x_{11} & x_{12} & \dots & x_{1p} \\ 1 & x_{21} & x_{22} & \dots & x_{2p} \\ \vdots & \vdots & \dots & \dots & \vdots \\ 1 & x_{n1} & x_{n2} & \dots & x_{np} \end{pmatrix} ; \quad \mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}$$

- Notons par ailleurs \mathbf{X}^\top la matrice transposée de \mathbf{X} .
- Nous avons alors l'écriture matricielle suivante :

$$scr(f) = (\mathbf{y} - \mathbf{X}\mathbf{a})^\top (\mathbf{y} - \mathbf{X}\mathbf{a})$$
- On cherche à déterminer les paramètres $\mathbb{P} = \{a_j\}_{j=0}^p$ représentés par le vecteur \mathbf{a} qui minimise $scr(f)$: c'est un problème d'optimisation quadratique non contraint :

$$\hat{\mathbf{a}}_{mco} = \arg \min_{\mathbf{a} \in \mathbb{R}^{p+1}} (\mathbf{y} - \mathbf{X}\mathbf{a})^\top (\mathbf{y} - \mathbf{X}\mathbf{a})$$

- La solution s'obtient en recherchant les points \mathbf{a} tel que $\nabla scr(\mathbf{a}) = \mathbf{0}$.

Rappels en calcul différentiel

- Si $f : \mathbb{R}^{p+1} \rightarrow \mathbb{R}$ est différentiable, alors la fonction ∇f défini par :

$$\nabla f(\mathbf{a}) = \begin{pmatrix} \frac{\partial f}{\partial a_0}(\mathbf{a}) \\ \frac{\partial f}{\partial a_1}(\mathbf{a}) \\ \vdots \\ \frac{\partial f}{\partial a_p}(\mathbf{a}) \end{pmatrix}$$

est appelé **gradient** de f .

- ∇f est une fonction de \mathbb{R}^{p+1} dans \mathbb{R}^{p+1} et peut être vue comme un **champ de vecteurs** (fonction qui associe à tout point un vecteur).
- Quelques formules de dérivations dans le cas multivarié. La dérivée est calculée par rapport à la variable \mathbf{x} . \mathbf{A} est une matrice de réels de taille $(m \times n)$ et \mathbf{y} un vecteur de réels de taille $(m \times 1)$:
 - Si $f(\mathbf{x}) = \mathbf{y}^\top \mathbf{A} \mathbf{x}$ ou si $f(\mathbf{x}) = \mathbf{x}^\top \mathbf{A}^\top \mathbf{y}$ alors $\nabla f(\mathbf{x}) = \mathbf{A}^\top \mathbf{y}$.
 - Si \mathbf{A} est carrée et $f(\mathbf{x}) = \mathbf{x}^\top \mathbf{A} \mathbf{x}$ alors $\nabla f(\mathbf{x}) = (\mathbf{A} + \mathbf{A}^\top) \mathbf{x}$.
 - Si \mathbf{A} est carrée symétrique et $f(\mathbf{x}) = \mathbf{x}^\top \mathbf{A} \mathbf{x}$ alors $\nabla f(\mathbf{x}) = 2\mathbf{A} \mathbf{x}$.

Régression linéaire multiple et MCO (suite)

- Une fois estimé $\hat{\mathbf{a}}_{mco}$ on peut calculer les prédictions du modèle pour un quelconque $\mathbf{x} \in \mathbb{X}$:

$$\hat{f}(\mathbf{x}) = \mathbf{x}^\top \hat{\mathbf{a}}_{mco} = \mathbf{x}^\top (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}$$

- Pour calculer l'erreur de prédiction on regarde ce que prédit le modèle estimé pour les données \mathbb{E} données par les lignes de \mathbf{X} :

$$\hat{\mathbf{y}} = \mathbf{X} \hat{\mathbf{a}}_{mco} = \mathbf{X} (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}$$

- L'erreur de prédiction est donc donnée par :

$$\begin{aligned} scr(\hat{f}) &= \sum_{i=1}^n (y_i - \hat{y}_i)^2 \\ &= \|\mathbf{y} - \hat{\mathbf{y}}\|^2 \end{aligned}$$

où $\|\cdot\|$ est la norme euclidienne.

Régression linéaire multiple et MCO (suite)

- On développe $scr(f)$ de la manière suivante :

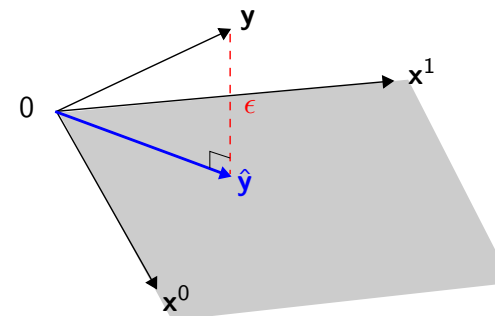
$$\begin{aligned} scr(f) &= (\mathbf{y} - \mathbf{X} \mathbf{a})^\top (\mathbf{y} - \mathbf{X} \mathbf{a}) \\ &= (\mathbf{y}^\top - (\mathbf{X} \mathbf{a})^\top) (\mathbf{y} - \mathbf{X} \mathbf{a}) \\ &= \mathbf{y}^\top \mathbf{y} - \mathbf{y}^\top \mathbf{X} \mathbf{a} - (\mathbf{X} \mathbf{a})^\top \mathbf{y} + (\mathbf{X} \mathbf{a})^\top \mathbf{X} \mathbf{a} \\ &= \mathbf{y}^\top \mathbf{y} - \mathbf{y}^\top \mathbf{X} \mathbf{a} - \mathbf{a}^\top \mathbf{X}^\top \mathbf{y} + \mathbf{a}^\top \mathbf{X}^\top \mathbf{X} \mathbf{a} \end{aligned}$$

- On a donc la solution suivante :

$$\begin{aligned} \nabla scr(f) = \mathbf{0} &\Leftrightarrow 2\mathbf{X}^\top \mathbf{X} \mathbf{a} - 2\mathbf{X}^\top \mathbf{y} = \mathbf{0} \\ &\Leftrightarrow \hat{\mathbf{a}}_{mco} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y} \end{aligned}$$

Régression linéaire multiple et MCO (suite)

- Interprétation géométrique :



$$\hat{\mathbf{y}} = \underbrace{\mathbf{X} (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top}_{\text{Opérateur de projection}} \mathbf{y}$$

- Les MCO consistent à projeter orthogonalement \mathbf{y} sur le sous-espace de \mathbb{R}^n engendré par $\{\mathbf{x}^0, \dots, \mathbf{x}^p\}$ (les $p + 1$ colonnes de \mathbf{X}).
- Remarque : comme on cherche à minimiser $scr(f)$, on voit que la plus courte distance entre \mathbf{y} et le sous-espace est donnée par la projection orthogonale.

Régression linéaire multiple et MCO (suite)

- Les MCO supposent que $\mathbf{X}^\top \mathbf{X}$ est non singulière (inversible). On suppose donc que \mathbf{X} est de **plein rang**. En pratique, ce sont les variables colinéaires qui rendent la matrice $\mathbf{X}^\top \mathbf{X}$ singulière. On pourra alors supprimer ces “redondances” au préalable.
 - Par ailleurs, les MCO supposent également que $n > p$, càd **nombre d'observations** > **nombre de variables**. Dans le cas contraire, (comme cela se produit pour les problèmes de grandes dimensions), on pourra réduire l'espace de représentation \mathbb{X} au préalable (ACP par exemple, régression sur composantes principales).
 - Si $\mathbf{X}^\top \mathbf{X}$ est singulière alors il existe une infinité de $\hat{\mathbf{a}}_{mco}$: les coefficients ne sont pas uniques et le **problème n'est pas identifiable**.
- ▷ Nous verrons plus loin qu'au-delà des artefacts que nous venons de mentionner pour s'accommoder de la singularité de $\mathbf{X}^\top \mathbf{X}$, il existe des méthodes élégantes permettant de pallier à ce problème.

Régression linéaire multiple et modèle gaussien

- Nous réinterprétons la régression linéaire multiple dans un cadre probabiliste. Nous avons le modèle suivant pour tout $i = 1, \dots, n$:

$$Y_i = \mathbf{X}_i^\top \mathbf{a} + \epsilon_i$$

- Nous faisons de plus l'hypothèse que le vecteur $\epsilon = (\epsilon_1, \dots, \epsilon_n) \sim \mathcal{N}_n(\mathbf{0}, \sigma^2 \mathbf{I}_n)$ où \mathbf{I}_n est la matrice identité d'ordre n .
- Autrement dit les ϵ_i sont i.i.d. selon $\mathcal{N}(0, \sigma^2)$.
- On en déduit la relation suivante :

$$P(Y|X; \mathbf{a}, \sigma^2) \sim \mathcal{N}(\mathbf{X}^\top \mathbf{a}, \sigma^2)$$

- L'étude de la régression linéaire multiple dans un cadre probabiliste permet d'introduire le principe d'inférence de **maximum de vraisemblance (MV)** ainsi que des propriétés statistiques des estimateurs associés.

Régression linéaire multiple et MCO (suite)

- Pour apprécier la plus ou moins grande adéquation du modèle linéaire vis à vis des données, on peut calculer l'erreur quadratique relative et/ou le coefficient de détermination :

$$scr_{rel}(f) = \frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{\sum_{i=1}^n (y_i - \bar{y})^2} ; \quad coef_{det}(f) = 1 - scr_{rel}$$

- $coef_{det}(f)$ est également appelé le “ R^2 ” : s'il est proche de 0 cela veut dire que le modèle estimé ne marche pas mieux que la moyenne \bar{y} .
- **Attention !** Le “ R^2 ” augmente naturellement si le nombre de variables explicatives augmente donc il ne permet pas de comparer des modèles linéaires n'ayant pas le même nombre de variables. Dans ce cas, on utilisera le “ R^2 ajusté”.
- Par ailleurs, il existe d'autres procédures statistiques permettant de valider ou non le modèle estimé notamment lorsque nous nous plaçons dans un cadre gaussien.

Régression linéaire multiple et modèle gaussien (suite)

- La **vraisemblance (“likelihood”)** est la probabilité d'observer l'échantillon :

$$vr(\mathbf{a}, \sigma^2) = P(Y_1, \dots, Y_n | X_1, \dots, X_n; \mathbf{a}, \sigma^2)$$

- Les Y_i sont supposés i.i.d. nous avons alors :

$$\begin{aligned} vr(\mathbf{a}, \sigma^2) &= \prod_{i=1}^n P(Y_i | X_i; \mathbf{a}, \sigma^2) \\ &= \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2} \left(\frac{Y_i - X_i^\top \mathbf{a}}{\sigma}\right)^2\right) \end{aligned}$$

- L'**estimateur du MV** est la valeur des paramètres qui maximise la probabilité d'observer l'échantillon. On résoud donc le problème :

$$\max_{(\mathbf{a}, \sigma^2) \in \mathbb{R}^{p+1} \times \mathbb{R}} \prod_{i=1}^n P(Y_i | X_i; \mathbf{a}, \sigma^2)$$

Régression linéaire multiple et modèle gaussien (suite)

- Il est plus commode de maximiser, de manière équivalente, le **logarithme de la vraisemblance** :

$$lvr(\mathbf{a}, \sigma^2) = \log\left(\prod_{i=1}^n P(Y_i|X_i; \mathbf{a}, \sigma^2)\right) = \sum_{i=1}^n \log(P(Y_i|X_i; \mathbf{a}, \sigma^2))$$

- Dans le modèle gaussien cela se réduit à :

$$\begin{aligned} lvr(\mathbf{a}, \sigma^2) &= \sum_{i=1}^n \log\left(\frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2}\left(\frac{Y_i - X_i^\top \mathbf{a}}{\sigma}\right)^2\right)\right) \\ &= \sum_{i=1}^n \left(-\log(\sqrt{2\pi\sigma^2}) - \frac{1}{2}\left(\frac{Y_i - X_i^\top \mathbf{a}}{\sigma}\right)^2\right) \\ &= -\frac{n}{2} \log(2\pi) - \frac{n}{2} \log(\sigma^2) - \frac{1}{2\sigma^2} \sum_{i=1}^n (Y_i - X_i^\top \mathbf{a})^2 \end{aligned}$$

- Nous avons la propriété suivante : $\max lvr(\mathbf{a}, \sigma^2) \Leftrightarrow \min scr(f)$

Rappels de probabilités

- Nous rappelons quelques propriétés de l'opérateur variance.
- Supposons que X soit un vecteur aléatoire et que \mathbf{b} et \mathbf{A} soient respectivement un vecteur et une matrice carrée donnés :
 - Propriétés de linéarité de l'espérance :

$$E_X(\mathbf{A}X + \mathbf{b}) = \mathbf{A}E_X(X) + \mathbf{b}$$

- Propriété de la variance :

$$V_X(\mathbf{A}X + \mathbf{b}) = \mathbf{A}V_X(X)\mathbf{A}^\top$$

Régression linéaire multiple et modèle gaussien (suite)

- Estimateur du MV :

$$\mathbf{a}_{mv} = \arg \max_{\mathbf{a} \in \mathbb{R}^{p+1}} \sum_{i=1}^n \log(P(Y_i|X_i; \mathbf{a}, \sigma^2))$$

- On a la solution analytique :

$$\mathbf{a}_{mv} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{Y}$$

- Espérance de \mathbf{a}_{mv} :

$$\begin{aligned} E_{Y|X}(\mathbf{a}_{mv}|\mathbf{X}) &= E_{Y|X}\left(\left(\mathbf{X}^\top \mathbf{X}\right)^{-1} \mathbf{X}^\top Y|\mathbf{X}\right) \\ &= \left(\mathbf{X}^\top \mathbf{X}\right)^{-1} \mathbf{X}^\top E_{Y|X}(Y|\mathbf{X}) \\ &= \left(\mathbf{X}^\top \mathbf{X}\right)^{-1} \mathbf{X}^\top \mathbf{X} \mathbf{a} = \mathbf{a} \end{aligned}$$

- L'estimateur du MV est donc **sans biais**.

Régression linéaire multiple et modèle gaussien (suite)

- Variance de $\hat{\mathbf{a}}_{mv}$:

$$\begin{aligned} V_{Y|X}(\mathbf{a}_{mv}|\mathbf{X}) &= V_{Y|X}\left(\left(\mathbf{X}^\top \mathbf{X}\right)^{-1} \mathbf{X}^\top Y|\mathbf{X}\right) \\ &= \left(\mathbf{X}^\top \mathbf{X}\right)^{-1} \mathbf{X}^\top V_{Y|X}(Y|\mathbf{X}) \mathbf{X} \left(\mathbf{X}^\top \mathbf{X}\right)^{-1} \\ &= \sigma^2 \left(\mathbf{X}^\top \mathbf{X}\right)^{-1} \end{aligned}$$

- Efficacité** de l'estimateur du MV :

Théorème. (Théorème de Gauss-Markov)

Sous l'hypothèse que le vecteur des résidus $\epsilon = (\epsilon_1, \dots, \epsilon_n)$ vérifie $E_\epsilon(\epsilon) = \mathbf{0}$ (espérance nulle) et $V_\epsilon = \sigma^2 \mathbf{I}_n$ (variance constante et non-corrélation), l'estimateur du MV (ou MCO) $\mathbf{a}_{mv} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{Y}$ est, parmi les estimateurs linéaires (çàd fonctions linéaires des $\{Y_i\}$) qui soient sans biais, celui de variance minimale.

Régression linéaire multiple (code R)

```
> X=read.table(file='houses_selling_prices.txt',header=TRUE)
> names(X)
[1] "nb_bathrooms"      "area_site"          "size_living_space" "nb_garages"
[5] "nb_rooms"          "nb_bedrooms"        "age_in_years"      "nb_fire_places"
[9] "selling_price"
> summary(X)
  nb_bathrooms  area_site  size_living_space  nb_garages  nb_rooms
Min.   ~:1.000   Min.   ~: 2.275   Min.   ~:0.975   Min.   ~:0.000   Min.   ~: 5.000
1st Qu.:1.000   1st Qu.: 4.855   1st Qu.:1.194   1st Qu.:1.000   1st Qu.: 6.000
Median~:1.000   Median~: 6.143   Median~:1.494   Median~:1.250   Median~: 6.000
Mean   ~:1.268   Mean   ~: 6.461   Mean   ~:1.512   Mean   ~:1.339   Mean   ~: 6.679
3rd Qu.:1.500   3rd Qu.: 7.850   3rd Qu.:1.655   3rd Qu.:2.000   3rd Qu.: 7.000
Max.   ~:2.500   Max.   ~:12.800   Max.   ~:3.420   Max.   ~:2.000   Max.   ~:10.000
...
> ols_fit=lm(selling_price ~ .,data=X)
```

Régression linéaire multiple (code R)

```
> cbind(coef(ols_fit))
              [,1]
(Intercept)  5.33673015
nb_bathrooms 12.63522786
area_site    0.08367513
size_living_space 14.09124293
nb_garages   2.68059314
nb_rooms     0.26409588
nb_bedrooms  -2.17602140
age_in_years -0.11477224
nb_fire_places 2.87254730
```

Régression linéaire multiple (code R)

```
> summary(ols_fit)

Call:
lm(formula = selling_price ~ ., data = X)

Residuals:
    Min       1Q   Median       3Q      Max
-7.329 -2.532 -0.113  2.670  7.437

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)    5.33673    7.37355   0.724  0.47803
nb_bathrooms   12.63523    5.11350   2.471  0.02311 *
area_site       0.08368    0.54402   0.154  0.87938
size_living_space 14.09124    4.67933   3.011  0.00718 **
...
Residual standard error: 4.266 on 19 degrees of freedom
Multiple R-squared:  0.9361, Adjusted R-squared:  0.9092
F-statistic: 34.79 on 8 and 19 DF,  p-value: 9.443e-10
```

Régularisation des modèles de régression linéaire

- L'estimateur du MV ou des MCO est de variance minimale parmi les estimateurs linéaires sans biais. Néanmoins, la variance aboutit dans certains cas à des erreurs de prédiction fortes. Dans ce cas, on cherche des **estimateurs de variance plus petite quite à avoir un léger biais**. On peut pour cela supprimer l'effet de certaines variables explicatives ce qui revient à leur attribuer un coefficient nul.
- Par ailleurs, dans le cas où p , le nombre de variables explicatives, est grand, l'interprétation des résultats obtenus par les MCO est parfois ardu. Ainsi, on pourra préférer un **modèle estimé avec moins de variables explicatives** afin de privilégier l'interprétation du phénomène sous-jacent aux données plutôt que la précision.
- On étudie ici des méthodes permettant de produire des estimateurs dont les valeurs sont d'amplitudes réduites. Notamment, on parle de **modèles parcimonieux** lorsque des variables ont des coefficients nuls.
- Dans ce qui suit nous verrons deux approches : la régression **ridge** et la régression **lasso**.

Régression ridge

- Nous sommes toujours dans le même contexte que précédemment et avons l'espace des hypothèses suivant :

$$\mathbb{H} = \{f : \mathbb{R}^p \rightarrow \mathbb{R} : f(X) = a_0 + \sum_{j=1}^p a_j X^j\}$$

- Soit $\mathbf{a}_{\setminus 0}$ le vecteur (a_1, \dots, a_p) .
- L'estimateur ridge noté $\hat{\mathbf{a}}_{ridge}$ est défini de la manière suivante :

$$\hat{\mathbf{a}}_{ridge} = \arg \min_{\mathbf{a} \in \mathbb{R}^{p+1}} \left\{ \sum_{i=1}^n \left(y_i - \left(a_0 + \sum_{j=1}^p a_j x_{ij} \right) \right)^2 + \lambda \|\mathbf{a}_{\setminus 0}\|_{\ell_2}^2 \right\}$$

- $R(\mathbf{a}_{\setminus 0}) = \|\mathbf{a}\|_{\ell_2}^2 = \sum_{j=1}^p a_j^2$ est appelé **fonction de pénalité**.
- λ est un réel positif ou nul qui permet de contrôler l'amplitude des valeurs $\{a_j\}_{j=1}^p$ (càd la norme du vecteur $\mathbf{a}_{\setminus 0}$). On parle de **coefficient de pénalité** ou de **"shrinkage"** (rétrécissement).
- Plus λ est grand plus la valeur des coefficients se rapproche de 0 et moins la variance de l'estimateur de $\mathbf{a}_{\setminus 0}$ est grande.

Régression ridge (suite)

- Contrairement à la régression linéaire multiple classique où on ne normalise pas nécessairement les variables, ici il est nécessaire de **réduire les variables explicatives** avant de résoudre le problème d'optimisation. En effet, si les variables sont dans des unités de mesures non commensurables le terme de pénalité (càd la contrainte) aura un impact non uniforme sur les X^j .
- En pratique, il faut également **centrer la matrice de données \mathbf{X}** à laquelle on enlève la première colonne remplie de 1. On supposera donc par la suite que la matrice \mathbf{X} est de taille $(n \times p)$ et est centrée-réduite, $\forall j = 1, \dots, p$:

$$\frac{1}{n} \sum_{i=1}^n x_{ij} = 0 \text{ et } \frac{1}{n} \sum_{i=1}^n x_{ij}^2 = 1$$

Régression ridge (suite)

- Une façon équivalente d'introduire la régression ridge est par le biais du problème d'optimisation contraint suivant :

$$\min_{\mathbf{a} \in \mathbb{R}^{p+1}} \sum_{i=1}^n \left(y_i - \left(a_0 + \sum_{j=1}^p a_j x_{ij} \right) \right)^2$$

$$s/c \quad \sum_{j=1}^p a_j^2 \leq \tau$$

- On montre qu'il existe une bijection entre λ et τ ce qui rend équivalent les deux problèmes.
- Cette formulation permet d'exprimer explicitement la contrainte sur l'amplitude des coefficients : on voit effectivement qu'il s'agit de minimiser $scr(f)$ avec la contrainte que $\mathbf{a}_{\setminus 0}$ appartienne à une boule de \mathbb{R}^p et de rayon τ .
- Géométriquement : si $\hat{\mathbf{a}}_{\setminus 0, mco}$ appartient à la boule alors $\hat{\mathbf{a}}_{\setminus 0, mco} = \hat{\mathbf{a}}_{\setminus 0, ridge}$ sinon, on projette $\hat{\mathbf{a}}_{\setminus 0, mco}$ sur la boule (pour satisfaire la contrainte).

Régression ridge (suite)

- On **centre le vecteur \mathbf{y}** également et on suppose par la suite :

$$\frac{1}{n} \sum_{i=1}^n y_i = 0$$

- L'ordonnée à l'origine a_0 n'intervient pas dans la fonction de pénalité car ceci rendrait la fonction de prédiction dépendante d'une ordonnée à l'origine que l'on trouverait pour Y .
- On montre en fait que si \mathbf{X} et \mathbf{y} sont centrés, on peut séparer l'estimation du modèle en deux étapes :
 - On prend $\hat{a}_{ridge,0} = \bar{y}$ (moyenne empirique avant centrage).
 - On estime (a_1, \dots, a_p) en résolvant :

$$\hat{\mathbf{a}}_{ridge} = \arg \min_{\mathbf{a} \in \mathbb{R}^p} \left\{ \sum_{i=1}^n \left(y_i - \sum_{j=1}^p a_j x_{ij} \right)^2 + \lambda \|\mathbf{a}\|^2 \right\}$$

où $\mathbf{a} = (a_1, \dots, a_p) \in \mathbb{R}^p$.

Régression ridge (suite)

- L'écriture matricielle de la fonction objectif devient :

$$\hat{\mathbf{a}}_{\text{ridge}} = \arg \min_{\mathbf{a} \in \mathbb{R}^p} \left\{ (\mathbf{y} - \mathbf{X}\mathbf{a})^\top (\mathbf{y} - \mathbf{X}\mathbf{a}) + \lambda \mathbf{a}^\top \mathbf{a} \right\}$$

- On a la **solution analytique** suivante :

$$\hat{\mathbf{a}}_{\text{ridge}} = (\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I}_p)^{-1} \mathbf{X}^\top \mathbf{y}$$

- Les prédictions sont alors données par :

$$\hat{\mathbf{y}} = \underbrace{\bar{\mathbf{y}}}_{\hat{\mathbf{a}}_{\text{ridge},0}} \mathbf{1}_n + \mathbf{X} \hat{\mathbf{a}}_{\text{ridge}} \quad \text{et} \quad \hat{f}(\mathbf{x}) = \underbrace{\bar{\mathbf{y}}}_{\hat{\mathbf{a}}_{\text{ridge},0}} + \mathbf{x}^\top \hat{\mathbf{a}}_{\text{ridge}}$$

où \mathbf{x} est un vecteur quelconque de taille $(p \times 1)$ et \mathbf{x} est de terme général : $x_j = \frac{x_j - \bar{x}^j}{\hat{\sigma}_{x_j}}$.

Régression ridge (suite)

- Espérance de $\mathbf{a}_{\text{ridge}}$ (suite) :

$$\begin{aligned} E_{Y|X}(\mathbf{a}_{\text{ridge}}|\mathbf{X}) &= (\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I}_p)^{-1} \mathbf{X}^\top \mathbf{X} \mathbf{a} \\ &= (\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I}_p)^{-1} (\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I}_p - \lambda \mathbf{I}_p) \mathbf{a} \\ &= \left(\mathbf{I}_p - \lambda (\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I}_p)^{-1} \right) \mathbf{a} \\ &= \mathbf{a} - \underbrace{\lambda (\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I}_p)^{-1} \mathbf{a}}_{\text{biais}} \end{aligned}$$

- L'estimateur ridge de \mathbf{a} n'est donc pas sans biais et de ce point de vue il est moins bon que l'estimateur des MCO.

Régression ridge (suite)

- Supposons que $\hat{\mathbf{a}}_{\text{mco}}$ est l'estimation des MCO sur \mathbf{X} et \mathbf{y} . Nous avons donc : $\mathbf{a}_{\text{mco}} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{Y}$, $E_{Y|X}(\mathbf{a}_{\text{mco}}|\mathbf{X}) = \mathbf{a}$ et $V_{Y|X}(\mathbf{a}_{\text{mco}}|\mathbf{X}) = \sigma^2 (\mathbf{X}^\top \mathbf{X})^{-1}$
- Espérance de $\mathbf{a}_{\text{ridge}}$:

$$\begin{aligned} E_{Y|X}(\mathbf{a}_{\text{ridge}}|\mathbf{X}) &= E_{Y|X} \left((\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I}_p)^{-1} \mathbf{X}^\top \mathbf{Y} | \mathbf{X} \right) \\ &= E_{Y|X} \left((\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I}_p)^{-1} \mathbf{X}^\top \mathbf{X} \mathbf{a}_{\text{mco}} | \mathbf{X} \right) \\ &= (\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I}_p)^{-1} \mathbf{X}^\top \mathbf{X} E_{Y|X}(\mathbf{a}_{\text{mco}}|\mathbf{X}) \\ &= (\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I}_p)^{-1} \mathbf{X}^\top \mathbf{X} \mathbf{a} \end{aligned}$$

Régression ridge (suite)

- Variance de $\mathbf{a}_{\text{ridge}}$:

$$\begin{aligned} V_{Y|X}(\mathbf{a}_{\text{ridge}}|\mathbf{X}) &= V_{Y|X} \left((\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I}_p)^{-1} \mathbf{X}^\top \mathbf{X} \mathbf{a}_{\text{mco}} | \mathbf{X} \right) \\ &= (\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I}_p)^{-1} \mathbf{X}^\top \mathbf{X} V_{Y|X}(\mathbf{a}_{\text{mco}}|\mathbf{X}) \mathbf{X}^\top \mathbf{X} (\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I}_p)^{-1} \\ &= \sigma^2 (\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I}_p)^{-1} \mathbf{X}^\top \mathbf{X} (\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I}_p)^{-1} \end{aligned}$$

- Les vecteurs propres de $\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I}_p$ sont les mêmes que ceux de $\mathbf{X}^\top \mathbf{X}$.
- Mais les valeurs propres de $\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I}_p$ sont plus grandes que celles de $\mathbf{X}^\top \mathbf{X}$.
- On en déduit que la variance de l'estimateur de $\mathbf{a}_{\text{ridge}}$ est plus petite que celle de \mathbf{a}_{mco} . De ce point de vue, on peut attendre de la régression ridge qu'elle donne des prédictions meilleures que celles de la régression linéaire classique sur des données non observées.

Régression ridge (suite)

- Interprétation à partir de la décomposition en valeurs singulières de \mathbf{X} .
- Soit $\mathbf{X} = \mathbf{UDV}^\top$ la svd de \mathbf{X} où :
 - ▶ \mathbf{U} de taille $(n \times p)$ comporte en colonnes les vecteurs représentant une base orthonormée du sous-espace engendré par les colonnes de \mathbf{X} .
 - ▶ \mathbf{V} de taille $(p \times p)$ comporte en colonnes les vecteurs représentant une base orthonormée du sous-espace engendré par les lignes de \mathbf{X} .
 - ▶ \mathbf{D} de taille $(p \times p)$ comporte sur sa diagonale les valeurs singulières de \mathbf{X} : $d_1 \geq d_2 \geq \dots \geq d_p \geq 0$.
- La prédiction par l'estimateur des MCO est $\hat{\mathbf{y}} = \mathbf{X}\hat{\mathbf{a}}_{mco}$:

$$\begin{aligned}\hat{\mathbf{y}} &= \mathbf{X}(\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y} \\ &= \mathbf{UDV}^\top ((\mathbf{UDV}^\top)^\top \mathbf{UDV}^\top)^{-1} (\mathbf{UDV}^\top)^\top \mathbf{y} \\ &= \mathbf{UDV}^\top (\mathbf{VDU}^\top \mathbf{UDV}^\top)^{-1} \mathbf{VDU}^\top \mathbf{y} \\ &= \mathbf{UU}^\top \mathbf{y} \text{ (en remarquant que } \mathbf{U}^\top \mathbf{U} = \mathbf{I}_p \text{)} \\ &= \sum_{j=1}^p \mathbf{u}^j (\mathbf{u}^j)^\top \mathbf{y} \text{ (où } \mathbf{u}^j \text{ est la } j\text{ème colonne de } \mathbf{U} \text{)}\end{aligned}$$

Régression ridge (suite)

- Comment choisir la valeur de λ , le coefficient de pénalité ?
- L'approche simple consiste à prendre une séquence de nombres \mathbb{S} allant de 0 jusqu'à un nombre positif maximal, on remplace λ par chacune de ses valeurs, on teste itérativement ces différents modèles (en utilisant de la validation croisée sur un ensemble de validation notamment) et on sélectionne à la fin la valeur de λ ayant donné le meilleur modèle selon un critère.
- Il existe en fait des algorithmes efficaces (utilisant la SVD) permettant de déterminer pour toute valeur λ les valeurs des différents coefficients $\hat{\mathbf{a}}_{ridge}$. On parle alors de **chemin de régularisation** ("regularization path" ou "solution path").
- Ces algorithmes sont notamment implémentés dans la librairie `glmnet`.

Régression ridge (suite)

- Dans le cas de l'estimateur ridge la prédiction vaut $\hat{\mathbf{y}} = \mathbf{X}\hat{\mathbf{a}}_{ridge}$:

$$\begin{aligned}\hat{\mathbf{y}} &= \mathbf{X}(\mathbf{X}^\top \mathbf{X} + \lambda \mathbf{I}_p)^{-1} \mathbf{X}^\top \mathbf{y} = \mathbf{UD}(\mathbf{D}^2 + \lambda \mathbf{I}_p)^{-1} \mathbf{DU}^\top \mathbf{y} \\ &= \sum_{j=1}^p \frac{d_j^2}{d_j^2 + \lambda} \mathbf{u}^j (\mathbf{u}^j)^\top \mathbf{y}\end{aligned}$$

- Le terme $(\mathbf{u}^j)^\top \mathbf{y}$ est la j ème coordonnée de \mathbf{y} dans la base \mathbf{U} .
- La régression ridge diminue cette coordonnée d'un facteur $d_j^2 / (d_j^2 + \lambda) \leq 1$ par rapport à la régression classique.
- Plus d_j^2 est petit plus le ratio $d_j^2 / (d_j^2 + \lambda)$ est petit. Par ailleurs, rappelons que \mathbf{u}^j est le vecteur propre orthonormée de $\mathbf{X}^\top \mathbf{X}$ associé à la valeur propre d_j^2 . Ainsi, les coordonnées sur les axes de variance petite (d_j^2 petit) sont davantage pénalisées par la régression ridge.
- Remarque : l'expression précédente utilisant la SVD de \mathbf{X} permet de calculer facilement les estimations du modèle quand λ varie.

Régression ridge (code R)

- Plusieurs librairies R implémentent la régression ridge (MASS).
- Nous utiliserons le package et la fonction `glmnet`².
- Le cas ridge correspond au paramètre $\alpha = 0$.

#Format des données

```
selling_price=X$selling_price
```

```
X=as.matrix(X[, -ncol(X)])
```

#Régression pénalisée

```
library(glmnet)
```

#ridge

```
ridge_fit=glmnet(x=X,y=selling_price,family = "gaussian", alpha=0)
```

```
plot(ridge_fit)
```

```
cv_ridge_fit=cv.glmnet(x=X,y=selling_price,family = "gaussian", alpha=0)
```

```
plot(cv_ridge_fit)
```

```
cv_ridge_fit$lambda.min
```

```
coef(cv_ridge_fit,s = "lambda.min")
```

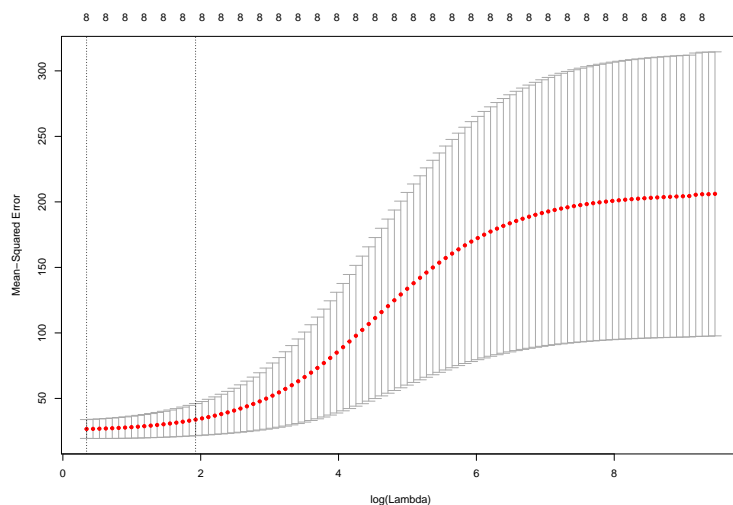
2. https://web.stanford.edu/~hastie/glmnet/glmnet_alpha.html

Régression ridge (code R)

```
> cv_ride_fit$lambda.min
[1] 1.411404
> coef(cv_ride_fit,s = "lambda.min")
9 x 1 sparse Matrix of class "dgCMatrix"
      1
(Intercept)      2.4751813
nb_bathrooms    12.2056258
area_site       0.3146111
size_living_space 9.9314174
nb_garages      1.9833801
nb_rooms        0.7228990
nb_bedrooms     -0.1651591
age_in_years    -0.1319908
nb_fire_places  3.1537881
```

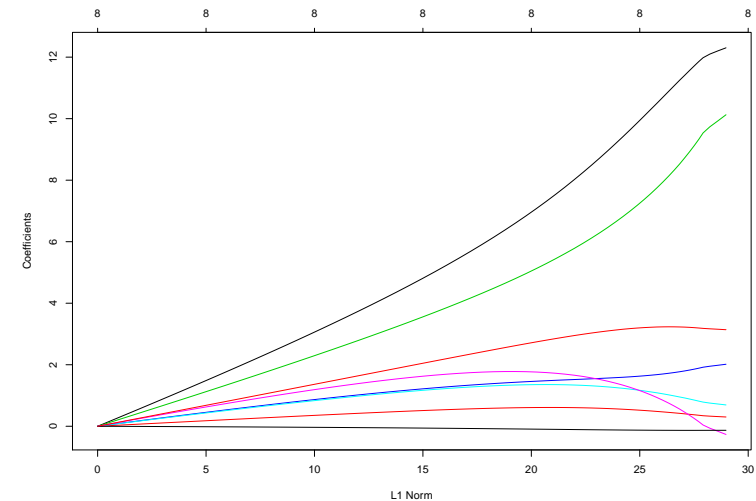
Régression ridge (sorties graphiques)

```
plot(cv_ride_fit)
```



Régression ridge (sorties graphiques)

```
plot(ridge_fit)
```



Régression ridge (sorties graphiques)

- `glmnet` suggère deux valeurs de λ à l'issue du calcul du chemin de régularisation (deux barres verticales en pointillé) :
 - ▶ `lambda.min` est la valeur de λ donnant la plus faible mse.
 - ▶ `lambda.1se` est une valeur plus grande que `lambda.min` et c'est la plus petite valeur de λ restant à une unité d'écart-type du modèle obtenu avec `lambda.min`.
- Comme `lambda.1se > lambda.min`, il correspond à un modèle plus simple (car pénalisation plus forte) tout en gardant une erreur "comparable" à celle obtenue avec `lambda.min`.
- Intuitivement, `lambda.min` conduirait à un modèle faisant du sur-apprentissage et `lambda.1se`, de variance plus petite, donnerait ainsi une erreur en généralisation plus faible.
- On retrouve ici encore le concept de biais-variance.

Régression Lasso

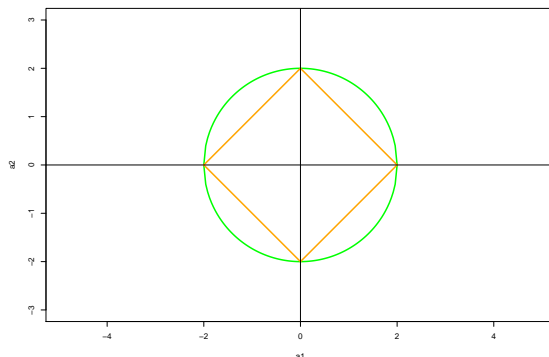
- Dans l'idée, la régression lasso est très proche de la régression ridge. L'estimateur lasso noté $\hat{\mathbf{a}}_{lasso}$ est défini de la manière suivante :

$$\hat{\mathbf{a}}_{lasso} = \arg \min_{\mathbf{a} \in \mathbb{R}^{p+1}} \left\{ \sum_{i=1}^n \left(y_i - \left(a_0 + \sum_{j=1}^p a_j x_{ij} \right) \right)^2 + \lambda \|\mathbf{a}_{\setminus 0}\|_{\ell_1} \right\}$$

- $R(\mathbf{a}_{\setminus 0}) = \|\mathbf{a}_{\setminus 0}\|_{\ell_1} = \sum_{j=1}^p |a_j|$ est une **fonction de pénalité** basée sur la norme ℓ_1 plutôt que la norme ℓ_2 comme c'est le cas pour la régression ridge.
- Le problème précédent est aussi équivalent au problème d'optimisation contraint :

$$\begin{aligned} \min_{\mathbf{a} \in \mathbb{R}^{p+1}} & \sum_{i=1}^n \left(y_i - \left(a_0 + \sum_{j=1}^p a_j x_{ij} \right) \right)^2 \\ \text{s.t.} & \sum_{j=1}^p |a_j| \leq \tau \end{aligned}$$

Comparaison ridge et lasso - domaines de recherche



- En **vert** la frontière ridge : $a_1^2 + a_2^2 = 4$.
- En **orange** la frontière lasso : $|a_1| + |a_2| = 2$.

Régression lasso (suite)

- La différence de norme dans la fonction de pénalité a en fait un impact important. Il existe ainsi des différences fortes entre régression lasso et régression ridge :
- Contrairement à la régression ridge, il n'y a **pas de solution analytique** car la valeur absolue rend le problème non différentiable.
- On a donc recours à des méthodes d'optimisation numérique ou à des algorithmes spécifiques (par exemple : "Least Angle Regression - Stagewise" [Efron et al., 2004]).
- Quand τ est relativement petit, la solution obtenue par **la régression lasso est parcimonieuse** c'est-à-dire que certains coefficients estimés seront nuls. La régression lasso peut ainsi être vue comme une **méthode de sélection de variables**. Il s'agit d'un modèle davantage parcimonieux que les modèles précédents. Lasso : "Least Absolute Shrinkage and Selection Operator".
- Quand $\tau \geq \|\hat{\mathbf{a}}_{mco}\|_{\ell_1}$ alors $\hat{\mathbf{a}}_{lasso} = \hat{\mathbf{a}}_{mco}$.

Régression lasso (suite)

- En pratique, comme pour la régression ridge, on centre-réduit \mathbf{X} et on centre \mathbf{y} . \mathbf{X} et \mathbf{y} étant centrés, on peut à nouveau séparer en deux l'inférence. On retrouve notamment dans ce cas $\hat{\mathbf{a}}_{lasso,0} = \bar{\mathbf{y}}$.
- En prenant $\mathbf{a} = (a_1, \dots, a_p)$, l'estimateur lasso est donné par :

$$\hat{\mathbf{a}}_{lasso} = \arg \min_{\mathbf{a} \in \mathbb{R}^p} \left\{ \sum_{i=1}^n \left(y_i - \sum_{j=1}^p a_j x_{ij} \right)^2 + \lambda \|\mathbf{a}\|_{\ell_1} \right\}$$

- Les prédictions sont calculées de la façon suivante :

$$\hat{\mathbf{y}} = \underbrace{\bar{\mathbf{y}}}_{\hat{\mathbf{a}}_{lasso,0}} \mathbf{1}_n + \mathbf{X} \hat{\mathbf{a}}_{lasso} \quad \text{et} \quad \hat{f}(\mathbf{x}) = \underbrace{\bar{\mathbf{y}}}_{\hat{\mathbf{a}}_{lasso,0}} + \mathbf{x}^T \hat{\mathbf{a}}_{lasso}$$

où \mathbf{x} est un objet quelconque et \mathbf{x} est un vecteur de taille $(p \times 1)$ et de terme général : $x_j = \frac{x_j - \bar{x}^j}{\hat{\sigma}_{xj}}$.

Régression lasso (suite)

- Comment choisir le coefficient de pénalité ?
- On considère le problème équivalent suivant :

$$\min_{\mathbf{a} \in \mathbb{R}^p} \sum_{i=1}^n \left(y_i - \sum_{j=1}^p a_j x_{ij} \right)^2$$

$$\text{s/c } \frac{\sum_{j=1}^p |a_j|}{\|\hat{\mathbf{a}}_{mco}\|_{\ell_1}} \leq \tau$$

où $\|\hat{\mathbf{a}}_{mco}\|_{\ell_1}$ est une constante pré-calculée.

- Une méthode consiste alors à faire varier τ de 0 à 1. On voit que lorsque τ vaut 1 on a $\hat{\mathbf{a}}_{lasso} = \hat{\mathbf{a}}_{mco}$.
- On peut alors procéder par validation croisée comme pour ridge. Cependant, le problème n'ayant pas de solution analytique la détermination du chemin de régularisation semble plus ardue. Néanmoins, l'étude des propriétés du problème lasso a permis de mettre en place des algorithmes efficaces [Efron et al., 2004, Friedman et al., 2010].

Least Angle Regression - Stagewise (lars)

- Nous avons cité précédemment l'algorithme lars. Il s'agit d'un algorithme efficace permettant de déterminer le chemin de régularisation du lasso c'est-à-dire l'ensemble des solutions $\hat{\mathbf{a}}_{lasso}(\lambda)$ pour $\lambda \in [0, \infty]$ [Efron et al., 2004].
- $\hat{\mathbf{a}}_{lasso}(\lambda)$ est linéaire par morceau (cf illustration slide ci-après).
- L'algorithme est proche d'une méthode de recherche pas à pas ascendante dite "**forward stagewise regression**" où on cherche itérativement la variable la plus corrélée avec le vecteur des résidus.
- lars commence avec $\lambda = \infty$ et dans ce cas $\hat{\mathbf{a}}_{lasso} = \mathbf{0}$. Puis il détermine itérativement la valeur λ (qui décroît) permettant de faire entrer une variable dans l'ensemble actif (c'est-à-dire tel que le coefficient est non nul). Lorsque $\lambda = 0$ on obtient la solution des MCO.
- L'algorithme a une complexité cubique en p . Plus récemment des méthodes d'optimisation numérique ("cyclic coordinate descent") ont montré de meilleures performances que lars [Friedman et al., 2010].

Procédures classiques de sélection de modèle

- Le lasso permet de faire de la **sélection de modèles** mais rappelons au préalable qu'il existe des techniques simples dans ce cas :
 - ▶ **Recherche exhaustive** du meilleur sous-ensemble de variables. Si p est le nombre d'attributs, il y a 2^p possibilités (impraticable si p est grand).
 - ▶ **Recherche pas à pas** (approche gloutonne, localement optimale) :
 - ★ **ascendante** : on ajoute itérativement la variable qui permet d'améliorer le plus un critère de sélection de modèles,
 - ★ **descendante** : on part du modèle avec p variables et on enlève itérativement la variable qui permet d'améliorer le plus un critère de sélection de modèles.
- Les **critères de sélection de modèles** sont de plusieurs sortes :
 - ▶ R^2 ajusté,
 - ▶ C_p de Mallows,
 - ▶ le critère AIC (Akaike Information Criterion),
 - ▶ le critère BIC (Bayesian Information Criterion) ...

Régression lasso (code R)

- Plusieurs bibliothèques R implémentent la régression lasso (lars).
- Nous utiliserons le package et la fonction `glmnet` (qui implémente la méthode d'optimisation "cyclic coordinate descent").
- Le cas lasso correspond au paramètre $\alpha = 1$ (par défaut).

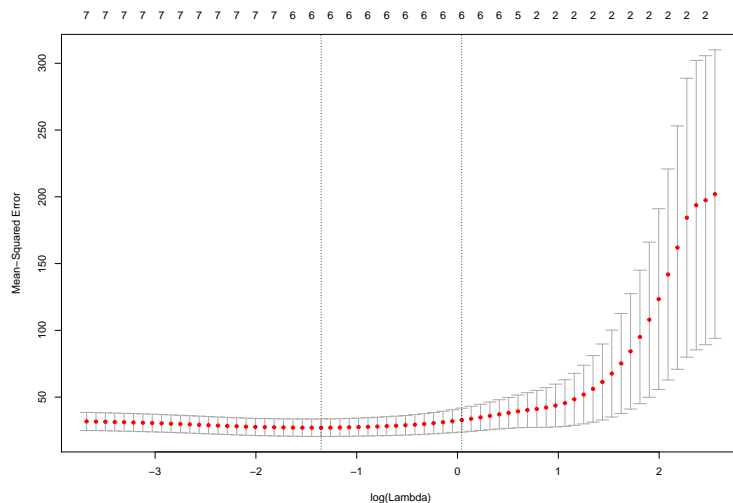
```
#lasso
lasso_fit=glmnet(x=X,y=selling_price,family = "gaussian", alpha=1)
plot(lasso_fit)
cv_lasso_fit=cv.glmnet(x=X,y=selling_price,family = "gaussian", alpha=1)
plot(cv_lasso_fit)
cv_lasso_fit$lambda.min
coef(cv_lasso_fit,s = "lambda.min")
```


Régression lasso (code R)

```
> cv_lasso_fit$lambda.min
[1] 0.1118513
> coef(cv_lasso_fit,s = "lambda.min")
9 x 1 sparse Matrix of class "dgCMatrix"
      1
(Intercept)      5.09977407
nb_bathrooms    12.29607390
area_site       0.05614472
size_living_space 13.17482462
nb_garages      2.19026979
nb_rooms        .
nb_bedrooms     -0.64604491
age_in_years    -0.12713019
nb_fire_places  3.09783476
```

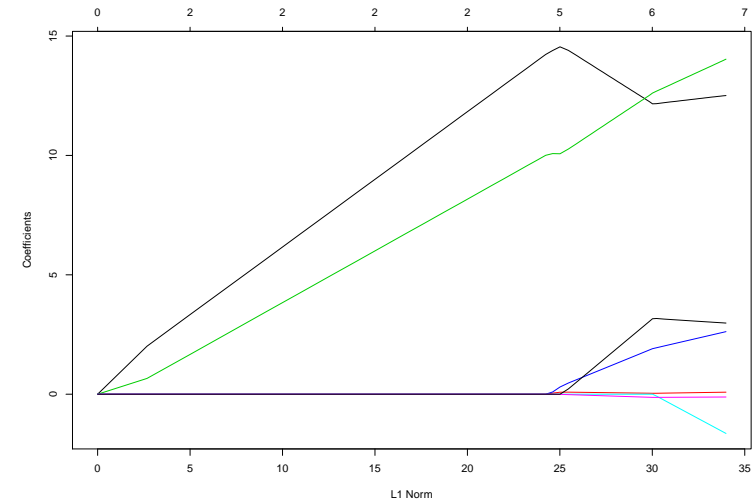
Régression lasso (code R)

```
plot(cv_lasso_fit)
```



Régression lasso (code R)

```
plot(lasso_fit)
```



Petite synthèse des régressions pénalisées ridge et lasso

- Les régressions ridge et lasso tentent de **diminuer la variance** des estimateurs (au prix d'un biais) afin d'obtenir une plus faible erreur en généralisation.
- Du point de vue optimisation on contraint la norme du vecteur des coefficients de respecter une borne supérieure (on parle de rétrécissement, "shrinkage"). Si on utilise la norme ℓ_2 on obtient le modèle ridge et si on utilise la norme ℓ_1 on obtient le modèle lasso.
- L'utilisation des normes ℓ_1 ou ℓ_2 donne des solutions bien différentes malgré le même but recherché : la solution lasso est parcimonieuse contrairement à la solution ridge.
- En théorie, la régression lasso est intéressante puisqu'elle permet à la fois une erreur en généralisation plus faible et une solution parcimonieuse.
- En pratique, elle est performante mais connaît des limites dans certaines situations.

Limites de la régression lasso

- Quand $p > n$ (données de grande dimension), la méthode lasso ne sélectionne au plus que n variables (en raison de la nature même du modèle d'optimisation sous-jacent).
- Si plusieurs variables sont corrélées entre elles, la méthode lasso ne sélectionnera qu'une seule d'entre elles et ignorera les autres.
- Dans de nombreux cas classiques avec $n > p$, s'il y a de fortes corrélations entre les variables explicatives, on trouve empiriquement que la méthode ridge donne de meilleures performances que la méthode lasso.

La régression elasticnet

- L'estimateur elasticnet "naïf" (cf plus loin pour des précisions) est défini par [Zou and Hastie, 2005] :

$$\hat{\mathbf{a}}_{nen} = \arg \min_{\mathbf{a} \in \mathbb{R}^p} \sum_{i=1}^n \left(y_i - \sum_{j=1}^p a_j x_{ij} \right)^2 + \lambda \left((1 - \alpha) \|\mathbf{a}\|_{\ell_1} + \alpha \|\mathbf{a}\|_{\ell_2}^2 \right)$$

où $R(\mathbf{a}) = (1 - \alpha) \|\mathbf{a}\|_{\ell_1} + \alpha \|\mathbf{a}\|_{\ell_2}^2$ est la **fonction de pénalité** de la régression elasticnet.

- Ce problème est équivalent au problème contraint suivant :

$$\min_{\mathbf{a} \in \mathbb{R}^p} \sum_{i=1}^n \left(y_i - \sum_{j=1}^p a_j x_{ij} \right)^2$$

$$\text{s.t. } (1 - \alpha) \sum_{j=1}^p |a_j| + \alpha \sum_{j=1}^p |a_j|^2 \leq \tau$$

- Si $\alpha = 1$ on a l'estimateur ridge et si $\alpha = 0$ on a l'estimateur lasso. Les cas nous intéressant sont donc ceux pour lesquels $0 < \alpha < 1$.

Mélange de ridge et de lasso

- On se place dans le même contexte que pour ridge et lasso où \mathbf{X} est centrée-réduite et \mathbf{y} est centrée.
- Pour surmonter les problèmes précédents, l'idée est de prendre un mélange de ridge et lasso. On obtient alors le problème suivant :

$$\min_{\mathbf{a} \in \mathbb{R}^p} \sum_{i=1}^n \left(y_i - \sum_{j=1}^p a_j x_{ij} \right)^2 + \lambda_1 \|\mathbf{a}\|_{\ell_1} + \lambda_2 \|\mathbf{a}\|_{\ell_2}^2$$

où λ_1 et λ_2 sont des paramètres positifs ou nuls.

- En posant $\alpha = \lambda_2 / (\lambda_1 + \lambda_2)$ on peut montrer que le problème est équivalent à :

$$\min_{\mathbf{a} \in \mathbb{R}^p} \sum_{i=1}^n \left(y_i - \sum_{j=1}^p a_j x_{ij} \right)^2 + \lambda \left((1 - \alpha) \|\mathbf{a}\|_{\ell_1} + \alpha \|\mathbf{a}\|_{\ell_2}^2 \right)$$

La régression elasticnet (suite)

Lemme.

Soit \mathbf{X} la matrice des variables explicatives de taille $(n \times p)$, et \mathbf{y} le vecteur de la variable cible réelle de taille n . Soit $(\lambda_1, \lambda_2) \in \mathbb{R}^+ \times \mathbb{R}^+$. Soient les données augmentées \mathbf{X}^+ et \mathbf{y}^+ de tailles respectives $((n+p) \times p)$ et $n+p$:

$$\mathbf{X}^+ = \frac{1}{\sqrt{1 + \lambda_2}} \begin{pmatrix} \mathbf{X} \\ \sqrt{\lambda_2} \mathbf{I}_p \end{pmatrix} \quad \text{et} \quad \mathbf{y}^+ = \begin{pmatrix} \mathbf{y} \\ \mathbf{0} \end{pmatrix}$$

Soit $\gamma = \lambda_1 / \sqrt{1 + \lambda_2}$. Alors la fonction objectif de la régression elasticnet peut s'écrire de façon équivalente :

$$\|\mathbf{y}^+ - \mathbf{X}^+ \mathbf{a}^+\|_{\ell_2}^2 + \gamma \|\mathbf{a}^+\|_{\ell_1}$$

Soit $\hat{\mathbf{a}}^+$ le minimiseur de cette fonction on a alors :

$$\hat{\mathbf{a}}_{nen} = \frac{1}{\sqrt{1 + \lambda_2}} \hat{\mathbf{a}}^+$$

La régression elasticnet

- Ce lemme de [Zou and Hastie, 2005] montre que la solution de la régression elasticnet peut être obtenue par la solution de la régression **lasso avec des données augmentées** !
- Comme \mathbf{X}^+ est de rang p , la solution elasticnet peut donc sélectionner potentiellement p variables contrairement à la régression lasso.
- Ce lemme permet également de montrer que la méthode elasticnet permet de faire de la **sélection de variables** comme la méthode lasso et contrairement à la méthode ridge.
- Dans le cas de grande dimension $n \ll p$, on observe souvent un **effet de groupes** entre variables qui ont tendance à être linéairement dépendantes. La régression elasticnet permet de tenir compte de cet effet : les variables fortement corrélées ont tendance à avoir la même valeur de coefficient dans $\hat{\mathbf{a}}_{nen}$.

Ré-échelonnement de l'estimateur elasticnet naïf

- L'estimateur elasticnet vu jusqu'à présent est dit **"naïf"**.
- En théorie, il permet de tenir compte des limites du lasso identifiées précédemment.
- En pratique, il ne donne satisfaction que lorsqu'il est proche de l'estimateur ridge ou de l'estimateur lasso.
- Ce comportement est en fait dû à un **double effet de rétrécissement** qui porte atteinte au modèle (on a une faible diminution de la variance pour une forte augmentation du biais).
- L'**estimateur elasticnet** $\hat{\mathbf{a}}_{en}$ retenu est alors un **ré-échelonnement** de la solution précédente :

$$\hat{\mathbf{a}}_{en} = (1 + \lambda_2)\hat{\mathbf{a}}_{nen} = \sqrt{1 + \lambda_2}\hat{\mathbf{a}}^+$$

- En pratique, l'estimateur ré-échelonné $\hat{\mathbf{a}}_{en}$ donne de meilleurs résultats pour $0 < \alpha < 1$ et peut ainsi surpasser le lasso.

Effet de groupe de la régression elasticnet

Théorème.

Soient \mathbf{X} et \mathbf{y} les données du problème de régression où les variables explicatives sont supposées centrées-réduites et la variable à expliquer centrée. Soit (λ_1, λ_2) des paramètres non négatifs. Soit $\hat{\mathbf{a}}_{nen}(\lambda_1, \lambda_2)$ la solution elasticnet naïve. Supposons que $\hat{a}_{nen,i}(\lambda_1, \lambda_2)\hat{a}_{nen,j}(\lambda_1, \lambda_2) > 0$ alors :

$$\frac{1}{\|\mathbf{y}\|_{\ell_1}} |\hat{a}_{nen,i}(\lambda_1, \lambda_2) - \hat{a}_{nen,j}(\lambda_1, \lambda_2)| \leq \frac{1}{\lambda_2} \sqrt{2(1 - \rho_{ij})}$$

où $\rho_{ij} = \langle \mathbf{x}^i, \mathbf{x}^j \rangle$ est le coefficient de corrélation entre X^i et X^j .

- Ce théorème de [Zou and Hastie, 2005] permet de caractériser l'effet de groupe d'elasticnet : l'écart entre les coefficients de deux variables est borné supérieurement par une grandeur qui dépend (inversement) de la corrélation linéaire entre celles-ci.

Elasticnet vue comme stabilisation du lasso

- On peut écrire la fonction objectif de l'estimateur lasso comme suit :

$$\hat{\mathbf{a}}_{lasso} = \arg \min_{\mathbf{a} \in \mathbb{R}^p} \mathbf{a}^\top (\mathbf{X}^\top \mathbf{X}) \mathbf{a} - 2\mathbf{y}^\top \mathbf{X} \mathbf{a} + \lambda_1 \|\mathbf{a}\|_{\ell_1}$$

- On montre que celle de l'estimateur elasticnet peut s'écrire :

$$\hat{\mathbf{a}}_{en} = \arg \min_{\mathbf{a} \in \mathbb{R}^p} \mathbf{a}^\top \left(\frac{\mathbf{X}^\top \mathbf{X} + \lambda_2 \mathbf{I}_p}{1 + \lambda_2} \right) \mathbf{a} - 2\mathbf{y}^\top \mathbf{X} \mathbf{a} + \lambda_1 \|\mathbf{a}\|_{\ell_1}$$

- Les variables étant centrées-réduites, $\mathbf{X}^\top \mathbf{X} = \hat{\Sigma}$ la matrice variance-covariance empirique. On a par ailleurs :

$$\frac{\mathbf{X}^\top \mathbf{X} + \lambda_2 \mathbf{I}_p}{1 + \lambda_2} = \gamma \hat{\Sigma} + (1 - \gamma) \mathbf{I}_p$$

en posant $\gamma = 1/(1 + \lambda_2)$.

- Elasticnet peut donc être vu comme un lasso où la matrice de variance-covariance est "rétrécie" c'est-à-dire proche de la matrice identité.

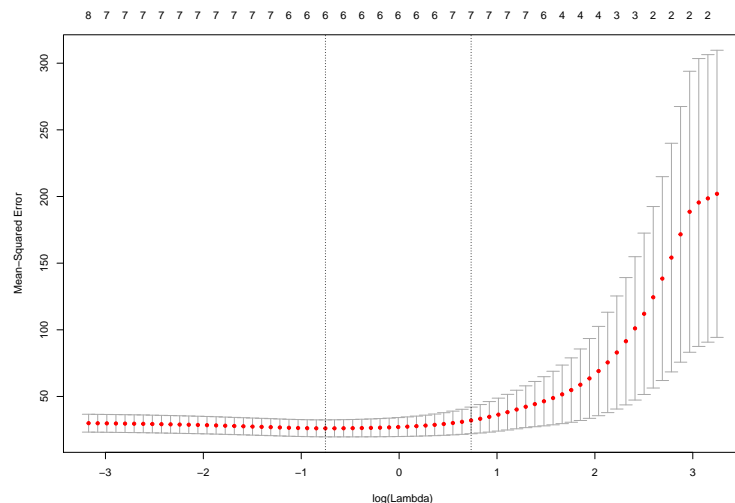
Régression elasticnet (code R)

```
#elasticnet
eln_fit=glmnet(x=X,y=selling_price,family = "gaussian", alpha=0.5)
plot(eln_fit)
cv_eln_fit=cv.glmnet(x=X,y=selling_price,family = "gaussian", alpha=0.5)
plot(cv_eln_fit)
cv_eln_fit$lambda.min
coef(cv_eln_fit,s = "lambda.min")

> cv_eln_fit$lambda.min
[1] 0.4708726
> coef(cv_eln_fit,s = "lambda.min")
9 x 1 sparse Matrix of class "dgCMatrix"
      1
(Intercept) 4.4651671
nb_bathrooms 12.5107837
area_site    0.1169648
size_living_space 11.9379818
nb_garages   1.9046787
nb_rooms    .
nb_bedrooms .
age_in_years -0.1231138
nb_fire_places 2.9517174
```

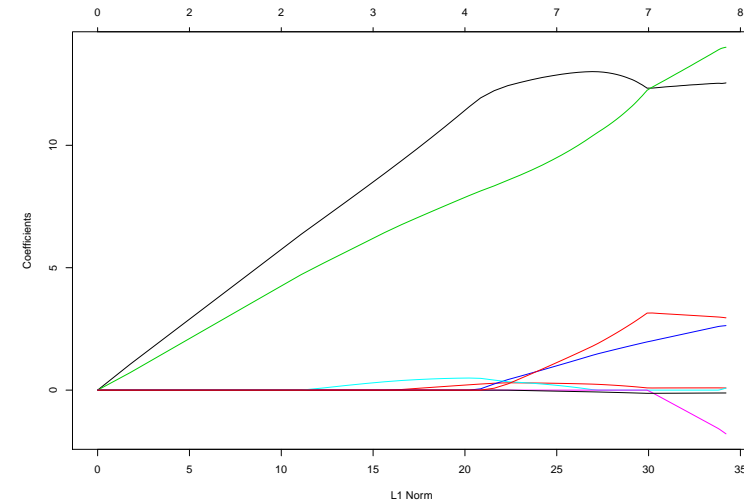
Régression elasticnet (code R)

```
plot(cv_eln_fit)
```



Régression elasticnet (code R)

```
plot(eln_fit)
```



Autres méthodes de régression

- Pour traiter les problèmes de singularité, au lieu de pénaliser les amplitudes des coefficients, d'autres méthodes cherchent à **transformer les variables** en de nouvelles qui sont appelées **composantes**. On utilise ensuite la régression linéaire multiple sur ces nouvelles composantes.
 - ▶ **Régression sur composantes principales** ("Principal Component Regression" ou régression **PCR**) : on pratique une ACP (Analyse en Composantes Principales) et on utilise un certain nombre de composantes principales à la place des variables initiales.
 - ▶ **Régression aux moindres carrés partiels** ("Partial Least Square Regression" ou régression **PLS**) : comme pour la régression PCR, on cherche des composantes qui sont des combinaisons linéaires des variables et qui soient orthogonales entre elles. Mais contrairement aux composantes principales qui tiennent compte de la variance entre variables, dans la régression PLS, on choisit ces composantes en tenant compte de leur pouvoir prédictif sur la variable cible.

Rappel du Sommaire

- 2 Les méthodes linéaires et leurs pénalisations (ridge, lasso, ...)
 - Méthodes linéaires pour la régression
 - Méthodes linéaires pour la catégorisation

Plusieurs types de méthodes de catégorisation

- \mathbf{Y} est une matrice concaténant des vecteurs binaires \mathbf{y}^l (variables indicatrices) avec $y_{il} = 1$ si $y_i = C_l$; $y_{il} = 0$ sinon.
- y_{il} peut s'interpréter comme la probabilité pour \mathbf{x}_i d'appartenir à C_l .
- On distingue plusieurs familles de méthodes de catégorisation [Bishop, 2006] :
 - ▶ **Fonctions discriminantes** : on apprend une fonction d'affectation f appartenant à un espace d'hypothèses \mathbb{H} qui, étant donné un \mathbf{x} , lui attribue une classe parmi \mathbb{Y} .
 - ▶ **Modèles (probabilistes) génératifs** : on estime $P(X|C_l)$ et $P(C_l)$ et on base la décision de catégorisation à l'aide de la probabilité a posteriori $P(C_l|X)$ (formule de Bayes) :

$$P(C_l|X) = \frac{P(X|C_l)P(C_l)}{P(X)}$$

- ▶ **Modèles (probabilistes) discriminatifs** : on estime directement la $P(C_l|X)$ sans passer par l'estimation de la densité jointe en estimant les paramètres d'une famille paramétrique \mathbb{H} .

Introduction

- Rappelons que pour le problème de catégorisation la variable cible à prédire est discrète : $Y \in \mathbb{Y}$ où $\mathbb{Y} = \{C_1, \dots, C_q\}$ avec $q \geq 2$.
- Nous étudions ici des méthodes qui sont dites linéaires étant donné qu'elles aboutissent à des **frontières de décision qui sont linéaires (hyperplans) en les variables explicatives** X^1, \dots, X^p .
- Ces variables explicatives peuvent être soit les variables initiales, soit une expansion de base des variables initiales (dans ce cas les frontières de décision sont non linéaires dans l'espace des variables initiales).
- La variable cible Y étant discrète, pour la représenter numériquement une méthode simple consiste à **transformer la variable discrète observée \mathbf{y} en une matrice binaire \mathbf{Y}** de taille $(n \times q)$ dont le terme général y_{il} est défini comme suit :

$$y_{il} = \begin{cases} 1 & \text{si } y_i = C_l \\ 0 & \text{sinon} \end{cases}$$

Fonction discriminante

- On cherche à étendre la régression linéaire multiple au cas discret.
- Rappelons dans un premier temps le cas binaire $\mathbb{Y} = \{C_1, C_2\}$ pour lequel on avait utilisé des variables artificielles $C_1 \leftrightarrow 1$ et $C_2 \leftrightarrow -1$.
- L'espace des hypothèses est $\mathbb{H} = \{g : \mathbb{R}^p \rightarrow \mathbb{R} : g(X) = a_0 + \sum_{j=1}^p a_j X^j\}$ qui peut s'écrire également par : $\mathbb{H} = \{g : \mathbb{R}^p \rightarrow \mathbb{R} : g(X) = a_0 + \mathbf{a}^\top X\}$ avec $\mathbf{a} = (a_1, \dots, a_p)$.
- La règle de décision est donnée par :

$$\hat{f}(\mathbf{x}) = \begin{cases} C_1 & \text{si } \hat{g}(\mathbf{x}) \geq 0 \\ C_2 & \text{sinon} \end{cases}$$

- La frontière de décision est donnée par $\hat{g}(\mathbf{x}) = 0$ et correspond à un hyperplan de dimension $p - 1$ dans \mathbb{X} .

Fonction discriminante (suite)

- On considère désormais le cas plus général d'un problème multiclassé : $\mathbb{Y} = \{C_1, \dots, C_q\}$, $q > 2$.
- Pour simplifier les notations considérons l'ajout de variables artificielles suivantes :

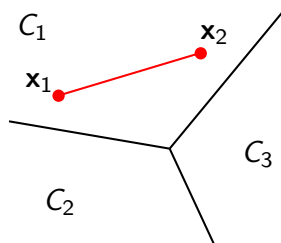
$$\mathbf{a} = \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_p \end{pmatrix} ; \quad \mathbf{X} = \begin{pmatrix} 1 & x_{11} & x_{12} & \dots & x_{1p} \\ 1 & x_{21} & x_{22} & \dots & x_{2p} \\ \vdots & \vdots & \dots & \dots & \vdots \\ 1 & x_{n1} & x_{n2} & \dots & x_{np} \end{pmatrix}$$

- Nous avons alors $\hat{g}(\mathbf{x}) = \hat{\mathbf{a}}^\top \mathbf{x}$ et $\hat{g}(\mathbf{x}) = 0$ est un hyperplan de dimension p de l'espace de représentation étendu à $p + 1$.

Fonction discriminante (suite)

- La règle de décision précédente donne lieu à des régions de catégorisation de chaque classe qui sont :

- ▶ connexes
- ▶ d'intersections vides
- ▶ convexes



Fonction discriminante (suite)

- Pour traiter efficacement le problème multiclassé, une façon de faire consiste à considérer une fonction vectorielle discriminante de dimension q , $\mathbf{g} : \mathbb{X} \rightarrow \mathbb{R}^q$, où la composante l de \mathbf{g} s'écrit :

$$g_l(\mathbf{x}) = \mathbf{a}_l^\top \mathbf{x}$$

- Pour tout $l = 1, \dots, q$, g_l peut être vue telle une fonction de score de la classe C_l .
- La règle de décision est alors la suivante :

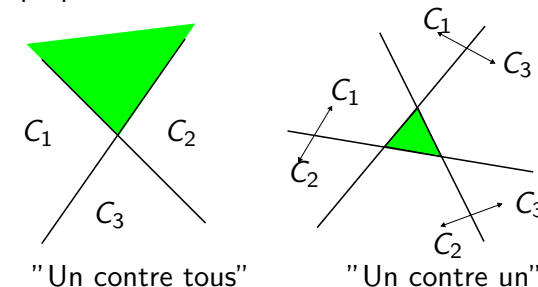
$$f(\mathbf{x}) = C_l \Leftrightarrow \forall l' \neq l : g_l(\mathbf{x}) \geq g_{l'}(\mathbf{x})$$

- La frontière de décision entre deux classes C_l et $C_{l'}$ est : $\{\mathbf{x} \in \mathbb{X} : g_l(\mathbf{x}) = g_{l'}(\mathbf{x})\}$. Il s'agit d'un hyperplan de dimension p défini par :

$$(\mathbf{a}_l - \mathbf{a}_{l'})^\top \mathbf{x} = 0$$

Fonction discriminante (suite)

- Remarque : des stratégies simples utilisant plusieurs classifieurs binaires telles que "un contre tous" ou "un contre un" ne possèdent pas de telles propriétés :



"Un contre tous"

"Un contre un"

- Autre remarque : il existe d'autres façons intéressantes de traiter le cas multiclassé comme par exemple l'approche ECOC ("Error Correcting Output Coding") [Dietterich and Bakiri, 1995].

Fonction discriminante basée sur les MCO

- Nous avons vu en introduction comment utiliser les MCO pour un problème de catégorisation binaire en utilisant des variables artificielles.
- Dans le cas général des problèmes multiclassés, on représente les différentes classes par des vecteurs binaires qui aboutit à la matrice binaire \mathbf{Y} introduite précédemment.
- Dans ce contexte, le critère scr est défini par :

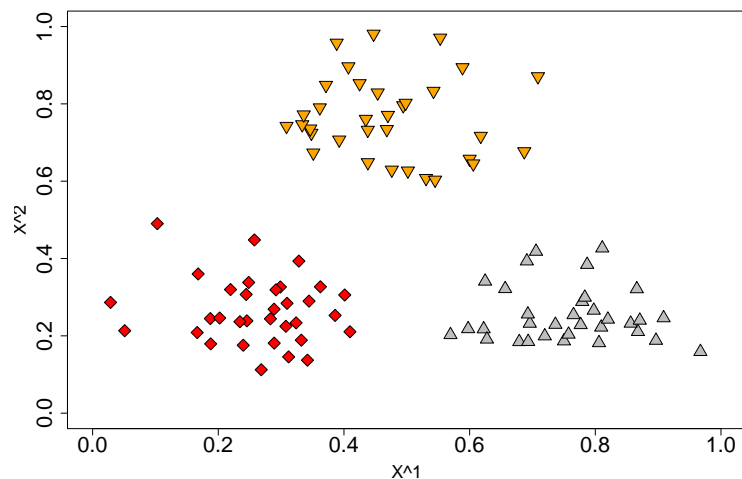
$$scr(g) = \sum_{i=1}^n \sum_{l=1}^q \left(y_{il} - \sum_{j=0}^p x_{ij} a_{lj} \right)^2$$

- L'écriture matricielle de la fonction de perte est :

$$scr(g) = tr \left((\mathbf{Y} - \mathbf{X}\mathbf{A})^\top (\mathbf{Y} - \mathbf{X}\mathbf{A}) \right)$$

où tr est l'opérateur trace d'une matrice carrée.

Fonction discriminante basée sur les MCO (exemple)



Fonction discriminante basée sur les MCO (suite)

- Formules de dérivation de l'opérateur trace :

$$\bullet \frac{\partial tr(\mathbf{A}\mathbf{X})}{\partial \mathbf{X}} = \mathbf{A}^\top$$

$$\bullet \frac{\partial tr(\mathbf{X}^\top \mathbf{A})}{\partial \mathbf{X}} = \mathbf{A}$$

$$\bullet \frac{\partial tr(\mathbf{X}^\top \mathbf{A}\mathbf{X})}{\partial \mathbf{X}} = \mathbf{A}\mathbf{X} + \mathbf{A}^\top \mathbf{X}$$

- En développant scr (tr est un opérateur linéaire) on obtient :

$$scr(g) = tr(\mathbf{A}^\top \mathbf{X}^\top \mathbf{X}\mathbf{A}) - tr(\mathbf{A}^\top \mathbf{X}^\top \mathbf{Y}) - tr(\mathbf{Y}^\top \mathbf{X}\mathbf{A}) + tr(\mathbf{Y}^\top \mathbf{Y})$$

- En appliquant les formules de dérivation on obtient :

$$\frac{\partial scr(g)}{\partial \mathbf{A}} = 2\mathbf{X}^\top \mathbf{X}\mathbf{A} - 2\mathbf{X}^\top \mathbf{Y}$$

- En posant $\frac{\partial scr(g)}{\partial \mathbf{A}} = 0$ on détermine les points critiques et il vient :

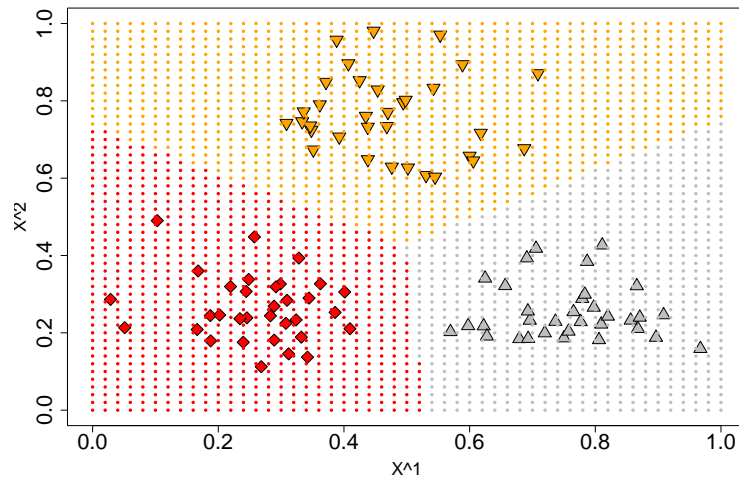
$$\hat{\mathbf{A}} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{Y}$$

Fonction discriminante basée sur les MCO (code R)

```
> #estimation utilisant lm
> X=data.frame(x1=X[,1],x2=X[,2])
> lm_disc_fit=lm(y~x1+x2,data=X)
> #estimation utilisant la solution analytique
> XX=as.matrix(cbind(rep(1,n),X))
> a=solve(t(XX)%*%XX)%*%t(XX)%*%y
> a
      [,1]      [,2]      [,3]
rep(1, n)  1.594965 -0.1604043 -0.43456102
x1         -1.672649  1.6329014  0.03974746
x2         -1.009553 -0.7398659  1.74941850
> #prédiction
> y_hat=XX%*%a
> y_hat
      [,1]      [,2]      [,3]
[1,]  1.0346306245 -0.035458347  0.0008277228
[2,]  1.0327193222  0.194725618 -0.22744449398
[3,]  1.1060822067 -0.042891350 -0.0631908571
...

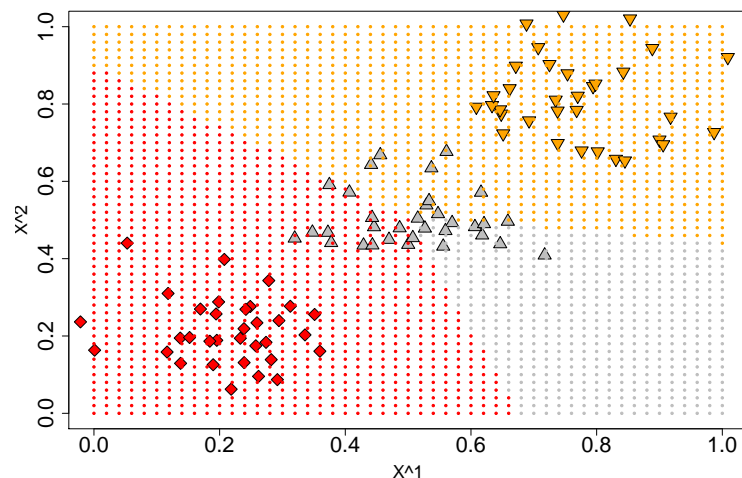
```

Fonction discriminante basée sur les MCO (exemple)



Fonction discriminante basée sur les MCO (suite)

- Cas où la fonction discriminante basée sur les MCO et variables indicatrices n'arrive pas à discriminer correctement une classe :



Fonction discriminante basée sur les MCO (suite)

- L'avantage du critère des MCO est qu'il permet de déterminer une solution analytique.
- Toutefois, cette méthode souffre de plusieurs problèmes :
 - Elle est sensible aux données aberrantes.
 - Quand q est grand et p est petit, il arrive souvent que les frontières de décision linéaires n'arrivent pas à discriminer correctement une ou plusieurs classes. Dans ces cas, utiliser des hyperplans dans \mathbb{X} comme frontière n'est pas suffisant il faut utiliser des expansions de base [Hastie et al., 2011].
- Rappelons que la méthode des MCO est identique à la méthode du MV avec l'hypothèse que la probabilité conditionnelle de Y sachant X est gaussienne. Or ici, les données cibles sont binaires et la loi normale n'est donc pas adaptée à ce type de données. Ceci explique les contre-performances de ces méthodes en pratique.

Analyse discriminante

- Nous voyons maintenant les méthodes d'**analyse discriminante** qui sont issues de la statistique. Elles peuvent être vues comme une extension des modèles linéaires pour la régression au problème de la catégorisation dans le cadre duquel on prédit une variable discrète à partir de variables explicatives continues.
- Nous aborderons d'abord les méthodes **géométriques** : on cherche à décrire dans un espace de dimension réduite les différentes classes de \mathbb{Y} de manière à bien les séparer. Dans cette approche les notions de distances entre points et de variance de nuage de points sont fondamentales.
- Nous donnerons ensuite une interprétation **probabiliste** de l'analyse discriminante : celle-ci permet d'étendre l'approche géométrique à des cas plus génériques.

Analyse discriminante géométrique (suite)

- Nous supposons que $\mathbb{Y} = \{C_1, \dots, C_q\}$ et que nous disposons d'un ensemble d'entraînement \mathbb{E} .
- En considérant tous les points de \mathbb{E} , nous introduisons le centre de gravité $\hat{\boldsymbol{\mu}}$ et la matrice de variance totale $\hat{\boldsymbol{\Sigma}}$:

$$\hat{\boldsymbol{\mu}} = \frac{1}{n} \sum_{\mathbf{x}_i \in \mathbb{E}} \mathbf{x}_i \text{ et } \hat{\boldsymbol{\Sigma}} = \frac{1}{n} \sum_{\mathbf{x}_i \in \mathbb{E}} (\mathbf{x}_i - \hat{\boldsymbol{\mu}})(\mathbf{x}_i - \hat{\boldsymbol{\mu}})^\top$$

- Pour chaque C_l nous pouvons localement définir ces mêmes mesures :

$$\hat{\boldsymbol{\mu}}_l = \frac{1}{|C_l|} \sum_{\mathbf{x}_i \in C_l} \mathbf{x}_i \text{ et } \hat{\boldsymbol{\Sigma}}'_w = \frac{1}{|C_l|} \sum_{\mathbf{x}_i \in C_l} (\mathbf{x}_i - \hat{\boldsymbol{\mu}}_l)(\mathbf{x}_i - \hat{\boldsymbol{\mu}}_l)^\top$$

Remarque : la notation $\hat{\boldsymbol{\Sigma}}'_w$ est justifiée dans ce qui suit.

Analyse discriminante géométrique (suite)

- L'idée de l'analyse discriminante est de déterminer un vecteur \mathbf{a} de \mathbb{X} de norme égale à 1 qui poursuit les deux objectifs suivants :
 - ▶ Lorsque l'on projette les différents centres de gravité $\hat{\boldsymbol{\mu}}_l$ sur l'espace vectoriel engendré par \mathbf{a} , la variance du nuage des vecteurs projetés doit être forte. Ceci permet de mieux séparer les différentes classes sur le vecteur \mathbf{a} .
 - ▶ Lorsque l'on projette les points d'une classe C_l sur \mathbf{a} , la variance du nuage des vecteurs projetés autour de $\hat{\boldsymbol{\mu}}_l$ doit être faible. Ceci permet de garder grouper les points de C_l lorsqu'ils sont projetés sur \mathbf{a} .

Analyse discriminante géométrique (suite)

- Nous définissons la matrice de variance **inter-groupe** $\hat{\boldsymbol{\Sigma}}_b$:

$$\hat{\boldsymbol{\Sigma}}_b = \frac{1}{n} \sum_{l=1}^q |C_l| (\hat{\boldsymbol{\mu}}_l - \hat{\boldsymbol{\mu}})(\hat{\boldsymbol{\mu}}_l - \hat{\boldsymbol{\mu}})^\top$$

- Nous introduisons aussi la matrice de variance **intra-groupe** $\hat{\boldsymbol{\Sigma}}_w$:

$$\hat{\boldsymbol{\Sigma}}_w = \frac{1}{n} \sum_{l=1}^q |C_l| \hat{\boldsymbol{\Sigma}}'_w$$

- Nous avons le théorème suivant :

Théorème. (Théorème de Huygens)

$$\hat{\boldsymbol{\Sigma}} = \hat{\boldsymbol{\Sigma}}_b + \hat{\boldsymbol{\Sigma}}_w$$

Analyse discriminante géométrique (suite)

- On suppose dans la suite que le vecteur \mathbf{a} que l'on cherche à déterminer doit être de norme unitaire.
- Dans ce cas la quantité suivante représente la variance des centres de gravités projetés sur \mathbf{a} :

$$\mathbf{a}^\top \hat{\boldsymbol{\Sigma}}_b \mathbf{a}$$

- Similairement, la quantité suivante représente la variance des vecteurs de C_l projetés sur \mathbf{a} :

$$\mathbf{a}^\top \hat{\boldsymbol{\Sigma}}'_w \mathbf{a}$$

- Comme on cherche à minimiser la variance intra-groupe de chaque classe on s'intéresse à la quantité suivante :

$$\frac{1}{n} \sum_{l=1}^q |C_l| \mathbf{a}^\top \hat{\boldsymbol{\Sigma}}'_w \mathbf{a} = \mathbf{a}^\top \hat{\boldsymbol{\Sigma}}_w \mathbf{a}$$

Analyse discriminante géométrique (suite)

- Le théorème de Huygens, nous permet d'établir que :

$$\hat{\Sigma} = \hat{\Sigma}_b + \hat{\Sigma}_w \Rightarrow \mathbf{a}^\top \hat{\Sigma} \mathbf{a} = \mathbf{a}^\top \hat{\Sigma}_b \mathbf{a} + \mathbf{a}^\top \hat{\Sigma}_w \mathbf{a}$$

- Formellement dans le cadre de l'analyse discriminante on cherche à résoudre le problème d'optimisation suivant :

$$\arg \max_{\mathbf{a} \in \mathbb{X}} r(\mathbf{a}) = \frac{\mathbf{a}^\top \hat{\Sigma}_b \mathbf{a}}{\mathbf{a}^\top \hat{\Sigma} \mathbf{a}}$$

- C'est un problème d'optimisation non contraint. $r(\mathbf{a})$ est également connu sous le nom de **quotient de Rayleigh**.
- Pour déterminer les solutions il faut déterminer les points critiques :

$$\nabla r(\mathbf{a}) = \mathbf{0}$$

Analyse discriminante géométrique (suite)

- Les valeurs propres λ sont comprises entre 0 et 1.
- Les cas particuliers sont les suivants :
 - $\lambda = 1$ signifie que la projection des vecteurs conduit à des variances intra-classes nulles. Les q classes sont alors bien séparées et appartiennent à des sous-espace orthogonaux à \mathbf{a} .
 - $\lambda = 0$ correspond au cas où tous les centre de gravités sont projetés en un même point sur \mathbf{a} . Dans ce cas, les différents nuages de points correspondants à chaque classe s'organisent dans \mathbf{X} sous forme de disques concentriques et il n'est pas possible de les séparer linéairement.
 - $0 < \lambda < 1$ correspond au cas le plus courant. Dans ce cas, il est tout de même possible d'avoir des classes séparées et non recouvrantes.
- L'approche que nous venons de décrire est appelée **analyse factorielle discriminante** (AFD) et sert comme technique de réduction de dimension d'un ensemble de données décrit par des variables numériques mais en tenant compte d'une variable cible discrète.
- Il y a $q - 1$ valeurs propres non nulles. L'espace factoriel engendré par les $q - 1$ vecteurs propres permet de ne pas perdre d'information.

Analyse discriminante géométrique (suite)

- Pour déterminer la solution optimale on développe les conditions de premier ordre suivantes :

$$\begin{aligned} \nabla r(\mathbf{a}) = \mathbf{0} &\Leftrightarrow 2 \frac{\hat{\Sigma}_b \mathbf{a}}{\mathbf{a}^\top \hat{\Sigma} \mathbf{a}} - 2 \frac{\mathbf{a}^\top \hat{\Sigma}_b \mathbf{a} \hat{\Sigma} \mathbf{a}}{(\mathbf{a}^\top \hat{\Sigma} \mathbf{a})^2} = \mathbf{0} \\ &\Leftrightarrow \frac{1}{\mathbf{a}^\top \hat{\Sigma} \mathbf{a}} \left(\hat{\Sigma}_b - \frac{\mathbf{a}^\top \hat{\Sigma}_b \mathbf{a}}{\mathbf{a}^\top \hat{\Sigma} \mathbf{a}} \hat{\Sigma} \right) \mathbf{a} = \mathbf{0} \\ &\Leftrightarrow \left(\hat{\Sigma}_b - r(\mathbf{a}) \hat{\Sigma} \right) \mathbf{a} = \mathbf{0} \\ &\Leftrightarrow \hat{\Sigma}_b \mathbf{a} = r(\mathbf{a}) \hat{\Sigma} \mathbf{a} \end{aligned}$$

- Ainsi la solution du problème précédent est donnée par la solution du problème de **décomposition spectrale généralisée** suivant :

$$\hat{\Sigma}_b \mathbf{a} = \lambda \hat{\Sigma} \mathbf{a}$$

- \mathbf{a}^* est en fait le vecteur propre de la matrice $\hat{\Sigma}^{-1} \hat{\Sigma}_b$ associé à la plus grande valeur propre.

Analyse discriminante géométrique (suite)

- On a la propriété que la maximisation du critère $\frac{\mathbf{a}^\top \hat{\Sigma}_b \mathbf{a}}{\mathbf{a}^\top \hat{\Sigma}_w \mathbf{a}}$ aboutit au même résultat que précédemment :

$$\begin{aligned} \hat{\Sigma}_b \mathbf{a} = \lambda \hat{\Sigma} \mathbf{a} &\Leftrightarrow \hat{\Sigma}_b \mathbf{a} = \lambda (\hat{\Sigma}_b + \hat{\Sigma}_w) \mathbf{a} \\ &\Leftrightarrow \hat{\Sigma}_b \mathbf{a} = \lambda \hat{\Sigma}_b \mathbf{a} + \lambda \hat{\Sigma}_w \mathbf{a} \\ &\Leftrightarrow (1 - \lambda) \hat{\Sigma}_b \mathbf{a} = \lambda \hat{\Sigma}_w \mathbf{a} \\ &\Leftrightarrow \hat{\Sigma}_b \mathbf{a} = \frac{\lambda}{1 - \lambda} \hat{\Sigma}_w \mathbf{a} \end{aligned}$$

- Autre propriété : l'AFD est une ACP des centres de gravité $\hat{\mu}_l$ mais avec une métrique de Mahalanobis c-à-d qui utilise $\hat{\Sigma}_w^{-1}$. L'équivalence précédente indique que la métrique $\hat{\Sigma}^{-1}$ donne aussi le même résultat.
- Au-delà de l'aspect "réduction de dimension", l'AFD permet de faire des prédictions. Pour catégoriser un objet \mathbf{x} , il faut calculer la **distance de Mahalanobis** séparant \mathbf{x} de chaque $\hat{\mu}_l$ et affecter celui-ci à la classe du centre de gravité le plus proche :

$$\hat{f}(\mathbf{x}) = \arg \min_{C_l \in \mathbb{Y}} (\mathbf{x} - \hat{\mu}_l)^\top \hat{\Sigma}_w^{-1} (\mathbf{x} - \hat{\mu}_l)$$

Analyse discriminante géométrique (dist. de Mahalanobis)

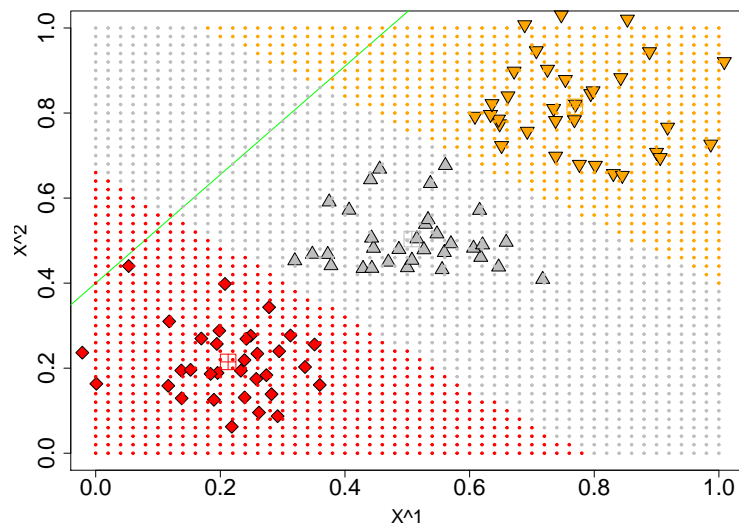
- L'utilisation de la métrique de Mahalanobis est en effet sous-jacente à l'AFD. Elle est due à l'objectif de minimisation de $\mathbf{a}^\top \hat{\Sigma}_w \mathbf{a}$ qui se retrouve au dénominateur du quotient de Rayleigh.
- Cette métrique permet de tenir compte de la dispersion et de l'orientation des nuages de points selon les différentes variables et de normaliser automatiquement l'hétérogénéité de cette dispersion.
- Prenons le cas de variables indépendantes ($\hat{\Sigma}_w$ diagonale) mais de variances non constantes ($\sigma_{x^k}^2 \neq \sigma_{x^l}^2, \forall k \neq l = 1, \dots, p$). Dans ce cas la distance de Mahalanobis s'écrit :

$$\sum_{k=1}^p \frac{(x_{ik} - x_{jk})^2}{\hat{\sigma}_{x^k}^2}$$

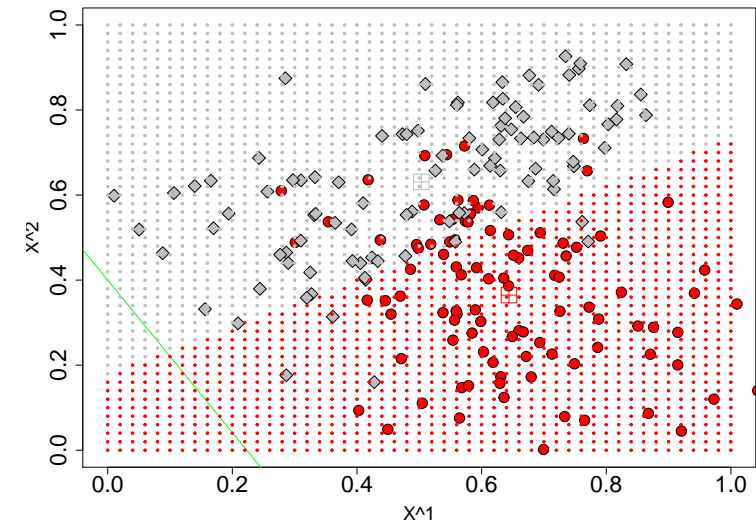
$\hat{\sigma}_{x^k}$ étant l'estimateur sans biais de σ_{x^k} .

Ainsi, si une variable est de forte variance alors elle aura moins de poids dans le calcul de la distance. La métrique de Mahalanobis permet de réduire l'impact de l'hétéroscédasticité.

Analyse discriminante géométrique (code R)



Analyse discriminante géométrique (code R)



Analyse discriminante probabiliste

- L'AFD que nous venons de voir est une approche géométrique. Elle connaît des limites lorsque les classes ne sont pas équilibrées.
- Nous allons interpréter l'AFD dans un cadre probabiliste.
- Soit $g_l(X) = P(X|Y = C_l)$ la probabilité conditionnelle d'observer le vecteur aléatoire X sachant la classe C_l .
- Soit $\pi_l = P(Y = C_l)$ la probabilité **a priori** d'observer la classe C_l .
- Le théorème de Bayes permet de déterminer la probabilité **a posteriori** d'observer la classe C_l sachant X :

$$\begin{aligned} P(C_l|X) &= \frac{P(X|C_l)P(C_l)}{P(X)} \\ &= \frac{g_l(X)\pi_l}{\sum_{k=1}^q g_k(X)\pi_k} \end{aligned}$$

- L'analyse discriminante est donc un **modèle génératif** et la probabilité jointe est telle que : $P(X, C_l) = g_l(X)\pi_l, \forall C_l \in \mathbb{Y}$.

Analyse discriminante probabiliste (suite)

- Nous cherchons donc à déterminer $P(Y = C_l | X)$ pour tout $C_l \in \mathbb{Y}$. Nous nous plaçons dans le cadre d'un **modèle paramétrique**.
- Supposons que chaque classe C_l génère des observations selon une loi normale multidimensionnelle $\mathcal{N}_p(\boldsymbol{\mu}_l, \boldsymbol{\Sigma}_w^l)$, d'espérance le vecteur $\boldsymbol{\mu}_l$ et de matrice variance-covariance $\boldsymbol{\Sigma}_w^l$. On a donc $\forall l = 1, \dots, q$:

$$g_l(\mathbf{x}) = \frac{1}{(2\pi)^{p/2} (\det(\boldsymbol{\Sigma}_w^l))^{1/2}} \exp\left(-\frac{1}{2} \left((\mathbf{x} - \boldsymbol{\mu}_l)^\top [\boldsymbol{\Sigma}_w^l]^{-1} (\mathbf{x} - \boldsymbol{\mu}_l) \right)\right)$$

où $\det(\boldsymbol{\Sigma}_w^l)$ et $[\boldsymbol{\Sigma}_w^l]^{-1}$ sont le déterminant et l'inverse de $\boldsymbol{\Sigma}_w^l$.

- L'approche bayésienne donne alors la règle de classification suivante :

$$f(\mathbf{x}) = \arg \max_{C_k \in \mathbb{Y}} P(C_k | X = \mathbf{x})$$

Analyse discriminante probabiliste (fc. disc. linéaires)

- L'homoscédasticité aboutit à une règle d'affection linéaire :

$$h_k(\mathbf{x}) = -\frac{1}{2} \boldsymbol{\mu}_k^\top \boldsymbol{\Sigma}_w^{-1} \boldsymbol{\mu}_k + \mathbf{x}^\top \boldsymbol{\Sigma}_w^{-1} \boldsymbol{\mu}_k + \log \pi_k$$

- En supposant l'**équiprobabilité des classes** ($\forall k, \pi_k = 1/q$) on a :

$$\begin{aligned} \arg \max_{C_k \in \mathbb{Y}} h_k(\mathbf{x}) &= \arg \max_{C_k \in \mathbb{Y}} \mathbf{x}^\top \boldsymbol{\Sigma}_w^{-1} \boldsymbol{\mu}_k - \frac{1}{2} \boldsymbol{\mu}_k^\top \boldsymbol{\Sigma}_w^{-1} \boldsymbol{\mu}_k \\ &= \arg \max_{C_k \in \mathbb{Y}} 2\mathbf{x}^\top \boldsymbol{\Sigma}_w^{-1} \boldsymbol{\mu}_k - \boldsymbol{\mu}_k^\top \boldsymbol{\Sigma}_w^{-1} \boldsymbol{\mu}_k - \mathbf{x}^\top \boldsymbol{\Sigma}_w^{-1} \mathbf{x} \\ &= \arg \min_{C_k \in \mathbb{Y}} (\mathbf{x} - \boldsymbol{\mu}_k)^\top \boldsymbol{\Sigma}_w^{-1} (\mathbf{x} - \boldsymbol{\mu}_k) \end{aligned}$$

- Si on estime $\boldsymbol{\Sigma}_w$ par l'estimateur (**non biaisée**) on a :

$$\hat{\boldsymbol{\Sigma}}_w = \frac{1}{n - q} \sum_{l=1}^q \sum_{\mathbf{x}_i \in C_l} (\mathbf{x}_i - \hat{\boldsymbol{\mu}}_l)(\mathbf{x}_i - \hat{\boldsymbol{\mu}}_l)^\top$$

On retrouve la règle basée sur la distance de Mahalanobis.

Analyse discriminante probabiliste (suite)

- Sous les hypothèses de gaussianité, nous avons :

$$\begin{aligned} \max_{C_k \in \mathbb{Y}} P(C_k | X = \mathbf{x}) &\Leftrightarrow \max \log(P(C_k | X = \mathbf{x})) \\ &\Leftrightarrow \max \log(g_k(\mathbf{x}) \pi_k) \\ &\Leftrightarrow \max \left[-\frac{1}{2} \left((\mathbf{x} - \boldsymbol{\mu}_k)^\top [\boldsymbol{\Sigma}_w^k]^{-1} (\mathbf{x} - \boldsymbol{\mu}_k) \right) \right. \\ &\quad \left. + \log \pi_k - \frac{p}{2} \log(2\pi) - \log \det(\boldsymbol{\Sigma}_w^k)^{1/2} \right] \\ &\Leftrightarrow \max \left[-\frac{1}{2} \left(\mathbf{x}^\top [\boldsymbol{\Sigma}_w^k]^{-1} \mathbf{x} + \boldsymbol{\mu}_k^\top [\boldsymbol{\Sigma}_w^k]^{-1} \boldsymbol{\mu}_k - 2\mathbf{x}^\top [\boldsymbol{\Sigma}_w^k]^{-1} \boldsymbol{\mu}_k \right) \right. \\ &\quad \left. + \log \pi_k - \frac{p}{2} \log(2\pi) - \frac{1}{2} \log \det(\boldsymbol{\Sigma}_w^k) \right] \end{aligned}$$

- Il s'agit donc d'une **fonction quadratique** en \mathbf{x} .
- Si on suppose l'**homoscédasticité**, $\forall k, \boldsymbol{\Sigma}_w^k = \boldsymbol{\Sigma}_w$, on obtient une fonction linéaire dite **fonction linéaire discriminante** :

$$h_k(\mathbf{x}) = -\frac{1}{2} \boldsymbol{\mu}_k^\top \boldsymbol{\Sigma}_w^{-1} \boldsymbol{\mu}_k + \mathbf{x}^\top \boldsymbol{\Sigma}_w^{-1} \boldsymbol{\mu}_k + \log \pi_k$$

- On a $\arg \max_{C_k \in \mathbb{Y}} P(C_k | X = \mathbf{x}) = \arg \max_{C_k \in \mathbb{Y}} h_k(\mathbf{x})$.
On définit alors $\hat{f}(\mathbf{x}) = \arg \max_{C_k \in \mathbb{Y}} \hat{h}_k(\mathbf{x})$.

Analyse discriminante probabiliste (modèle général)

- En résumé, nous pouvons avoir 3 hypothèses :

- ▶ Hypothèse de gaussianité : $\forall k, P(X | C_k) \sim \mathcal{N}_p(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_w^k)$.
- ▶ Hypothèse d'homoscédasticité : $\forall k, \boldsymbol{\Sigma}_w^k = \boldsymbol{\Sigma}_w$.
- ▶ Hypothèse d'équiprobabilité : $\forall k, P(C_k) = \pi_k = 1/q$.

- Si on suppose toutes les hypothèses, on obtient la règle de décision géométrique de l'AFD classique.
- Si on suppose toutes les hypothèses sauf l'équiprobabilité, on obtient la règle de décision dite probabiliste de l'analyse discriminante linéaire. Dans ce cas, la méthode traite mieux les cas non-équiprobables. La fonction de discrimination ainsi que les frontières en découlant sont linéaires dans l'espace \mathbb{X} .
- Si on ne suppose que le modèle paramétrique gaussien, on obtient une méthode dite **analyse discriminante quadratique**. Dans ce cas, le plus général des 3, on obtient des fonctions discriminantes quadratiques et les frontières dans \mathbb{X} sont courbées.

Analyse discriminante probabiliste (fc. disc. quadratique)

- Dans le cas de l'analyse discriminante quadratique, on estime les paramètres du modèle comme suit, $\forall k = 1, \dots, q$:

$$\begin{aligned}\hat{\pi}_k &= \frac{|C_k|}{n} \\ \hat{\boldsymbol{\mu}}_k &= \frac{1}{|C_k|} \sum_{\mathbf{x}_i \in C_k} \mathbf{x}_i \\ \hat{\boldsymbol{\Sigma}}_w^k &= \frac{1}{|C_k|} \sum_{\mathbf{x}_i \in C_k} (\mathbf{x}_i - \hat{\boldsymbol{\mu}}_k)(\mathbf{x}_i - \hat{\boldsymbol{\mu}}_k)^\top\end{aligned}$$

- Les **fonctions discriminantes quadratiques** sont, $\forall k = 1, \dots, q$:

$$h_k(\mathbf{x}) = -\frac{1}{2} \left((\mathbf{x} - \boldsymbol{\mu}_k)^\top [\boldsymbol{\Sigma}_w^k]^{-1} (\mathbf{x} - \boldsymbol{\mu}_k) \right) + \log \pi_k - \frac{1}{2} \log \det(\boldsymbol{\Sigma}_w^k)$$

- La règle de décision est donc : $\hat{f}(\mathbf{x}) = \arg \max_{C_k \in \mathbb{Y}} \hat{h}_k(\mathbf{x})$

Analyse discriminante régularisée (suite)

- L'approche est globalement similaire à la pénalisation ridge vue précédemment dans le cadre de la régression linéaire.
- Similairement au slide 172, on a un hyperparamètre $\gamma \in [0, 1]$ qui vise à régulariser les matrices de variance-covariance intra-groupe, $\forall k = 1, \dots, q$:

$$\hat{\boldsymbol{\Sigma}}_w^k \leftarrow (1 - \gamma) \hat{\boldsymbol{\Sigma}}_w^k + \gamma \mathbf{I}_p$$

- Comme décrit précédemment et similairement au contenu du slide 145, en pratique, on peut utiliser la décomposition spectrale des $\hat{\boldsymbol{\Sigma}}_w^k$. Dans ce cas, pour déterminer \hat{h}_k , on peut utiliser :

$$\begin{aligned}(\mathbf{x} - \boldsymbol{\mu}_k)^\top [\boldsymbol{\Sigma}_w^k]^{-1} (\mathbf{x} - \boldsymbol{\mu}_k) &= ([\mathbf{U}^k]^\top (\mathbf{x} - \boldsymbol{\mu}_k))^\top [\boldsymbol{\Lambda}^k]^{-1} ([\mathbf{U}^k]^\top (\mathbf{x} - \boldsymbol{\mu}_k)) \\ \log \det(\boldsymbol{\Sigma}_w^k) &= \sum_{j=1}^p \log(\lambda_j^k)\end{aligned}$$

où $[\mathbf{U}^k]$ est la matrice des vecteurs propres de $\hat{\boldsymbol{\Sigma}}_w^k$ et $\boldsymbol{\Lambda}^k$ est la matrice diagonale des valeurs propres de $\hat{\boldsymbol{\Sigma}}_w^k$.

Analyse discriminante régularisée

- Dans le cas de données de grande dimension $n \ll p$, où dans le cas de petits échantillons d'entraînement, $n < p$, on a pour chaque classe C_k , $|C_k| \ll p$ et alors l'estimateur $\hat{\boldsymbol{\Sigma}}_w^k$ est instable car la matrice n'est pas de plein rang et est donc singulière.
- Si $|C_k| \ll p$ les petites valeurs propres de $\hat{\boldsymbol{\Sigma}}_w^k$ sont biaisées (elles sont trop petites par rapport à la valeur théorique) et ceci a un impact sur son inverse (qui intervient dans \hat{h}_k). Supposons que $\{\lambda_j^k\}_{j=1, \dots, p}$ et $\{\mathbf{u}_j^k\}_{j=1, \dots, p}$ sont les valeurs et vecteurs propres de $\hat{\boldsymbol{\Sigma}}_w^k$ alors nous avons :

$$[\hat{\boldsymbol{\Sigma}}_w^k]^{-1} = \sum_{j=1}^p \frac{\mathbf{u}_j^k [\mathbf{u}_j^k]^\top}{\lambda_j^k}$$

- Ainsi les axes principaux associés aux valeurs propres les plus petites jouent un rôle sur-estimé dans les fonctions discriminantes.
- Dans [Friedman, 1989], on propose de régulariser la matrice de variance-covariance afin de tenir compte de ce problème.

Régression logistique polytomique

- La régression logistique fait partie des modèles de type **discriminatif** : on cherche à modéliser directement la probabilité conditionnelle de chaque classe C_j étant donné le vecteur aléatoire X .
- Cette probabilité conditionnelle est une fonction linéaire en X :

$$\begin{aligned}\log \frac{P(Y = C_1 | X = \mathbf{x})}{P(Y = C_q | X = \mathbf{x})} &= a_{10} + \mathbf{a}_1^\top \mathbf{x} \\ \log \frac{P(Y = C_2 | X = \mathbf{x})}{P(Y = C_q | X = \mathbf{x})} &= a_{20} + \mathbf{a}_2^\top \mathbf{x} \\ &\vdots \\ \log \frac{P(Y = C_{q-1} | X = \mathbf{x})}{P(Y = C_q | X = \mathbf{x})} &= a_{q-10} + \mathbf{a}_{q-1}^\top \mathbf{x}\end{aligned}$$

- On spécifie ainsi le modèle en prenant $q - 1$ **fonctions logit** comparant chaque classe C_1, \dots, C_{q-1} à la classe de référence C_q : $\text{logit}(p) = \log\left(\frac{p}{1-p}\right)$ avec $p \in]0, 1[$.

Régression logistique polytomique (suite)

- De plus, on suppose que **chaque fonction logit peut être représentée par un modèle linéaire**.
- Le modèle spécifié précédemment conduit aux propriétés suivantes $\forall k = 1, \dots, q - 1$:

$$P(Y = C_k | X = \mathbf{x}) = \frac{\exp(a_{k0} + \mathbf{a}_k^\top \mathbf{x})}{1 + \sum_{l=1}^{q-1} \exp(a_{l0} + \mathbf{a}_l^\top \mathbf{x})}$$

$$P(Y = C_q | X = \mathbf{x}) = \frac{1}{1 + \sum_{l=1}^{q-1} \exp(a_{l0} + \mathbf{a}_l^\top \mathbf{x})}$$

- Remarque : le choix de la classe C_q au dénominateur est arbitraire mais fait jouer à cette dernière un rôle particulier.
- Nous pouvons clairement vérifier que $\sum_{l=1}^q P(Y = C_l | X = \mathbf{x}) = 1$.
- Ici, l'ensemble des paramètres est $\mathbb{P} = \{(a_{l0}, \mathbf{a}_l)\}_{l=1}^{q-1}$.

Régression logistique polytomique (suite)

- Le modèle n'est pas encore totalement spécifié, il nous faut choisir une famille de loi de probabilité pour $P(Y|X)$. Dans le cadre de cette méthode, on choisit la **distribution multinomiale** où étant donné X chaque classe C_k a une probabilité $P(C_k|X)$ d'être observée.
- Une fois le modèle paramétrique établi, on détermine les paramètres par la méthode du **maximum de vraisemblance** :

$$\max_{\mathbb{P}} \text{vr}(\mathbb{P}) = P(Y_1, \dots, Y_n | X_1, \dots, X_n; \mathbb{P})$$

Régression logistique polytomique (suite)

- Dans les équations précédentes, la classe C_q , prise comme référence, est traitée de manière particulière. Afin de rendre uniforme le traitement des classes nous poserons de façon équivalente :

$$\forall k = 1, \dots, q : P(Y = C_k | X = \mathbf{x}) = \frac{\exp(a_{k0} + \mathbf{a}_k^\top \mathbf{x})}{\sum_{l=1}^q \exp(a_{l0} + \mathbf{a}_l^\top \mathbf{x})}$$

- La fonction $\frac{\exp(a_{k0} + \mathbf{a}_k^\top \mathbf{x})}{\sum_{l=1}^q \exp(a_{l0} + \mathbf{a}_l^\top \mathbf{x})}$ est appelée fonction **softmax** et nous voyons que dans ce cas également $\sum_{l=1}^q P(Y = C_l | X = \mathbf{x}) = 1$.
- L'appellation softmax vient du fait que s'il existe une classe C_k telle que $a_{k0} + \mathbf{a}_k^\top \mathbf{x}$ est largement supérieure aux autres classes $C_l \neq C_k$ alors, la fonction retourne une valeur proche de 1. Ainsi la fonction agit comme la "fonction max" excepté qu'elle est **différentiable**.

Régression logistique polytomique (suite)

- Dans le cas général multiclassés, nous représentons l'appartenance des individus aux différentes classes par une matrice binaire \mathbf{Y} de taille $(n \times q)$ et de terme général :

$$y_{il} = \begin{cases} 1 & \text{si } \mathbf{x}_i \in C_l \\ 0 & \text{sinon} \end{cases}$$

- On modélise la probabilité par une **distribution multinomiale**. Sous l'hypothèse i.i.d., la vraisemblance s'écrit alors comme suit :

$$\text{vr}(\mathbb{P}) = \prod_{l=1}^q \prod_{i=1}^n P(Y = C_l | \mathbf{x}_i; \mathbf{a}_l)^{y_{il}}$$

- La log-vraisemblance vaut alors :

$$\text{lvr}(\mathbb{P}) = \sum_{l=1}^q \sum_{i=1}^n y_{il} \log(P(Y = C_l | \mathbf{x}_i; \mathbf{a}_l))$$

Régression logistique polytomique (suite)

- En remplaçant $P(Y = C_l | \mathbf{x}_i; \mathbf{a}_l)$ par la forme paramétrique introduite précédemment, on a :

$$lvr(\mathbb{P}) = \sum_{l=1}^q \sum_{i=1}^n y_{il} \log \left(\frac{\exp(a_{l0} + \mathbf{a}_l^\top \mathbf{x}_i)}{\sum_{k=1}^q \exp(a_{k0} + \mathbf{a}_k^\top \mathbf{x}_i)} \right)$$

- Le problème n'ayant **pas de solution analytique**, on a recourt à des outils d'**optimisation numérique**. On utilise ici l'**algorithme de Newton-Raphson** (ou la méthode IRLS "Iteratively Reweighted Least Squares") pour déterminer une solution approchée de l'estimateur du MV. Pour cela, il faut déterminer le gradient de la lvr par rapport à \mathbf{a}_l ainsi que la matrice hessienne.

Régression logistique polytomique (code R)

```
> install.packages("nnet")
> library("nnet")
> mult_log_reg_fit=multinom(formula=c~.,data=as.data.frame(X))
> summary(mult_log_reg_fit)
Call:
multinom(formula = c ~ ., data = as.data.frame(X))
```

```
Coefficients:
(Intercept)      V1      V2
2  -41.43095  50.04171  65.94771
3  -123.68747 116.61844 125.96799
```

```
Std. Errors:
(Intercept)      V1      V2
2   45.57533 106.8095 107.3117
3  105.34071 178.3756 163.4239
```

```
Residual Deviance: 0.1266945
AIC: 12.12669
```

Pseudo-code de l'algorithme de Newton-Raphson

Input : $f \in \mathcal{C}^2, \mathbf{x}^0$

- $k \leftarrow 0$
- Tant que** condition d'arrêt non satisfaite **faire**
- $\mathbf{x}^{(k+1)} \leftarrow \mathbf{x}^{(k)} - \nabla^2 f(\mathbf{x}^{(k)})^{-1} \nabla f(\mathbf{x}^{(k)})$
- $k \leftarrow k + 1$
- Fin Tant que**
- Output :** $\mathbf{x}^{(k)}$

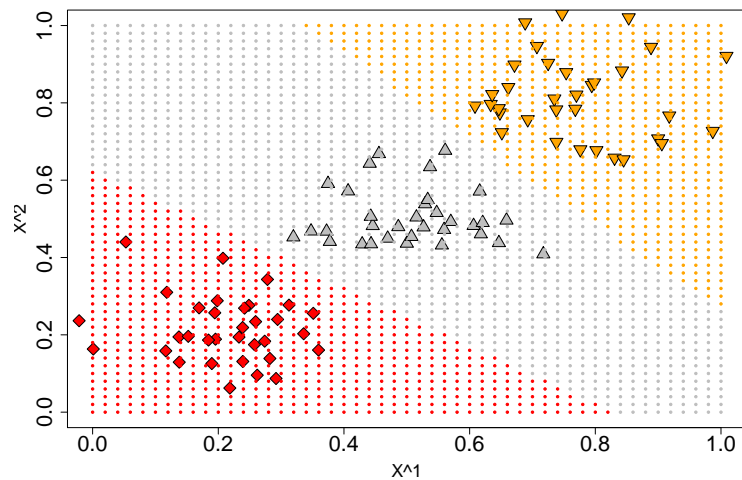
où la matrice $\nabla^2 f(\mathbf{x})$ est appelée **matrice hessienne** de f en \mathbf{x} avec :

$$\nabla^2 f = \begin{pmatrix} \frac{\partial^2 f}{\partial x_1 \partial x_1} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \cdots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n \partial x_n} \end{pmatrix}$$

Régression logistique (code R)

```
> predict(object=mult_log_reg_fit,newdata=as.data.frame(X),"probs")
          1          2          3
1  1.000000e+00  3.680581e-10  7.706627e-37
2  1.000000e+00  3.440253e-12  5.631024e-40
3  1.000000e+00  1.193568e-11  7.103875e-40
4  9.999974e-01  2.625437e-06  1.890287e-27
5  1.000000e+00  4.243419e-08  4.993025e-32
6  9.999997e-01  2.811394e-07  1.646368e-31
7  9.999788e-01  2.121208e-05  9.949065e-27
...
90 1.708672e-23  9.320129e-05  9.999068e-01
91 2.724571e-31  1.827336e-09  1.000000e+00
92 3.540231e-27  9.936262e-07  9.999990e-01
93 1.762006e-26  4.946121e-06  9.999951e-01
94 9.321240e-37  1.745066e-12  1.000000e+00
95 5.468878e-38  2.216307e-12  1.000000e+00
96 1.783892e-23  3.787065e-05  9.999621e-01
```

Régression logistique (code R)



Régression logistique pénalisée (code R)

- Le package `glmnet` implémente la régression logistique pénalisée ainsi que d'autres modèles linéaires généralisés pénalisés. [Friedman et al., 2010].

```
library(glmnet)
data("iris")
#lasso
X=as.matrix(iris[,1:4])
y=as.numeric(iris[,5])
lasso_fit=glmnet(x=X,y=y,family = "multinomial", alpha=1)
plot(lasso_fit)
cv_lasso_fit=cv.glmnet(x=X,y=y,family = "multinomial", alpha=1, type.measure = "class")
plot(cv_lasso_fit)
cv_lasso_fit$lambda.min
coef(cv_lasso_fit,s = "lambda.min")

> cv_lasso_fit$lambda.min
[1] 0.00149224
```

Régression logistique pénalisée

- Le **principe de régularisation** pour obtenir un modèle de faible variance et parcimonieux a été également appliqué à d'autres fonctions objectif que les MCO telle que la log-vraisemblance.
- Dans le cas de la régression logistique polytomique utilisant les fonctions softmax, notons l'ensemble des paramètres $\mathbb{P} = \{(a_0, \mathbf{a}_l) \in \mathbb{R}^{p+1}\}_{l=1}^q$ nous obtenons le modèle pénalisé suivant :

$$\max_{\{(a_0, \mathbf{a}_l)\}_{l=1}^q \in \mathbb{R}^{(p+1)q}} \text{lv}r(\mathbb{P}) - \lambda \sum_{l=1}^q ((1 - \alpha) \|\mathbf{a}_l\|_{\ell_1} + \alpha \|\mathbf{a}_l\|_{\ell_2}^2)$$

Régression logistique pénalisée (code R)

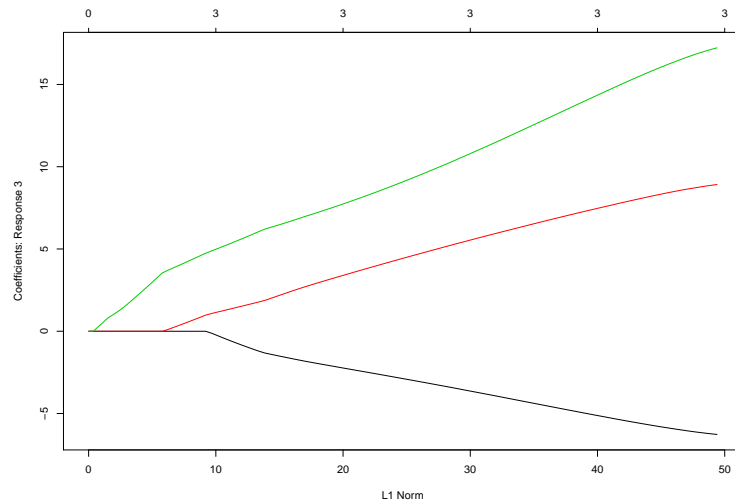
```
> coef(cv_lasso_fit,s = "lambda.min")
$'1'
5 x 1 sparse Matrix of class "dgCMatrix"
      1
(Intercept) 11.406474
Sepal.Length .
Sepal.Width 3.427404
Petal.Length -2.771932
Petal.Width -1.562535

$'2'
5 x 1 sparse Matrix of class "dgCMatrix"
      1
(Intercept) 5.779377
Sepal.Length 1.368141
Sepal.Width .
Petal.Length .
Petal.Width .

$'3'
5 x 1 sparse Matrix of class "dgCMatrix"
      1
(Intercept) -17.185851
Sepal.Length 2.923410
Sepal.Width 3.427404
Petal.Length -2.771932
Petal.Width -1.562535
```


Régression logistique pénalisée (code R)

```
plot(lasso_fit)
```

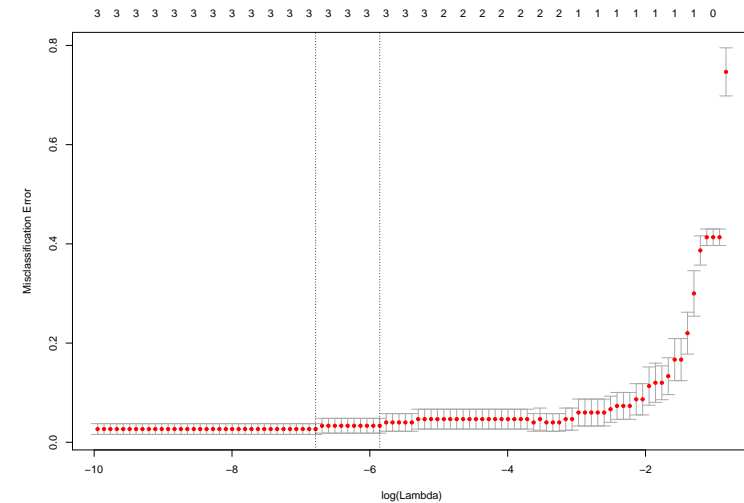


Rappel du Sommaire

- 1 Introduction
- 2 Les méthodes linéaires et leurs pénalisations (ridge, lasso, ...)
- 3 Les machines à vecteurs supports ("Support Vector Machines")
- 4 Les arbres de décisions ("Decision Trees")
- 5 Décider en comité ("Ensemble Learning")

Régression logistique pénalisée (code R)

```
plot(cv_lasso_fit)
```

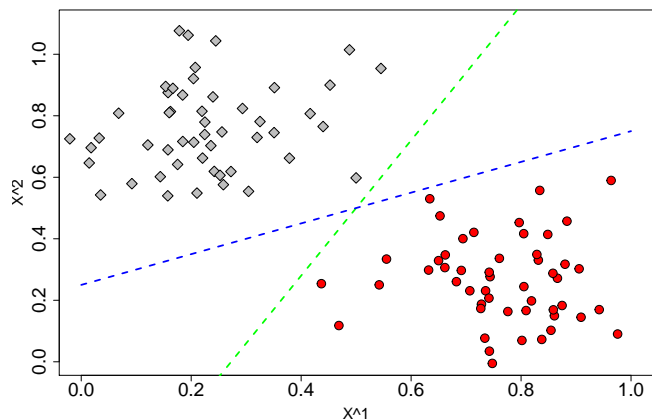


Introduction

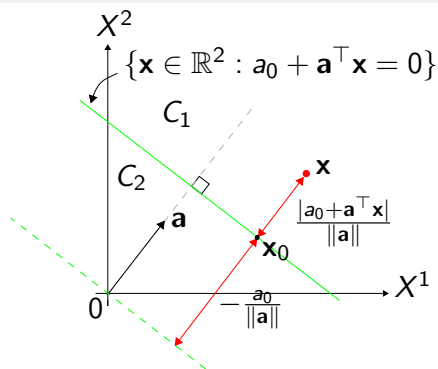
- C'est une famille de méthodes "récentes" développées initialement par Vapnik [Vapnik, 1995] dans les années 90.
- Nous étudierons dans un premier temps l'application de cette méthode pour le problème de catégorisation puis nous verrons comment elle permet également de traiter les problèmes de régression.
- C'est une **méthode discriminante** qui estime la frontière de décision entre deux catégories.
- Cette **frontière peut-être définie par des objets de \mathbb{E}** et non nécessairement par les variables \mathbb{A} .
- La méthode repose sur la matrice de Gram c-à-d la matrice des produits scalaires entre objets de \mathbb{E} (et non nécessairement sur la représentation vectorielle).
- La méthode cherche à résoudre un problème d'**optimisation convexe** et il existe donc une **solution unique**.

Hyperplans de séparation entre deux classes

- On suppose un problème avec deux catégories C_1 et C_2 .
- Il existe une infinité d'hyperplans permettant de séparer deux nuages de points linéairement séparable.



Optimisation de la marge



- Dans \mathbb{R}^p , le vecteur normal (càd orthogonal) de la frontière est \mathbf{a} .
- La distance (signée) entre la frontière et l'origine est $-a_0/\|\mathbf{a}\|$.
- Soit \mathbf{x}_0 un point de la frontière, la distance entre \mathbf{x} et la frontière est :

$$\frac{|\mathbf{a}^\top (\mathbf{x} - \mathbf{x}_0)|}{\|\mathbf{a}\|} = \frac{|\mathbf{a}^\top \mathbf{x} + a_0|}{\|\mathbf{a}\|}$$

Hyperplans de séparation optimale entre deux classes

- Dans le cas des svm, on cherche la frontière linéaire représentée par $a_0 \in \mathbb{R}$ et $\mathbf{a} \in \mathbb{R}^p$ telle que :

$$\begin{cases} a_0 + \mathbf{a}^\top \mathbf{x} \geq \delta & \text{pour tout } \mathbf{x} \in C_1 \\ a_0 + \mathbf{a}^\top \mathbf{x} \leq -\delta & \text{pour tout } \mathbf{x} \in C_2 \end{cases}$$

avec $\delta \geq 0$.

- Contrairement aux fonctions discriminantes où on regardait uniquement le signe par rapport à la frontière ($g(\mathbf{x}) \leq 0$), on veut aussi une distance δ par rapport à la frontière.
- On appelle la **marge**, la distance entre la frontière et les objets \mathbf{x} les plus proches de celle-ci.
- L'apprentissage consiste alors à déterminer l'hyperplan permettant de **maximiser la marge** (on traduit parfois svm par "Séparateur à Vaste Marge") afin d'obtenir une meilleure généralisation.

Optimisation de la marge (suite)

- On a alors le problème suivant :

$$\begin{aligned} & \max_{a_0, \mathbf{a} \in \mathbb{R}^p} \delta \\ \text{s.t.} & \forall i, y_i \frac{(\mathbf{a}^\top \mathbf{x}_i + a_0)}{\|\mathbf{a}\|} \geq \delta \end{aligned}$$

où $y_i = 1$ si $\mathbf{x}_i \in C_1$ et $y_i = -1$ si $\mathbf{x}_i \in C_2$.

- Dans les contraintes, on peut écrire de façon équivalente :

$$y_i (\mathbf{a}^\top \mathbf{x}_i + a_0) \geq \delta \|\mathbf{a}\|$$

- Puis sans perte de généralité on peut poser :

$$\delta = \frac{1}{\|\mathbf{a}\|}$$

- Le problème devient alors :

$$\begin{aligned} & \min_{a_0, \mathbf{a} \in \mathbb{R}^p} \frac{1}{2} \|\mathbf{a}\|^2 \\ \text{s.t.} & \forall i, y_i (\mathbf{a}^\top \mathbf{x}_i + a_0) \geq 1 \end{aligned}$$

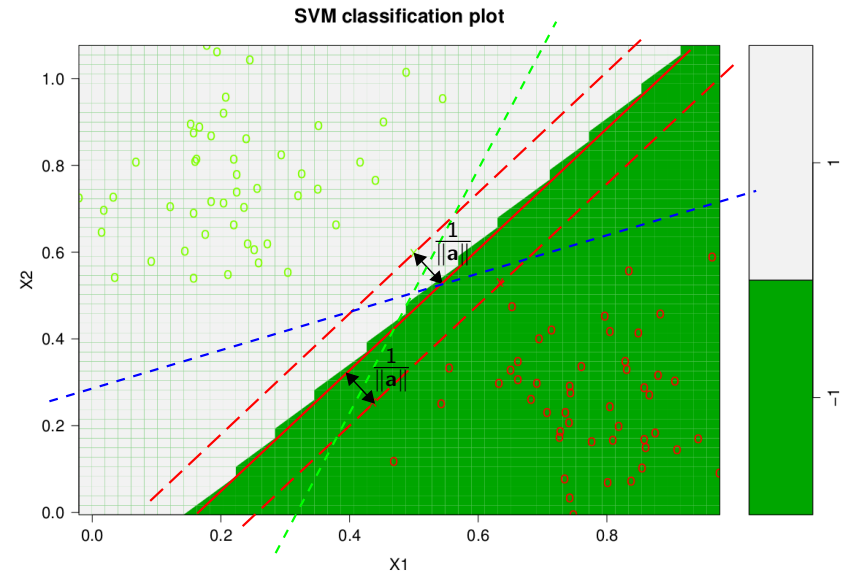
Exemple



Problème quadratique contraint

- La marge $\delta = 1/\|\mathbf{a}\|$ donc $2/\|\mathbf{a}\|$ est l'épaisseur de la bande (ou tube).
- Il n'y a uniquement que quelques points (ceux marqués d'une croix dans l'exemple précédent) qui participent à la définition de la frontière (cf plus loin).
- Pour **maximiser la marge** cela revient donc à **minimiser la norme euclidienne au carré du vecteur normal \mathbf{a} de la frontière**. Il s'agit d'un **problème quadratique avec des contraintes d'inégalités linéaires** (de type \geq). La fonction objectif ainsi que les contraintes sont des fonctions convexes. Il s'agit donc d'un **problème convexe** que l'on peut résoudre en utilisant des solvers où en appliquant des méthodes d'optimisation numériques dédiées à ce problème.
- Toutefois, on peut reformuler de façon équivalente ce problème en écrivant le **Lagrangien associé** et en formant ainsi le **dual (de Wolfe)**. Les propriétés de dualité des problèmes convexes aboutissent, dans le cas des SVM, à des caractéristiques particulièrement intéressantes.

Exemple (suite)



Lagrangien et problème dual

- Reprenons le problème primal suivant :

$$\begin{aligned} \min_{a_0, \mathbf{a} \in \mathbb{R}^p} \quad & \frac{1}{2} \|\mathbf{a}\|^2 \\ \text{s.t.} \quad & \forall i, y_i (\mathbf{a}^\top \mathbf{x}_i + a_0) \geq 1 \end{aligned}$$

- Le **Lagrangien (primal)** est noté $lag_p(a_0, \mathbf{a}, \boldsymbol{\alpha})$ où $\boldsymbol{\alpha}$ est le vecteur de taille $(n \times 1)$ des **multiplicateurs de Lagrange**. Il est défini par :

$$lag_p(a_0, \mathbf{a}, \boldsymbol{\alpha}) = \frac{1}{2} \|\mathbf{a}\|^2 - \sum_{i=1}^n \alpha_i (y_i (\mathbf{a}^\top \mathbf{x}_i + a_0) - 1)$$

- Le Lagrangien doit être minimisé selon a_0 et \mathbf{a} et maximiser par rapport à $\boldsymbol{\alpha} = (\alpha_1, \dots, \alpha_n)$ (propriété de dualité faible).
- Par conséquent les CNPO impliquent que la solution se trouve en un point selle.

Lagrangien et problème dual (suite)

- Le problème étant convexe, il est équivalent de résoudre le dual qui consiste à maximiser le Lagrangien lag_d par rapport à α sous les contraintes que les gradients de lag_p par rapport à a_0 et \mathbf{a} soient nuls :

$$\begin{cases} \frac{\partial lag_p}{\partial a_0} = 0 \\ \frac{\partial lag_p}{\partial \mathbf{a}} = \mathbf{0} \end{cases} \Leftrightarrow \begin{cases} \sum_{i=1}^n \alpha_i y_i = 0 \\ \mathbf{a} - \sum_{i=1}^n \alpha_i y_i \mathbf{x}_i = \mathbf{0} \end{cases}$$

- On obtient les relations suivantes $\sum_{i=1}^n \alpha_i y_i = 0$ et $\mathbf{a} = \sum_{i=1}^n \alpha_i y_i \mathbf{x}_i$
- En intégrant ces relations au sein du Lagrangien lag_p on obtient :

$$\begin{aligned} lag_p(a_0, \mathbf{a}, \alpha) &= \frac{1}{2} \|\mathbf{a}\|^2 - \sum_{i=1}^n \alpha_i (y_i (\mathbf{a}^\top \mathbf{x}_i + a_0) - 1) \\ &= \frac{1}{2} \mathbf{a}^\top \mathbf{a} - \sum_{i=1}^n \alpha_i y_i (\mathbf{a}^\top \mathbf{x}_i + a_0) + \sum_{i=1}^n \alpha_i \\ &= \frac{1}{2} \mathbf{a}^\top \mathbf{a} - \sum_{i=1}^n \alpha_i y_i \mathbf{a}^\top \mathbf{x}_i - a_0 \sum_{i=1}^n \alpha_i y_i + \sum_{i=1}^n \alpha_i \end{aligned}$$

Lagrangien et problème dual (suite)

- En intégrant ces relations au sein du Lagrangien on obtient (suite) :

$$\begin{aligned} lag_p(a_0, \mathbf{a}, \alpha) &= \frac{1}{2} \mathbf{a}^\top \mathbf{a} - \mathbf{a}^\top \sum_{i=1}^n \alpha_i y_i \mathbf{x}_i + \sum_{i=1}^n \alpha_i \\ &= \frac{1}{2} \mathbf{a}^\top \mathbf{a} - \mathbf{a}^\top \mathbf{a} + \sum_{i=1}^n \alpha_i \\ &= \sum_{i=1}^n \alpha_i - \frac{1}{2} \mathbf{a}^\top \mathbf{a} \\ &= \sum_{i=1}^n \alpha_i - \frac{1}{2} \left(\sum_{i=1}^n \alpha_i y_i \mathbf{x}_i \right)^\top \sum_{j=1}^n \alpha_j y_j \mathbf{x}_j \\ &= \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j y_i y_j \mathbf{x}_i^\top \mathbf{x}_j \\ &= lag_d(\alpha) \end{aligned}$$

Lagrangien et problème dual (suite)

- Le **problème dual** est alors le suivant :

$$\begin{aligned} \max_{\alpha \in \mathbb{R}^n} \quad & \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j y_i y_j \mathbf{x}_i^\top \mathbf{x}_j \\ \text{s.c.} \quad & \forall i, \alpha_i \geq 0 \\ & \sum_{i=1}^n \alpha_i y_i = 0 \end{aligned}$$

- En plus de la contrainte sur les multiplicateurs de Lagrange, la solution optimale du dual doit également satisfaire les autres **conditions de Karush-Kuhn-Tucker (KKT)** suivantes (dites conditions complémentaires) :

$$\forall i, \alpha_i (y_i (\mathbf{a}^\top \mathbf{x}_i + a_0) - 1) = 0$$

- Ces conditions complémentaires s'interprètent de la façon suivante :
 - Si $\alpha_i > 0$ alors la contrainte est saturée (on dit aussi active) càd $y_i (\mathbf{a}^\top \mathbf{x}_i + a_0) = 1$ et \mathbf{x}_i se situe sur une frontière de la bande.
 - Si $y_i (\mathbf{a}^\top \mathbf{x}_i + a_0) > 1$ alors $\alpha_i = 0$ et dans ce cas \mathbf{x}_i se situe hors de la bande.

Interprétation du svm

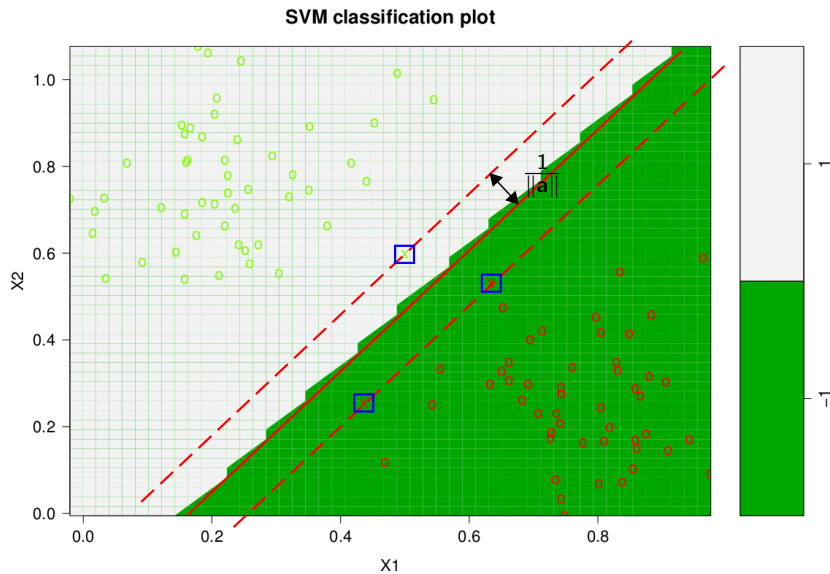
- Rappelons que nous avons $\hat{\mathbf{a}}_{svm} = \sum_{i=1}^n \hat{\alpha}_i y_i \mathbf{x}_i$.
- De plus, seuls les \mathbf{x}_i sur les frontières de la bande sont tels que $\hat{\alpha}_i > 0$. On les appelle les **vecteurs supports**.
- En d'autres termes, $\hat{\mathbf{a}}_{svm}$ est défini comme une combinaison linéaire des vecteurs supports.
- Les objets \mathbf{x}_i tel que $\hat{\alpha}_i = 0$ sont des points hors de la bande et ne sont pas intéressants pour définir la frontière entre les deux classes (ils sont relativement loin de la frontière).
- On obtient $\hat{a}_{svm,0}$ à l'aide de l'équation suivante pour n'importe quel vecteur support (càd tel que $\alpha_i > 0$) :

$$\hat{a}_{svm,0} = y_i - \hat{\mathbf{a}}_{svm}^\top \mathbf{x}_i$$

- La fonction de décision $\hat{f}(\mathbf{x})$ dépend de $\hat{g}(\mathbf{x}) = \hat{\mathbf{a}}_{svm}^\top \mathbf{x} + \hat{a}_{svm,0}$:

$$\hat{f}(\mathbf{x}) = \begin{cases} C_1 & \text{si } \hat{g}(\mathbf{x}) > 0 \\ C_2 & \text{sinon} \end{cases}$$

Interprétation du svm (suite)



Le cas non linéairement séparable

- Nous avons traité précédemment le cas linéairement séparable.
- Si dans l'espace de description initial \mathbb{X} , les **classes se recouvrent** alors elles ne sont pas linéairement séparables et **le problème d'optimisation n'a pas de solution**.
- En effet, il est alors impossible de satisfaire toutes les contraintes :

$$\forall i, y_i(\mathbf{a}^\top \mathbf{x}_i + a_0) \geq 1$$

- On cherche alors un hyperplan qui continue à maximiser la marge mais tout en faisant le moins d'erreur possible.
- Pour ce faire, on intègre des **variables d'écart** $\xi_i \geq 0$ qui permettent des erreurs :

$$\forall i, y_i(\mathbf{a}^\top \mathbf{x}_i + a_0) \geq 1 - \xi_i$$

- On parle alors de "**soft margin**" ou de **méthodes discriminantes flexibles**.

svm (code R)

```
> library("e1071")
> c=as.factor(c)
> XX=data.frame(c,X)
> res_svm=svm(c~,data=XX,kernel="linear",cost=20)
> res_svm$index
[1] 6 14 56
> res_svm$coefs
      [,1]
[1,] 7.3271078
[2,] 0.3981978
[3,] -7.7253056
> res_svm$rho
[1] 0.380473
> res_svm$decision.values
      -1/1
1      3.0514436
2      5.4245154
...
100 -7.0483953
```

Le cas non linéairement séparable (suite)

$$\forall i, y_i(\mathbf{a}^\top \mathbf{x}_i + a_0) \geq 1 - \xi_i$$

- Nous remarquerons les cas particuliers suivants :
 - ▶ Si $\xi_i = 0$ alors il n'y a pas de problème de catégorisation avec \mathbf{x}_i .
 - ▶ Si $0 < \xi_i < 1$ alors \mathbf{x}_i est du bon côté de la frontière mais se situe dans la bande.
 - ▶ Si $\xi_i \geq 1$ alors \mathbf{x}_i est catégorisée de façon incorrecte.
- $|\{\mathbf{x}_i \in \mathbb{E} : \xi_i > 1\}|$ est le nb de vecteurs incorrectement classifiés.
- $|\{\mathbf{x}_i \in \mathbb{E} : \xi_i > 0\}|$ est le nb de vecteurs non linéairement séparables en considérant la marge.
- On définit alors le "**soft error**" :

$$\sum_i \xi_i$$

que l'on cherche à minimiser en l'intégrant dans la fonction objectif.

Hyperplan flexible de séparation optimale

- Nous avons le problème suivant :

$$\begin{aligned} \min_{a_0, \mathbf{a} \in \mathbb{R}^p, \xi \in \mathbb{R}^n} & \frac{1}{2} \|\mathbf{a}\|^2 + c \sum_{i=1}^n \xi_i \\ \text{s/c} & \forall i, y_i(\mathbf{a}^\top \mathbf{x}_i + a_0) \geq 1 - \xi_i \\ & \forall i, \xi_i \geq 0 \end{aligned}$$

où c est une constante positive tel un coefficient de pénalité, permettant de contrôler l'équilibre entre la maximisation de la marge et les erreurs. Nous remarquerons que pour un cas linéairement séparable les ξ_i sont nuls et donc " $c = \infty$ ".

- Le Lagrangien (primal) est alors donné par :

$$\begin{aligned} \text{lag}_p(a_0, \mathbf{a}, \xi, \alpha, \mu) & \\ = & \\ \frac{1}{2} \|\mathbf{a}\|^2 + c \sum_{i=1}^n \xi_i - \sum_{i=1}^n \alpha_i (y_i(\mathbf{a}^\top \mathbf{x}_i + a_0) - (1 - \xi_i)) - \sum_{i=1}^n \mu_i \xi_i & \\ \text{où } \alpha \in \mathbb{R}^{+n} \text{ et } \mu \in \mathbb{R}^{+n} \text{ sont les multiplicateurs de Lagrange.} & \end{aligned}$$

Lagrangien et problème dual (suite)

- Après simplification, on obtient le problème dual suivant :

$$\begin{aligned} \max_{\alpha \in \mathbb{R}^n} & \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j y_i y_j \mathbf{x}_i^\top \mathbf{x}_j \\ \text{s/c} & \forall i, 0 \leq \alpha_i \leq c \\ & \sum_{i=1}^n \alpha_i y_i = 0 \end{aligned}$$

- Nous obtenons la solution $\hat{\alpha}$ et le vecteur normal de la frontière de décision $\hat{\mathbf{a}}_{svm} \in \mathbb{X}$, est tel que :

$$\hat{\mathbf{a}}_{svm} = \sum_{i=1}^n \hat{\alpha}_i y_i \mathbf{x}_i$$

- Les conditions complémentaires de KKT permettent également de caractériser la solution optimale obtenue vis à vis du primal :

$$\begin{cases} \forall i, \alpha_i (y_i(\mathbf{a}^\top \mathbf{x}_i + a_0) - (1 - \xi_i)) = 0 \\ \forall i, \mu_i \xi_i = 0 \end{cases}$$

Lagrangien et problème dual

- On doit minimiser le Lagrangien par rapport à a_0 , \mathbf{a} , ξ et le maximiser par rapport à α et μ (point selle, propriété de dualité faible).
- Comme précédemment, on peut de façon équivalente maximiser le Lagrangien par rapport à α et μ sous les contraintes que les gradients de lag_p par rapport aux variables primales soient nuls :

$$\begin{cases} \frac{\partial \text{lag}_p}{\partial a_0} = 0 \\ \frac{\partial \text{lag}_p}{\partial \mathbf{a}} = \mathbf{0} \\ \frac{\partial \text{lag}_p}{\partial \xi} = \mathbf{0} \end{cases} \Leftrightarrow \begin{cases} \sum_{i=1}^n \alpha_i y_i = 0 \\ \mathbf{a} - \sum_{i=1}^n \alpha_i y_i \mathbf{x}_i = \mathbf{0} \\ c \mathbf{1} - \alpha - \mu = \mathbf{0} \end{cases}$$

où $\mathbf{1}$ est le vecteur de taille $(n \times 1)$ rempli de 1.

- On obtient les relations suivantes $\sum_{i=1}^n \alpha_i y_i = 0$, $\mathbf{a} = \sum_{i=1}^n \alpha_i y_i \mathbf{x}_i$ et $\forall i, \alpha_i = c - \mu_i$.
- Comme $\forall i, \mu_i \geq 0$, la dernière condition implique que $\forall i, 0 \leq \alpha_i \leq c$.

Lagrangien et problème dual (suite)

- Le vecteur normal de la frontière étant :

$$\hat{\mathbf{a}}_{svm} = \sum_{i=1}^n \hat{\alpha}_i y_i \mathbf{x}_i$$

- Nous avons les interprétations suivantes :
 - Si $\hat{\alpha}_i > 0$ alors \mathbf{x}_i participe à la définition de $\hat{\mathbf{a}}_{svm}$.
 - Si $\hat{\mu}_i > 0$ alors $0 \leq \hat{\alpha}_i < c$ (car $\alpha_i = c - \mu_i$).
Par ailleurs, comme $\hat{\mu}_i \hat{\xi}_i = 0$ (KKT), alors $\hat{\mu}_i > 0 \Rightarrow \hat{\xi}_i = 0 \Rightarrow \hat{\alpha}_i (y_i(\hat{\mathbf{a}}_{svm}^\top \mathbf{x}_i + \hat{a}_{svm,0}) - 1) = 0$.
Si de plus $\hat{\alpha}_i > 0$, alors \mathbf{x}_i est sur une frontière de la bande puisque (KKT) $y_i(\hat{\mathbf{a}}_{svm}^\top \mathbf{x}_i + \hat{a}_{svm,0}) = 1$.
 - Si $\hat{\xi}_i > 0$ alors (KKT) $\hat{\mu}_i = 0$ et dans ce cas $\hat{\alpha}_i = c > 0$.
Par ailleurs, $y_i(\hat{\mathbf{a}}_{svm}^\top \mathbf{x}_i + \hat{a}_{svm,0}) = 1 - \hat{\xi}_i < 1$ et en fonction de la valeur de $\hat{\xi}_i$ nous avons : si $0 < \hat{\xi}_i < 1$ alors \mathbf{x}_i est dans l'intérieur de la bande et du bon côté ; si $1 < \hat{\xi}_i \leq 2$ alors \mathbf{x}_i est dans l'intérieur de la bande mais du mauvais côté et si $2 < \hat{\xi}_i$ alors \mathbf{x}_i est à l'extérieur de la bande et du mauvais côté.

Lagrangien et problème dual (suite)

- On obtient $\hat{a}_{svm,0}$ à l'aide de l'équation suivante pour n'importe quel vecteur support (càd tel que $0 < \hat{\alpha}_i < c$) :

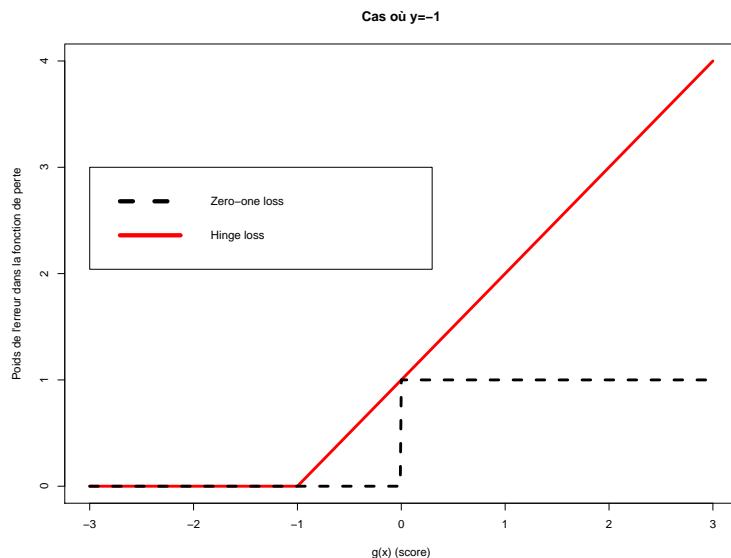
$$\hat{a}_{svm,0} = y_i - \hat{\mathbf{a}}_{svm}^\top \mathbf{x}_i$$

- La **fonction de décision** $\hat{f}(\mathbf{x})$ dépend alors de la fonction $\hat{g}(\mathbf{x}) = \hat{\mathbf{a}}_{svm}^\top \mathbf{x} + \hat{a}_{svm,0}$:

$$\hat{f}(\mathbf{x}) = \begin{cases} C_1 & \text{si } \hat{g}(\mathbf{x}) > 0 \\ C_2 & \text{sinon} \end{cases}$$

- Le problème dual est plus simple à résoudre que le problème primal.
- Le problème **dual** permet de faire dépendre la **complexité** du problème **en fonction de** n plutôt qu'en fonction de p !
- Les svm peuvent ainsi traiter les **problèmes de grande dimension** ($n \ll p$) plus efficacement que les modèles linéaires précédents !

Fonction "hinge loss"



Pénalisation ridge et le "hinge loss"

- On remarquera la similitude entre la fonction objectif du svm et les modèles pénalisés précédents :

$$\min_{a_0, \mathbf{a} \in \mathbb{R}^p, \xi \in \mathbb{R}^n} \frac{1}{2} \underbrace{\|\mathbf{a}\|^2}_{\text{pénalité}} + c \underbrace{\sum_{i=1}^n \xi_i}_{\text{perte}}$$

- Rappelons que par définition :

$$\forall i, y_i (\underbrace{\mathbf{a}^\top \mathbf{x}_i + a_0}_{g(\mathbf{x}_i) \text{ (score)}}) \geq 1 - \xi_i$$

ξ_i est positive et mesure l'écart entre 1 et $y_i g(\mathbf{x}_i)$, on a donc :

$$\forall i, \xi_i = \max(0, 1 - y_i (\mathbf{a}^\top \mathbf{x}_i + a_0))$$

- Le **"soft error"** est également appelé **"hinge loss"** :

$$\sum_i \xi_i = \sum_i \max(0, 1 - y_i g(\mathbf{x}_i))$$

Choix du paramètre c

- Le svm nécessite également le "tuning" du paramètre c qui arbitre entre la fonction de perte et la fonction de pénalité :

$$\min_{a_0, \mathbf{a} \in \mathbb{R}^p, \xi \in \mathbb{R}^n} \frac{1}{2} \|\mathbf{a}\|^2 + c \sum_{i=1}^n \xi_i$$

- c peut être sélectionné par validation croisée comme indiqué en slide 147 (mais en utilisant le taux d'erreur comme critère).
- Il existe aussi des travaux pour déterminer le **chemin de régularisation** d'un svm, c'est-à-dire le calcul de $\hat{\mathbf{a}}_{svm}(c)$ pour $c \in [0, \infty]$.
- Dans [Hastie et al., 2004], les auteurs montrent que $\hat{\mathbf{a}}_{svm}(c)$ est linéaire par morceaux (comme le lasso). Leur algorithme est inspiré de lars.

svm (code R)

```
library(svmpath)
res_svmpath=svmpath(x = X,y = c,trace = TRUE)
summary(res_svmpath)
```

```
> summary(res_svmpath)
```

```
Call:
svmpath(x = X, y = c, trace = TRUE)
```

Number of steps: 146

Selected steps:

	Lambda	Error	Size.Elbow	Support	SumEps
1	12.319518	6	2	100	60.25142
50	2.385540	5	3	64	29.16954
100	0.391334	5	3	34	16.97741
146	0.001727	5	3	13	11.50147

Expansions de base et noyaux

- Si le problème n'est pas linéairement séparable, nous pouvons appliquer une **expansion de base** de \mathbb{X} dans un espace étendu \mathbb{F} :

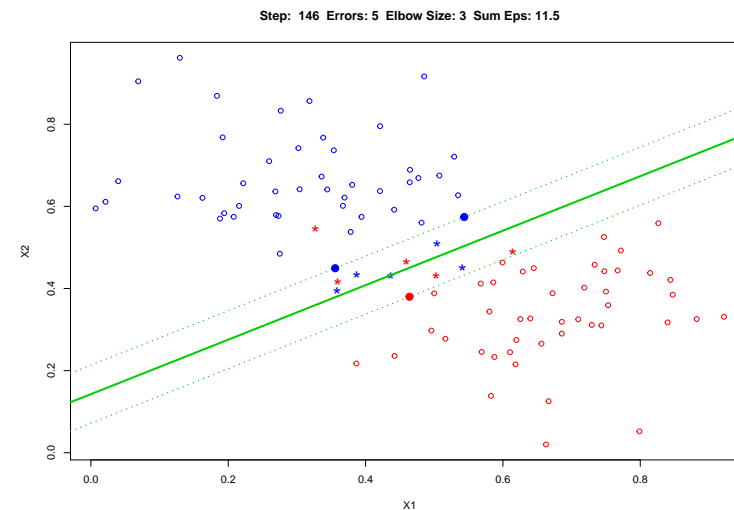
$$\phi : \mathbb{X} \rightarrow \mathbb{F}$$

- Dans ce cas un modèle linéaire dans \mathbb{F} correspond à un modèle non linéaire dans \mathbb{X} . Donc au lieu de manipuler les vecteurs $\mathbf{x} \in \mathbb{X}$, on manipule des vecteurs $\phi(\mathbf{x}) \in \mathbb{F}$.
- Les développements précédents sont les mêmes pour obtenir le problème dual suivant :

$$\begin{aligned} & \max_{\alpha \in \mathbb{F}} \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j y_i y_j \phi(\mathbf{x}_i)^\top \phi(\mathbf{x}_j) \\ \text{s/c } & \forall i, 0 \leq \alpha_i \leq c \\ & \sum_{i=1}^n \alpha_i y_i = 0 \end{aligned}$$

svm (code R)

```
plot(res_svmpath)
```



Expansions de base et noyaux

- La solution du dual est donnée par :

$$\hat{\mathbf{a}}_{svm} = \sum_{i=1}^n \hat{\alpha}_i y_i \phi(\mathbf{x}_i)$$

où $\hat{\mathbf{a}}_{svm} \in \mathbb{F}$.

- Par ailleurs :

$$\hat{\mathbf{a}}_{svm,0} = y_i - \hat{\mathbf{a}}_{svm}^\top \mathbf{x}_i$$

où \mathbf{x}_i est un vecteur support càd tel que $\hat{\alpha}_i > 0$.

- La fonction de score ou de discrimination du svm est donnée par :

$$\hat{g}(\mathbf{x}) = \hat{\mathbf{a}}_{svm}^\top \phi(\mathbf{x}) + \hat{\mathbf{a}}_{svm,0}$$

- La fonction de décision reste :

$$\hat{f}(\mathbf{x}) = \begin{cases} C_1 & \text{si } \hat{g}(\mathbf{x}) > 0 \\ C_2 & \text{sinon} \end{cases}$$

Expansions de base et noyaux

- Regardons de plus près la fonction de score du svm :

$$\begin{aligned}\hat{g}(\mathbf{x}) &= \hat{\mathbf{a}}_{svm}^T \phi(\mathbf{x}) + \hat{a}_{svm,0} \\ &= \underbrace{\sum_{i=1}^n \hat{\alpha}_i y_i \phi(\mathbf{x}_i)^T \phi(\mathbf{x})}_{\hat{\mathbf{a}}_{svm}^T} + \hat{a}_{svm,0}\end{aligned}$$

- En utilisant le dual, les éléments importants dans le cadre du svm peuvent s'exprimer en termes de **produit scalaire** dans l'espace étendu \mathbb{F} : $\phi(\mathbf{x}_i)^T \phi(\mathbf{x})$.
- Posons alors $K : \mathbb{X} \times \mathbb{X} \rightarrow \mathbb{R}$ tel que :

$$K(\mathbf{x}_i, \mathbf{x}) = \phi(\mathbf{x}_i)^T \phi(\mathbf{x})$$

Les noyaux

- Attention $K(.,.)$ ne doit pas être confondu avec le noyau de Parzen (régression non paramétrique) !
- $K(\mathbf{x}, \mathbf{y})$ représente un produit scalaire et doit satisfaire plusieurs types de contraintes.
- Notons \mathbf{K} la matrice carrée de taille $(n \times n)$ de produits scalaires dont le terme général est tel que :

$$\begin{aligned}\mathbf{K}_{ij} &= K(\mathbf{x}_i, \mathbf{x}_j) \\ &= \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j) \\ &= \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle\end{aligned}$$

- On appelle une matrice de produits scalaires une **matrice de Gram**
- La matrice \mathbf{K} doit alors satisfaire les propriétés suivantes :
 - ▶ Symétrie : $\forall i, j, \mathbf{K}_{ij} = \mathbf{K}_{ji}$.
 - ▶ Semi-définie positivité : $\forall \mathbf{z} \in \mathbb{R}^n, \mathbf{z}^T \mathbf{K} \mathbf{z} \geq 0$.

Expansions de base et noyaux

- Le problème d'optimisation dual s'écrit donc :

$$\begin{aligned}\max_{\alpha \in \mathbb{F}} & \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) \\ \text{s.t.} & \forall i, 0 \leq \alpha_i \leq c \\ & \sum_{i=1}^n \alpha_i y_i = 0\end{aligned}$$

- La fonction de score obtenue également :

$$\hat{g}(\mathbf{x}) = \sum_{i=1}^n \hat{\alpha}_i y_i K(\mathbf{x}_i, \mathbf{x}) + \hat{a}_{svm,0}$$

- La fonction $K(.,.)$ est appelée **noyau** ("kernel") et les méthodes qui remplacent le produit scalaire dans \mathbb{X} par un produit scalaire dans un espace issu d'une expansion de base \mathbb{F} sont dites **méthodes à noyaux** ("kernel methods" ou "kernel machines").
- L'intérêt de ces fonctions est qu'elles ne nécessitent pas de représenter explicitement \mathbf{x} dans \mathbb{F} (càd on ne calcule jamais $\phi(\mathbf{x})$ - "kernel trick").

Les noyaux (suite)

- Soit dans $\mathbb{X} = \mathbb{R}^2$ deux vecteurs $\mathbf{x} = (x_1, x_2)$ et $\mathbf{y} = (y_1, y_2)$.
- Exemple classique de noyau :

$$\begin{aligned}K(\mathbf{x}, \mathbf{y}) &= (\langle \mathbf{x}, \mathbf{y} \rangle + 1)^2 \\ &= (x_1 y_1 + x_2 y_2 + 1)^2 \\ &= (x_1 y_1)^2 + (x_2 y_2)^2 + 1 + 2x_1 y_1 x_2 y_2 + 2x_1 y_1 + 2x_2 y_2\end{aligned}$$

- Ce noyau correspond à l'expansion de base ϕ suivante :

$$\phi(\mathbf{x}) = (x_1^2, x_2^2, 1, \sqrt{2}x_1 x_2, \sqrt{2}x_1, \sqrt{2}x_2)$$

- On vérifie bien en effet que : $K(\mathbf{x}, \mathbf{y}) = \phi(\mathbf{x})^T \phi(\mathbf{y})$.
- En utilisant K , la complexité de calcul reste en $O(\dim(\mathbb{X}))$ plutôt que $O(\dim(\mathbb{F}))$!

Les noyaux (suite)

- Il existe plusieurs familles de noyaux :
 - Les **noyaux polynomiaux** de degré d ("Polynomial kernels") :

$$K(\mathbf{x}, \mathbf{y}) = (\langle \mathbf{x}, \mathbf{y} \rangle + 1)^d$$

Ces noyaux sont relatifs à une expansion de base reposant sur des polynômes de degré d des composantes initiales. Le cas $d = 1$ est appelé noyau linéaire (produit scalaire dans l'espace initial \mathbb{X}).

- Les **fonctions à bases radiales** ("Radial basis functions" (RBF)) :

$$K(\mathbf{x}, \mathbf{y}) = \exp\left(-\frac{\|\mathbf{x} - \mathbf{y}\|^2}{2\sigma^2}\right)$$

Ces noyaux sont (pour le coup) en lien avec le noyau de Parzen puisqu'ils reposent sur la notion de voisinage (hypersphère de centre \mathbf{x} et de rayon σ^2). Pour autant, ce ne sont pas des distributions de probabilité et leur interprétation reste en terme d'expansion de bases.

Les noyaux (suite)

- Expansion de base associée au noyau RBF :

$$\begin{aligned} K(\mathbf{x}, \mathbf{y}) &= \exp\left(-\frac{\|\mathbf{x} - \mathbf{y}\|^2}{2\sigma^2}\right) \\ &= \exp\left(-\frac{\langle \mathbf{x}, \mathbf{x} \rangle + \langle \mathbf{y}, \mathbf{y} \rangle - 2\langle \mathbf{x}, \mathbf{y} \rangle}{2\sigma^2}\right) \\ &= \exp\left(-\frac{\langle \mathbf{x}, \mathbf{x} \rangle}{2\sigma^2}\right) \exp\left(-\frac{\langle \mathbf{y}, \mathbf{y} \rangle}{2\sigma^2}\right) \exp\left(\frac{\langle \mathbf{x}, \mathbf{y} \rangle}{\sigma^2}\right) \\ &= \frac{\exp\left(\frac{\langle \mathbf{x}, \mathbf{y} \rangle}{\sigma^2}\right)}{\sqrt{\exp\left(\frac{\langle \mathbf{x}, \mathbf{x} \rangle}{\sigma^2}\right) \exp\left(\frac{\langle \mathbf{y}, \mathbf{y} \rangle}{\sigma^2}\right)}} \end{aligned}$$

Les noyaux (suite)

- Rappelons le développement en séries de Taylor-Maclaurin de \exp :

$$\exp(x) = \sum_{k=0}^{\infty} \frac{x^k}{k!}$$

- Dans ce cas on a :

$$\begin{aligned} \exp\left(\frac{\langle \mathbf{x}, \mathbf{y} \rangle}{\sigma^2}\right) &= \exp(\langle \mathbf{x}, \mathbf{y} \rangle)^{1/\sigma^2} \\ &= \left(\sum_{k=0}^{\infty} \frac{\langle \mathbf{x}, \mathbf{y} \rangle^k}{k!}\right)^{1/\sigma^2} \end{aligned}$$

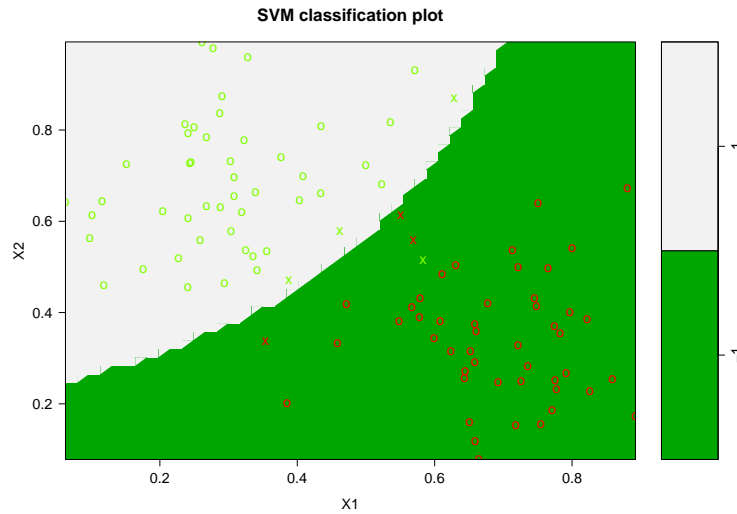
- Le noyau RBF correspond donc à une mesure **cosinus** dans un espace de **dimension infinie**.

Les noyaux (suite)

- Les noyaux permettent donc de travailler implicitement dans un espace \mathbb{F} qui peut être de très grande dimension.
- En projetant les données dans \mathbb{F} , on espère pouvoir rendre le problème davantage linéairement séparable que dans \mathbb{X} . Ceci permettrait d'utiliser le concept d'optimisation de la marge dans un espace plus adéquat afin d'avoir de meilleures performances.
- Dans l'espace \mathbb{F} on obtient donc une frontière linéaire qui s'exprime à l'aide de vecteurs supports : $\hat{g}(\mathbf{x}) = \sum_{i=1}^n \hat{\alpha}_i y_i K(\mathbf{x}_i, \mathbf{x}) + \hat{a}_{svm,0}$.
- En revanche, dans l'espace initial \mathbb{X} on obtient une frontière de décision **non linéaire**.
- Pour un noyau polynomial, plus le paramètre d est petit, plus la frontière dans \mathbb{X} que l'on obtient est lisse ("smooth").
- Pour un noyau RBF, plus le paramètre σ^2 est grand, plus la frontière dans \mathbb{X} que l'on obtient est lisse.
- Les paramètres des noyaux peuvent être estimés par validation croisée.

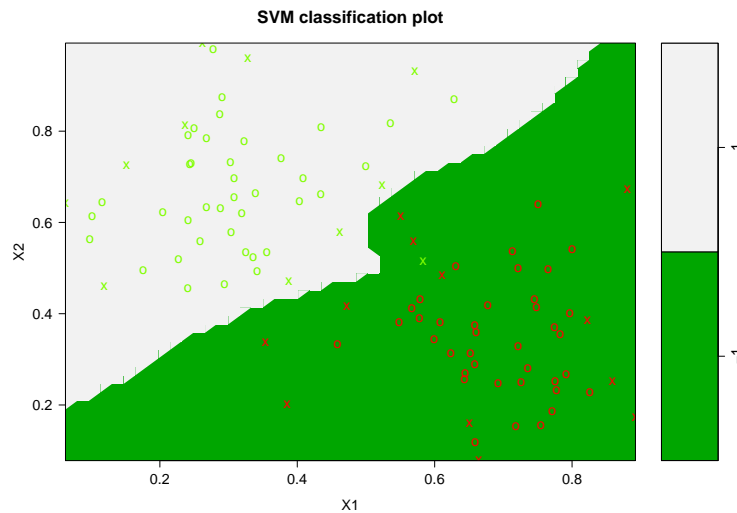
Exemple (suite)

- Avec un noyau polynomial $K(\mathbf{x}, \mathbf{y}) = (\langle \mathbf{x}, \mathbf{y} \rangle + 1)^2$ ($d = 2$).



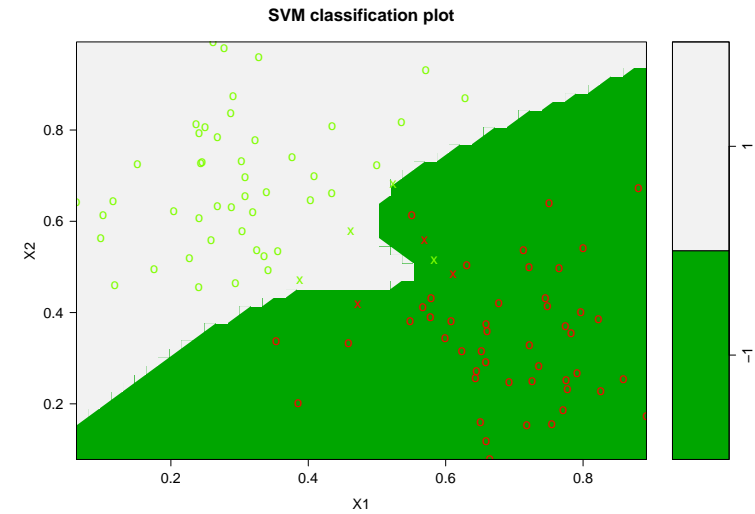
Exemple (suite)

- Avec un noyau RBF $K(\mathbf{x}, \mathbf{y}) = \exp\left(-\frac{\|\mathbf{x}-\mathbf{y}\|^2}{0.5}\right)$ ($\sigma^2 = 0.25$)



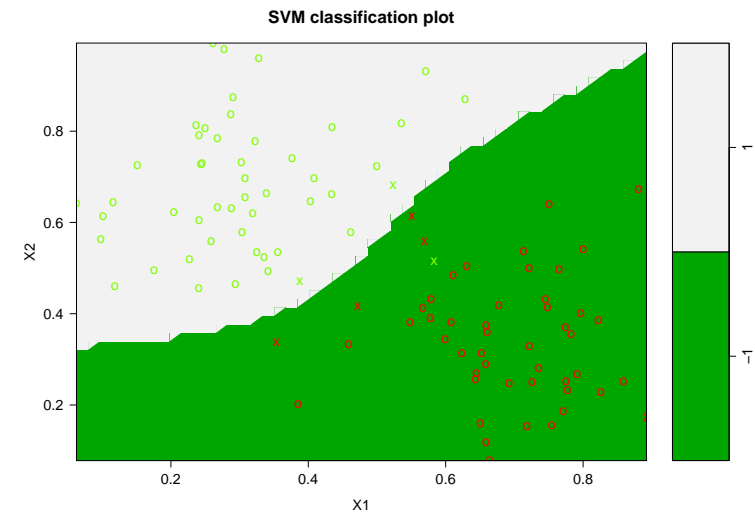
Exemple (suite)

- Avec un noyau polynomial $K(\mathbf{x}, \mathbf{y}) = (\langle \mathbf{x}, \mathbf{y} \rangle + 1)^{10}$ ($d = 10$)



Exemple (suite)

- Avec un noyau RBF $K(\mathbf{x}, \mathbf{y}) = \exp\left(-\frac{\|\mathbf{x}-\mathbf{y}\|^2}{2}\right)$ ($\sigma^2 = 1$)



Construction de noyaux

- Soient K_1 et K_2 deux noyaux alors les fonctions K suivantes forment également des noyaux valides (symétriques et s.d.p.) :
 - ▶ $K(\mathbf{x}, \mathbf{y}) = aK_1(\mathbf{x}, \mathbf{y})$ où $a > 0$.
 - ▶ $K(\mathbf{x}, \mathbf{y}) = K_1(\mathbf{x}, \mathbf{y}) + K_2(\mathbf{x}, \mathbf{y})$.
 - ▶ $K(\mathbf{x}, \mathbf{y}) = K_1(\mathbf{x}, \mathbf{y})K_2(\mathbf{x}, \mathbf{y})$.
 - ▶ $K(\mathbf{x}, \mathbf{y}) = \mathbf{x}^T \mathbf{A} \mathbf{y}$ où $\mathbf{A} = \mathbf{A}^T$ et $\mathbf{A} \geq 0$ (càd s.d.p.).
 - ▶ $K(\mathbf{x}, \mathbf{y}) = p(K_1(\mathbf{x}, \mathbf{y}))$ où p est un polynôme à coefficients positifs.
 - ▶ $K(\mathbf{x}, \mathbf{y}) = \exp(K_1(\mathbf{x}, \mathbf{y}))$.
- Nous avons vu la catégorisation binaire. Dans le cas **multiclasse** on pourra appliquer les stratégies abordées précédemment :
 - ▶ "un contre tous" avec q fonctions de score et on prend ensuite le max,
 - ▶ "un contre un" avec à $q(q-1)/2$ classifieurs et on fait ensuite des votes. Un DAG (Directed Acyclic Graph) permet également de prendre la décision finale.
 - ▶ Il est également possible d'apprendre de façon jointe q classifieurs [Weston et al., 1999] ou d'appliquer l'approche ECOC.

Fonction de perte

- En comparaison des méthodes précédentes, les svm cherchent à minimiser la fonction de perte "hinge" ou " **ϵ -insensitive loss**" :

$$\begin{aligned} \ell_\epsilon(f(\mathbf{x}), y) &= \begin{cases} 0 & \text{si } |y - f(\mathbf{x})| < \epsilon \\ |y - f(\mathbf{x})| - \epsilon & \text{sinon} \end{cases} \\ &= \max(0, |y - f(\mathbf{x})| - \epsilon) \end{aligned}$$

où $\epsilon > 0$ est un paramètre relatif à une **marge** d'erreur.

- On peut interpréter ℓ_ϵ de la façon suivante :
 - ▶ On tolère des erreurs d'ajustement jusqu'à une quantité ϵ .
 - ▶ Au delà de ϵ le poids d'une erreur est linéaire et non quadratique.
 - ▶ ℓ_ϵ est plus robuste vis à vis du bruit.
- Les svm pour la régression combinent $\ell_\epsilon(f(\mathbf{x}), y)$ et la fonction de pénalité quadratique :

$$\min_{a_0, \mathbf{a} \in \mathbb{R}^p} c \sum_{i=1}^n \ell_\epsilon(f(\mathbf{x}_i), y_i) + \|\mathbf{a}\|^2$$

Les svm appliqués au problème de régression

- Nous supposons maintenant que $\mathbb{Y} = \mathbb{R}$.
- Les idées de marge, de variables d'écart, de noyaux... peuvent être généralisées pour le problème de régression.
- Supposons d'abord un noyau linéaire. Nous avons alors la famille d'hypothèses \mathbb{H} qui est l'ensemble des fonctions de type :

$$f(\mathbf{x}) = a_0 + \mathbf{a}^T \mathbf{x}$$

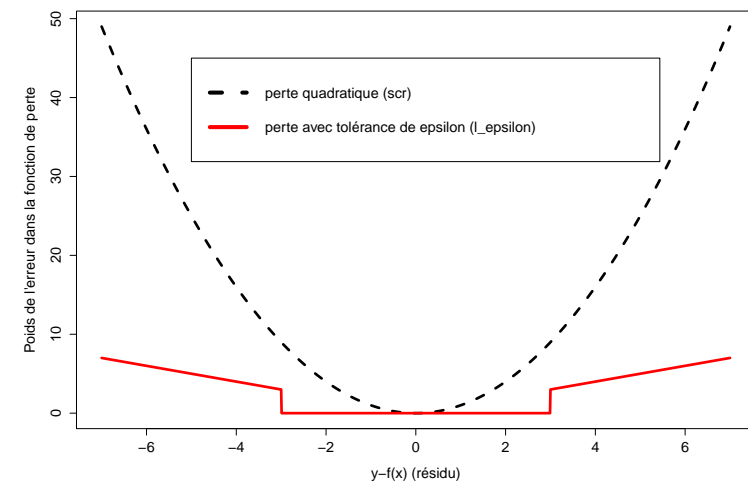
- Rappelons que la régression linéaire et le problème des MCO sont :

$$\min_{a_0, \mathbf{a} \in \mathbb{R}^p} \sum_{i=1}^n (y_i - (a_0 + \mathbf{a}^T \mathbf{x}))^2$$

- Par ailleurs, la régression ridge ajoute au *scr* une fonction de pénalité :

$$\min_{a_0, \mathbf{a} \in \mathbb{R}^p} \sum_{i=1}^n (y_i - (a_0 + \mathbf{a}^T \mathbf{x}))^2 + \lambda \|\mathbf{a}\|_{\ell_2}^2$$

Fonction de perte (suite)



Fonction de perte (suite)

- Sortir de l'intervalle de tolérance de taille $\epsilon > 0$ se produit quand :
 - $a_0 + \mathbf{a}^\top \mathbf{x}_i > y_i + \epsilon$: le modèle prédit une valeur trop forte.
 - $a_0 + \mathbf{a}^\top \mathbf{x}_i < y_i - \epsilon$: le modèle prédit une valeur trop faible.
- On introduit des variables d'écarts pour formaliser ces "sorties" du tube. Soient $\forall i, \xi_i^+ \geq 0$ et $\xi_i^- \geq 0$, les "sorties" possibles sont alors :

$$\begin{cases} (a_0 + \mathbf{a}^\top \mathbf{x}_i) - y_i > \epsilon + \xi_i^+ \\ y_i - (a_0 + \mathbf{a}^\top \mathbf{x}_i) > \epsilon + \xi_i^- \end{cases}$$

- On voit que $|y_i - (a_0 + \mathbf{a}^\top \mathbf{x}_i)| \leq \epsilon \Leftrightarrow \xi_i^+ = \xi_i^- = 0$.
- Minimiser les variables d'écart est équivalent à minimiser l_ϵ .
- Le problème peut donc se reformuler de façon équivalente comme :

$$\begin{aligned} \min_{a_0, \mathbf{a} \in \mathbb{R}^p, \xi^+, \xi^- \in \mathbb{R}^n} & \frac{1}{2} \|\mathbf{a}\|^2 + c \sum_{i=1}^n (\xi_i^+ + \xi_i^-) \\ \text{s.c.} & \forall i, (a_0 + \mathbf{a}^\top \mathbf{x}_i) - y_i \leq \epsilon + \xi_i^+ \\ & \forall i, y_i - (a_0 + \mathbf{a}^\top \mathbf{x}_i) \leq \epsilon + \xi_i^- \\ & \forall i, \xi_i^+, \xi_i^- \geq 0 \end{aligned}$$

Lagrangien et problème dual (suite)

- En injectant les relations précédentes dans la fonction objectif, on obtient le problème dual suivant :

$$\begin{aligned} \max_{\alpha^+, \alpha^- \in \mathbb{R}^n} & -\frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n (\alpha_i^- - \alpha_i^+) (\alpha_j^- - \alpha_j^+) \mathbf{x}_i^\top \mathbf{x}_j \\ & - \epsilon \sum_{i=1}^n (\alpha_i^- + \alpha_i^+) + \sum_{i=1}^n (\alpha_i^- - \alpha_i^+) y_i \\ \text{s.c.} & \forall i, 0 \leq \alpha_i^+ \leq c \\ & \forall i, 0 \leq \alpha_i^- \leq c \\ & \sum_{i=1}^n (\alpha_i^+ - \alpha_i^-) = 0 \end{aligned}$$

- Une fois résolu ce problème quadratique contraint, on obtient la fonction de prédiction suivante qui dépend donc de vecteurs supports :

$$\begin{aligned} \hat{f}(\mathbf{x}) &= \hat{a}_0 + \hat{\mathbf{a}}^\top \mathbf{x} \\ &= \hat{a}_{svm,0} + \sum_{i=1}^n (\hat{\alpha}_i^- - \hat{\alpha}_i^+) \mathbf{x}_i^\top \mathbf{x} \end{aligned}$$

Lagrangien et problème dual

- Le Lagrangien (primal) dépend des variables primales $a_0, \mathbf{a}, \xi^+, \xi^-$ et des multiplicateurs de Lagrange $\alpha^+, \alpha^-, \mu^+, \mu^-$ qui sont des vecteurs de \mathbb{R}^n . Il est donné par :

$$\begin{aligned} lag_p &= \frac{1}{2} \|\mathbf{a}\|^2 + c \sum_{i=1}^n (\xi_i^+ + \xi_i^-) \\ &+ \sum_i \alpha_i^+ ((a_0 + \mathbf{a}^\top \mathbf{x}_i) - y_i - \epsilon - \xi_i^+) \\ &+ \sum_i \alpha_i^- (y_i - (a_0 + \mathbf{a}^\top \mathbf{x}_i) - \epsilon - \xi_i^-) \\ &- \sum_i (\mu_i^+ \xi_i^+ + \mu_i^- \xi_i^-) \end{aligned}$$

- A l'optimum, les gradients de lag_p par rapport aux variables primales sont nuls :

$$\begin{cases} \frac{\partial lag_p}{\partial a_0} = 0 \\ \frac{\partial lag_p}{\partial \mathbf{a}} = \mathbf{0} \\ \frac{\partial lag_p}{\partial \xi^+} = \mathbf{0} \\ \frac{\partial lag_p}{\partial \xi^-} = \mathbf{0} \end{cases} \Leftrightarrow \begin{cases} \sum_{i=1}^n (\alpha_i^+ - \alpha_i^-) = 0 \\ \mathbf{a} - \sum_{i=1}^n (\alpha_i^- - \alpha_i^+) \mathbf{x}_i = \mathbf{0} \\ c\mathbf{1} - \alpha^+ - \mu^+ = \mathbf{0} \\ c\mathbf{1} - \alpha^- - \mu^- = \mathbf{0} \end{cases}$$

où $\mathbf{1}$ est le vecteur de taille $(n \times 1)$ rempli de 1

Lagrangien et problème dual (suite)

- Les relations $\alpha_i^\pm = c - \mu_i^\pm, \forall i$ et les conditions complémentaires de KKT nous permettent de voir que :

- Si $\alpha_i^+ = \alpha_i^- = 0$ alors $\mu_i^+ = \mu_i^- = c > 0$ et donc (KKT) $\xi_i^+ = \xi_i^- = 0$. Si $\alpha_i^+ = \alpha_i^- = 0$ alors on a également (KKT) $\epsilon + y_i - (a_0 + \mathbf{a}^\top \mathbf{x}_i) > 0$ et $\epsilon - y_i + (a_0 + \mathbf{a}^\top \mathbf{x}_i) > 0$. Dans ce cas, \mathbf{x}_i est donc dans le tube.
- Si $\hat{\alpha}_i^+ > 0$ ou (exclusif) $\hat{\alpha}_i^- > 0$, on a alors deux sous-cas :
 - Si $\hat{\alpha}_i^+ \neq c$ ou $\hat{\alpha}_i^- \neq c$ alors resp. $\mu_i^+ > 0$ ou $\mu_i^- > 0$ et donc (KKT) $\xi_i^+ = 0$ ou $\xi_i^- = 0$. \mathbf{x}_i est donc sur une frontière du tube.
 - Si $\hat{\alpha}_i^+ = c$ ou $\hat{\alpha}_i^- = c$ alors resp. $\mu_i^+ = 0$ ou $\mu_i^- = 0$ et donc (KKT) $\xi_i^+ > 0$ ou $\xi_i^- > 0$. \mathbf{x}_i est donc à l'extérieur du tube.

- Pour la régression, ce sont **les points sur ou à l'extérieur du tube qui sont des vecteurs supports**.

- Les points \mathbf{x}_i sur la frontière ($0 < \hat{\alpha}_i^+ < c$ ou $0 < \hat{\alpha}_i^- < c$) permettent de calculer $\hat{a}_{svm,0}$ puisque dans ce cas, nous avons (KKT) :

$$\epsilon + y_i - (\hat{a}_{svm,0} + \hat{\mathbf{a}}_{svm}^\top \mathbf{x}_i) = 0 \text{ ou } \epsilon - y_i + (\hat{a}_{svm,0} + \hat{\mathbf{a}}_{svm}^\top \mathbf{x}_i) = 0$$

svm (code R)

```
> #svm régression
> library("e1071")
> eps=0.2
> res_svm_lin=svm(y~x,data=data.frame(x,y),kernel="linear",cost=10,epsilon=eps,type="eps-regr)
> res_svm_pol=svm(y~x,data=data.frame(x,y),kernel="polynomial",degree=2,cost=10,epsilon=eps,t
> res_svm_rbf=svm(y~x,data=data.frame(x,y),kernel="radial",gamma=2,cost=10,epsilon=eps,type="
> #Vecteurs supports et coefficients associés
> data.frame(res_svm_lin$index,res_svm_lin$coefs,x[res_svm_lin$index],y[res_svm_lin$index])
  res_svm_lin.index res_svm_lin.coefs x.res_svm_lin.index y.res_svm_lin.index.
1          1          10.00000 -1.4733525          0.1362840
2          2          10.00000 -0.8034692          0.7532798
3          3          10.00000  0.4577512          0.9431111
4          4         -10.00000  2.5648452         -0.8626718
5          6          -3.08357  2.5031562         -0.7718411
6          7         -10.00000  2.7939771         -0.9898869
7          8          10.00000  1.0103223          0.3544117
8          9          10.00000  0.8112475          0.7785888
9         10         -10.00000 -2.7533781         -0.9291811
10         11         -6.91643 -1.8474162         -0.2744008
11         12         -10.00000 -2.0322539         -0.3697468
```

Les noyaux

- Comme pour la catégorisation, le problème dual et la fonction de discrimination s'expriment par le biais de produits scalaires.
- Nous pouvons donc étendre l'approche à des noyaux conduisant alors à des modèles non linéaires dans \mathbb{X} .
- Formellement, les svm appliquées au problème de régression consistent à résoudre le problème suivant :

$$\max_{\alpha^+, \alpha^- \in \mathbb{R}^n} -\frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n (\alpha_i^- - \alpha_i^+) (\alpha_j^- - \alpha_j^+) K(\mathbf{x}_i, \mathbf{x}_j) - \epsilon \sum_{i=1}^n (\alpha_i^- + \alpha_i^+) + \sum_{i=1}^n (\alpha_i^- - \alpha_i^+) y_i$$

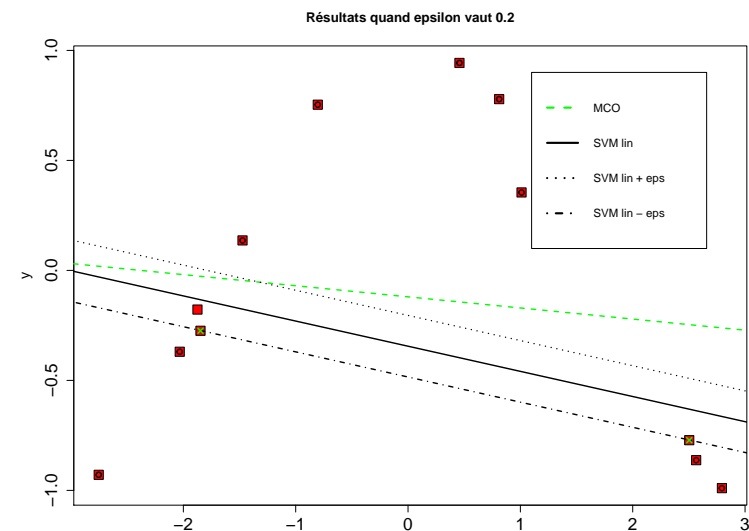
slc $\forall i, 0 \leq \alpha_i^+ \leq c$
 $\forall i, 0 \leq \alpha_i^- \leq c$
 $\sum_{i=1}^n (\alpha_i^+ - \alpha_i^-) = 0$

- La fonction de prédiction est alors :

$$\hat{f}(\mathbf{x}) = \hat{a}_{svm,0} + \sum_{i=1}^n (\hat{\alpha}_i^- - \hat{\alpha}_i^+) K(\mathbf{x}_i, \mathbf{x})$$

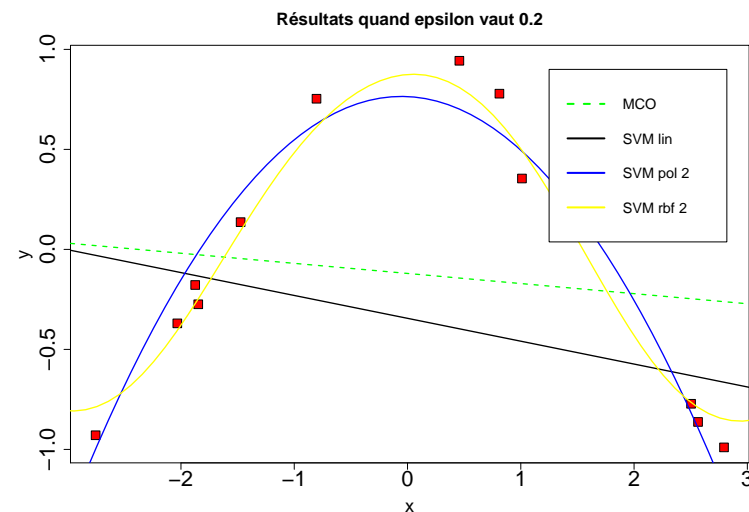
Exemple

- Avec un noyau linéaire



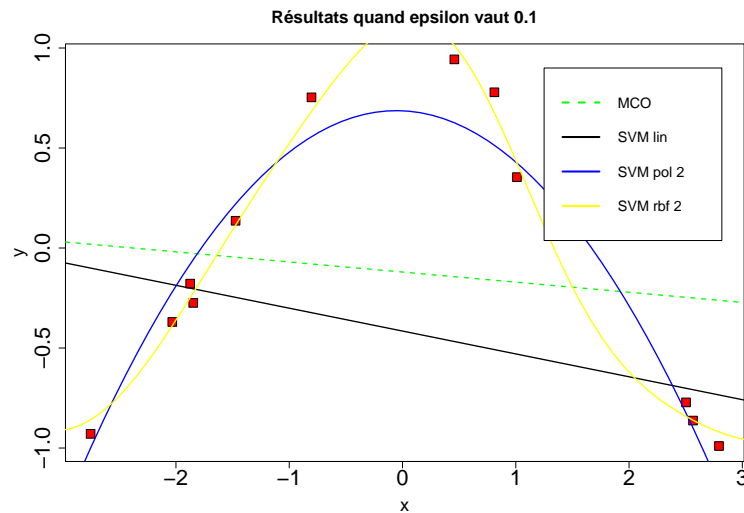
Exemple (suite)

- Avec différents types de noyaux et $\epsilon = 0.2$.



Exemple (suite)

- Avec différents types de noyaux et $\epsilon = 0.1$ (moins de tolérance).



Introduction

- Un arbre décisionnel est une **structure hiérarchique** qui peut être représenté par un graphe dont les nœuds représentent des sous-espaces de \mathbb{X} .
- La racine contient tout \mathbb{X} tandis que les feuilles des régions unitaires.
- Entre la racine et les feuilles, les nœuds intermédiaires représentent des **régions emboîtées** : $\mathbb{X} = \mathbb{X}^1 \oplus \dots \oplus \mathbb{X}^m \oplus \dots \oplus \mathbb{X}^{p'}$ avec $p' \leq p$ et chaque \mathbb{X}^m peut être à nouveau décomposé en sous-régions...
- **A chaque nœud** m est associé une région $\mathbb{X}^m \subset \mathbb{X}$ et une **fonction de décision** dénotée f^m qui prend en entrée un élément $\mathbf{x} \in \mathbb{X}^m$ et qui donne en sortie un sous-espace $\mathbb{X}^{m'} \subset \mathbb{X}^m$.
- Les arbres décisionnels sont considérés comme des **méthodes non paramétriques** dans la mesure où :
 - ▶ Aucune hypothèse sur la distribution de probabilités des classes.
 - ▶ La structure de l'arbre n'est pas donnée à l'avance : on ajoute nœuds, arcs et feuilles, en fonction des données à l'étude.

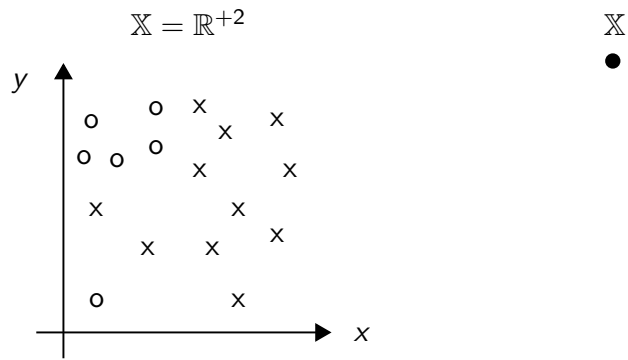
Rappel du Sommaire

- 1 Introduction
- 2 Les méthodes linéaires et leurs pénalisations (ridge, lasso, ...)
- 3 Les machines à vecteurs supports ("Support Vector Machines")
- 4 Les arbres de décisions ("Decision Trees")
- 5 Décider en comité ("Ensemble Learning")

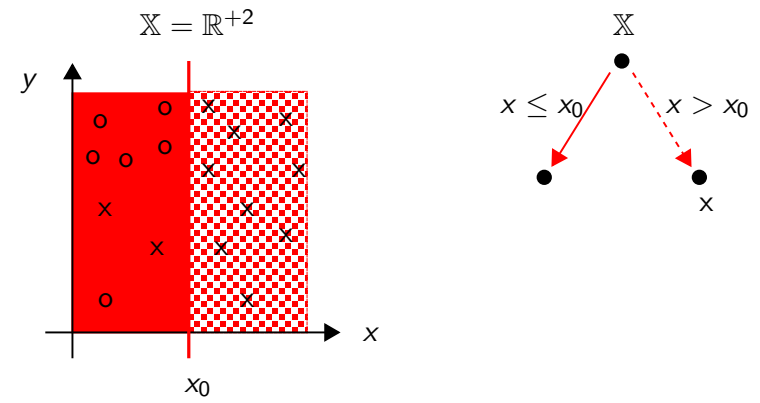
Introduction (suite)

- f^m , la fonction de discrimination du nœud m est une **fonction simple**. Mais, l'ensemble des fonctions f^m de chaque nœud de l'arbre tout entier aboutit à une **fonction de décision complexe**.
- Les méthodes de cette famille se distinguent selon :
 - ▶ Le type de fonction f^m choisi pour discriminer un ensemble de points.
 - ▶ Le type de critère permettant d'évaluer la qualité d'une fonction de discrimination.
- **A chaque feuille** de l'arbre est associée un **élément de \mathbb{Y}** :
 - ▶ Pour un problème de catégorisation il s'agit donc d'une classe.
 - ▶ Pour un problème de régression il s'agit donc d'un réel.
- Chaque feuille correspond à une région de \mathbb{X} et tout \mathbf{x} appartenant à une même feuille a le même élément de \mathbb{Y} associé à la feuille.
- Comme pour les svm nous traiterons d'abord le problème de catégorisation puis celui de régression.

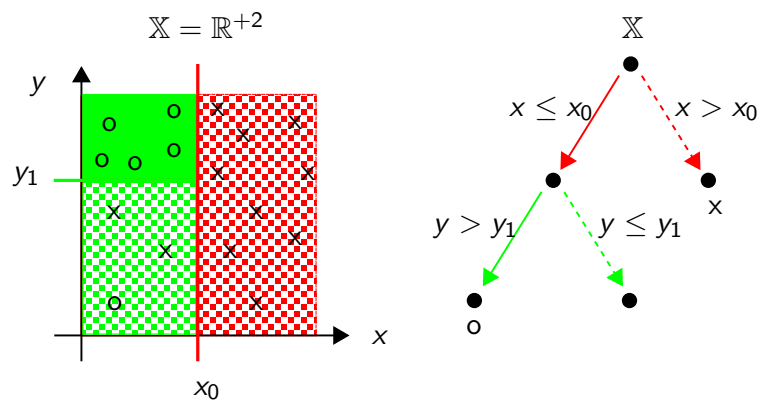
Exemple



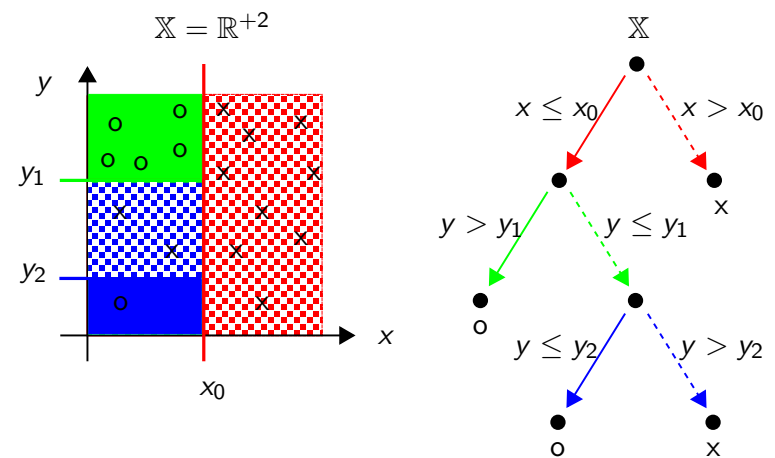
Exemple (suite)



Exemple (suite)



Exemple (suite)



Arbre de décision (ad) pour la catégorisation

- On considère $\mathbb{Y} = \{C_1, \dots, C_q\}$ comme un ensemble discret. On parle alors d'**arbre de classification**.
- Par contre \mathbb{X} peut être hétérogène (càd mélange de variables continues et discrètes).
- Nous traiterons essentiellement des méthodes **univariées** càd à chaque nœud m on utilise une seule variable $X^j \in \mathbb{A}$ pour définir f^m .
- Si X^j est **discrète** avec q_j **catégories** $\{X^{j,1}, \dots, X^{j,q_j}\}$ alors :

$$\forall \mathbf{x} \in \mathbb{X}, f^m(\mathbf{x}) \in \{X^{j,1}, \dots, X^{j,q_j}\}$$

Il s'agit dans ce cas d'une séparation ou division en q_j régions.

- Si X^j est **continue** alors :

$$\forall \mathbf{x} \in \mathbb{X}, f^m(\mathbf{x}) \in \{X^{j,l}, X^{j,r}\}$$

où $X^{j,l} = \{\mathbf{x} \in \mathbb{X} : x_j \leq \delta_j\}$ et $X^{j,r} = \{\mathbf{x} \in \mathbb{X} : x_j > \delta_j\}$ et $\delta_j \in \mathbb{R}$ est une valeur permettant de faire une séparation adéquate. Il s'agit dans ce cas d'une **division en 2 régions** (séparation binaire de l'espace).

Arbre de décision (ad) pour la catégorisation (suite)

- La mesure de qualité d'une division en plusieurs branches est relative au concept d'**impureté**.
- Une séparation f^m est pure si chacune des branches conduit à des nœuds dont les éléments sont tous de la même catégorie.
- Dénotons par \mathbf{N}^m le nombre d'éléments du nœud m . Pour la racine on a donc $\mathbf{N}^m = n$.
- Parmi les éléments du nœud m , dénotons par \mathbf{N}_l^m le nombre de ceux appartenant à la classe C_l . On a alors :

$$P(C_l|X, m) = \frac{\mathbf{N}_l^m}{\mathbf{N}^m}$$

- Le nœud m est pur** si $\exists C_l \in \mathbb{Y} : P(C_l|X, m) = 1$. Ainsi, f^m est **pure** si pour tous les nœuds qu'elle engendre, ceux-ci sont purs.
- Pour alléger les formules nous utiliserons la notation suivante :

$$P(C_l|X, m) = p_l^m$$

Arbre de décision (ad) pour la catégorisation (suite)

- L'apprentissage ("tree induction") consiste à construire un ad étant donné \mathbb{E} .
- Il existe de très nombreux arbres permettant de découper l'espace \mathbb{X} de sorte à n'avoir aucune erreur.
- Si on applique le principe du rasoir d'Occam, on cherche l'**ad d'erreur nulle qui est le plus petit** en termes de nombre de nœuds.
- Ce **problème est NP-complet**, on a donc recourt à des **heuristiques** :
 - On part de \mathbb{X} tout entier : ce nœud représente la racine.
 - Pour chaque nœud m on détermine la fonction f^m permettant d'optimiser localement un critère.
 - La fonction f^m permet alors de séparer l'espace en plusieurs régions (arcs) et chacune d'entre elles représente un nouveau nœud.
 - On ajoute nœuds et arcs jusqu'à satisfaire un critère d'arrêt.

Arbre de décision (ad) pour la catégorisation (suite)

- Pour mesurer la pureté d'une séparation, nous utiliserons la méthode classique proposée par [Quinlan, 1986] qui est basée sur l'**entropie** :

$$\begin{aligned} ent(p^m) &= - \sum_{l=1}^q P(C_l|X, m) \log_2(P(C_l|X, m)) \\ &= - \sum_{l=1}^q p_l^m \log_2(p_l^m) \end{aligned}$$

où par convention $0 \log_2(0) = 0$.

- L'entropie correspond intuitivement à la quantité d'information contenue ou délivrée par une source d'information.
- Dans le cas binaire, si $p_1^m = 1$ et $p_2^m = 0$: il faut 0 bit pour transmettre l'information.
- Si $p_1^m = p_2^m = 1/2$ alors la quantité d'information est maximale : il faut 1 bit (1 pour la classe C_1 et 0 pour la classe C_2) pour transmettre l'information.

Arbre de décision (ad) pour la catégorisation (suite)

- D'autres mesures h permettant d'évaluer l'impureté d'une division existent. Dans le cas binaire $q = 2$, ces critères doivent vérifier :
 - ▶ $\forall p \in [0, 1] : h(1/2, 1/2) \geq h(p)h(1-p)$.
 - ▶ $h(0, 1) = h(1, 0) = 0$.
 - ▶ $h(p, 1-p)$ est \nearrow par rapport à p sur $[0, 1/2]$ et \searrow sur $[1/2, 1]$.
- Des exemples classiques sont donc l'entropie (*ent*), l'indice de Gini (*gini*) et l'erreur de classification (*cerr*) :

$$ent(p) = -p \log_2(p) - (1-p) \log_2(1-p)$$

$$gini(p) = 2p(1-p)$$

$$cerr(p) = 1 - \max(p, 1-p)$$

- *cerr* ne se comporte pas toujours correctement. *ent* et *gini* donnent de meilleurs résultats mais leurs différences ne sont pas statistiquement significatives.

Arbre de décision (ad) pour la catégorisation (suite)

- Pour alléger les formules notons :

$$P(C_l | X, m, X^{j,k}) = p_{kl}^m$$

- L'**impureté totale** issue de la division engendrée par X^j est :

$$ent(p^m, X^j) = - \sum_{k=1}^{q_j} \frac{\mathbf{N}_k^m}{\mathbf{N}^m} \sum_{l=1}^q p_{kl}^m \log_2(p_{kl}^m)$$

- **A chaque nœud on détermine X^j qui minimise $ent(p^m, X^j)$.**
- Si X^j est **qualitative**, les séparations sont données par les différentes catégories $\{X^{j,1}, \dots, X^{j,q_j}\}$.
- Si X^j est **quantitative**, il faut en plus déterminer δ_j donnant la meilleure division $\{X^{j,l}, X^{j,r}\}$. A la base il y a $\mathbf{N}^m - 1$ possibilités. Le meilleur point de division est toujours entre deux objets adjacents de classes distinctes.

Arbre de décision (ad) pour la catégorisation (suite)

- Si un nœud m n'est pas pur alors l'objectif est de séparer le sous-ensemble de ce nœud de sorte à réduire les impuretés.
- Comme on cherche l'ad le plus petit, intuitivement, **on choisit localement la séparation qui réduit le plus l'impureté.**
- Supposons qu'au nœud m , il y a \mathbf{N}_k^m objets qui suivent la branche k . Il s'agit des objets \mathbf{x} tel que $f^m(\mathbf{x}) = X^{j,k}$ où X^j est la variable ayant servi à faire la séparation.
- Supposons que le nombre de branches est q_j ($q_j = 2$ si X^j est quantitative) alors nous avons $\sum_{k=1}^{q_j} \mathbf{N}_k^m = \mathbf{N}^m$.
- Considérons la variable cible Y et soit \mathbf{N}_{kl}^m le nombre d'objets de C_l ayant suivis la branche donnée par $X^{j,k}$. La probabilité d'observer la classe C_l dans le nœud issu de la branche $X^{j,k}$ vaut :

$$P(C_l | X, m, X^{j,k}) = \frac{\mathbf{N}_{kl}^m}{\mathbf{N}_k^m}$$

Arbre de décision (ad) pour la catégorisation (suite)

- **L'ad se construit de façon récursive** : à chaque nœud on cherche localement la variable X^j minimisant l'impureté d'une nouvelle division et ce jusqu'à ce que l'on obtienne une séparation pure.
- Il existe un biais à cette approche : les variables qualitatives ayant beaucoup de catégories donnent une plus faible entropie.
 - ▶ Nous pouvons alors décider de nous restreindre à des ad binaires c'est-à-dire que chaque division est composée de deux branches. Mais dans le cas d'une variable qualitative à q_j catégories, il existe $2^{q_j-1} - 1$ possibilités et dans le cas général, si q_j est grand le problème devient exponentiel.
 - ▶ En revanche, pour un problème de catégorisation binaire ($\{C_1, C_2\}$), on peut ordonner les catégories de X^j dans l'ordre décroissant de p_{k1}^m et traiter ensuite cet ordre telle une variable ordonnée avec cette fois-ci uniquement $q_j - 1$ possibilités de séparation.
 - ▶ Dans ce cas on préférera un ad binaire puisque celui-ci peut retrouver l'ad avec plusieurs branches si ce cas était le meilleur.

Arbre de décision (ad) pour la catégorisation (suite)

- Un autre problème survient si le critère d'arrêt est l'obtention de **feuilles toutes pures** (càd on s'arrête une fois que tous les nœud terminaux obtenus n'ont qu'une seule classe représentée). Dans ce cas, on risque (i) d'avoir un **ad trop grand** et (ii) de faire du **sur-apprentissage**.
- Pour remédier à ce problème, on se donne un **seuil** $\theta \in [0, 1]$ en dessous duquel on estime que la pureté obtenue est suffisante.
- Ainsi la condition d'arrêt de l'apprentissage est que pour tout nœud final m : $ent(m) \leq \theta$.
- Chaque feuille m est alors associée à la classe la plus représentative c'àd la classe C_l tel que $\forall l' \neq l : p_l^m \geq p_{l'}^m$.
- Dans certaines applications, on représente chaque feuille m par la distribution de probabilités (p_1^m, \dots, p_q^m) . Par exemple si on souhaite calculer un risque associé aux catégorisations données par l'ad.

Pseudo-code de l'apprentissage d'un ad (catégorisation)

```

Fonction : ArbreGeneration
Input :  $\mathbf{X}^m, \theta$ 
1 Si  $ent(p^m) \leq \theta$  faire
2     Créer une feuille et l'étiqueter avec la classe majoritaire
3     Retour
4 Fin Si
5  $j^* \leftarrow \text{DivisionAttribut}(\mathbf{X}^m)$ 
6 Initialiser un sous-arbre  $S$ 
7 Pour toute Branche  $m'$  dans  $\{X^{j^*,1}, \dots, X^{j^*,q_j}\}$  faire
8     Déterminer  $\mathbf{X}^{m'}$ 
9      $S' \leftarrow \text{ArbreGeneration}(\mathbf{X}^{m'}, \theta)$ 
10    Ajouter  $S'$  à une branche de  $S$ 
11 Fin Pour

```

Arbre de décision (ad) pour la catégorisation (suite)

- θ peut-être vu comme un **paramètre de la complexité de l'ad** similaire au k du k -ppv dans le contexte des méthodes non paramétriques.
- Si θ est petit, la variance est large alors que l'ad est grand de sorte à reproduire les données d'entraînement de façon précise.
- Si θ est grand, la variance est plus faible et l'arbitrage biais-variance nous indique que le biais risque en revanche d'être plus grand.
- Dans la suite nous utiliserons les notations suivantes :
 - ▶ \mathbf{X} est la matrice initiale des données avec n objets $\{X_1, \dots, X_n\}$ et p attributs $\{X^1, \dots, X^p\}$.
 - ▶ \mathbf{X}^m est la matrice des données relatives au nœud m qui comporte \mathbf{N}^m objets et les p attributs. Il s'agit d'une sous-matrice de \mathbf{X} .
 - ▶ On remarquera qu'un nœud fils comporte un sous-ensemble des objets de son nœud parent.
 - ▶ L'algorithme qui suit synthétise différentes évolutions des ad (CART [Breiman et al., 1984], ID3[Quinlan, 1986], C4.5[Quinlan, 1993]).

Pseudo-code de l'apprentissage d'un ad (catégorisation)

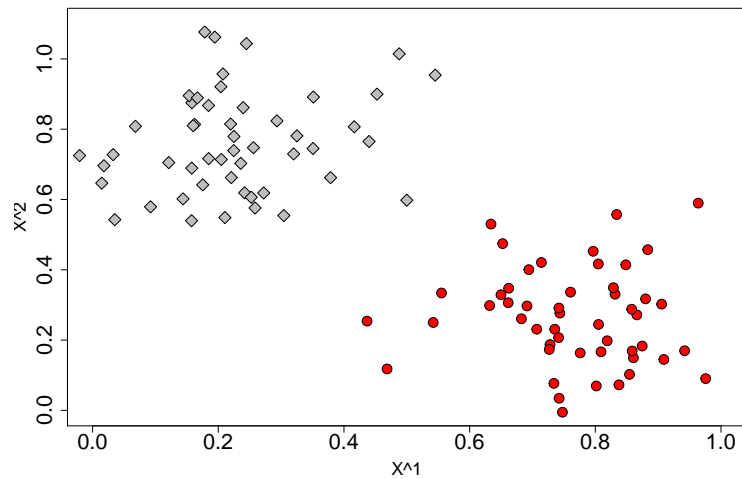
```

Fonction : DivisionAttribut
Input :  $\mathbf{X}^m$ 
1  $MinE \leftarrow +\infty$ 
2 Pour tout Attribut  $X^j$  de  $\{X^1, \dots, X^p\}$  faire
3     Si  $X^j$  est qualitative avec  $q_j$  catégories faire
4          $E \leftarrow ent(p^m, X^j)$ 
5         Si  $E < MinE$  faire  $MinE \leftarrow E, j^* \leftarrow j$  Fin Si
6     Sinon ( $X^j$  est quantitative)
7         Pour toute Séparation en  $\{X^{j,l}, X^{j,r}\}$  possibles faire
8              $E \leftarrow ent(p^m, \{X^{j,l}, X^{j,r}\})$ 
9             Si  $E < MinE$  faire  $MinE \leftarrow E, j^* \leftarrow j$  Fin Si
10        Fin Pour
11    Fin Si
12 Fin Pour
13 Output :  $j^*$ 

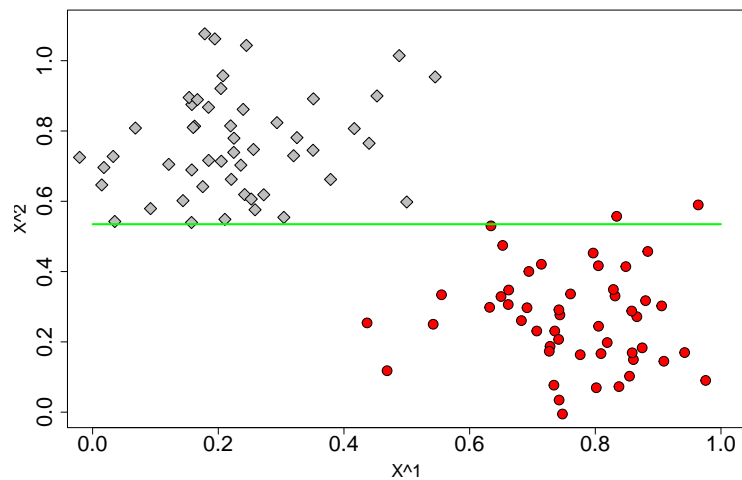
```

ad pour la catégorisation (exemple)

- Reprenons l'exemple précédent :



ad pour la catégorisation (exemple)



ad pour la catégorisation (code R)

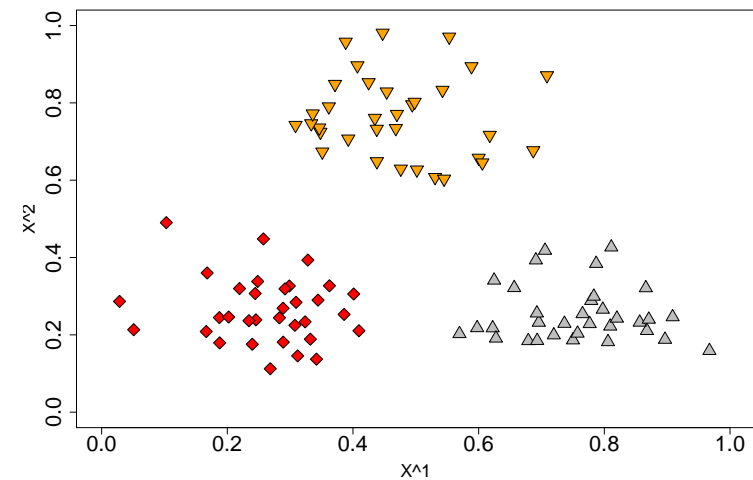
```
> library("rpart")
> c=as.factor(c)
> XX=data.frame(c,X)
> res_dt=rpart(c~.,data=XX)
> res_dt
n= 100

node), split, n, loss, yval, (yprob)
* denotes terminal node

1) root 100 50 -1 (0.50000000 0.50000000)
2) X2< 0.5348561 48 0 -1 (1.00000000 0.00000000) *
3) X2>=0.5348561 52 2 1 (0.03846154 0.96153846) *

> res_dt$control$cp#paramètre par défaut de \theta
[1] 0.01
```

ad pour la catégorisation (autre exemple)



ad pour la catégorisation (code R)

```

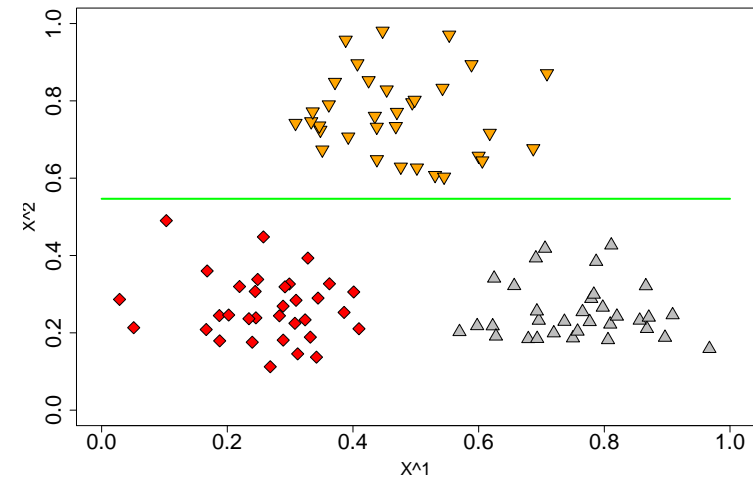
> res_dt=rpart(c~.,data=XX)
> res_dt
n= 99

node), split, n, loss, yval, (yprob)
* denotes terminal node

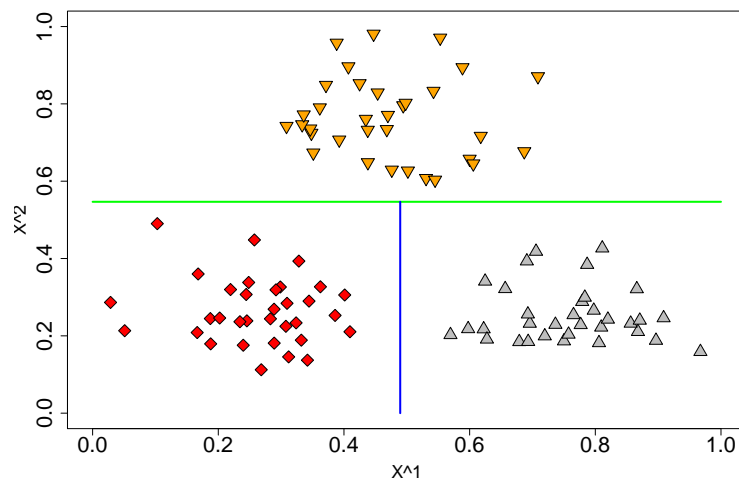
1) root 99 66 1 (0.3333333 0.3333333 0.3333333)
2) X2< 0.5467184 66 33 1 (0.5000000 0.5000000 0.0000000)
4) X1< 0.4895161 33 0 1 (1.0000000 0.0000000 0.0000000) *
5) X1>=0.4895161 33 0 2 (0.0000000 1.0000000 0.0000000) *
3) X2>=0.5467184 33 0 3 (0.0000000 0.0000000 1.0000000) *

```

ad pour la catégorisation (autre exemple)



ad pour la catégorisation (autre exemple)



ad pour la régression

- On considère maintenant $\mathbb{Y} = \mathbb{R}$. On parle alors d'**arbre de régression**.
- \mathbb{X} peut toujours être hétérogène (càd mélange de variables continues et discrètes).
- Ce qui change par rapport aux arbres de classification c'est la fonction d'"impureté" (càd la fonction objectif).
- Soit \mathbf{X}^m la sous-matrice de données de taille $(\mathbf{N}^m \times p)$ relative au nœud m . Par abus de langage on dira qu'il s'agit de l'ensemble des objets qui ont suivi le chemin pour arriver jusqu'à m . Notons alors la fonction indicatrice suivante :

$$ind(\mathbf{x}, m) = \begin{cases} 1 & \text{si } \mathbf{x} \in \mathbf{X}^m \\ 0 & \text{sinon} \end{cases}$$

ad pour la régression (suite)

- Pour mesurer la pureté d'un nœud m on utilise le critère suivant :

$$err(m) = \frac{1}{\mathbf{N}^m} \sum_{i=1}^n (y_i - g^m)^2 ind(\mathbf{x}_i, m)$$

où $\mathbf{N}^m = \sum_{i=1}^n ind(\mathbf{x}_i, m)$ est le nombre d'objets de \mathbf{X}^m .

- g^m est une tendance centrale relative au nœud m : c'est une mesure qui résume les valeurs des objets appartenants à m . On utilise la moyenne (la médiane si les données sont très bruitées) :

$$g^m = \frac{\sum_{i=1}^n y_i ind(\mathbf{x}_i, m)}{\sum_{i=1}^n ind(\mathbf{x}_i, m)} = \frac{1}{\mathbf{N}^m} \sum_{i=1}^n y_i ind(\mathbf{x}_i, m)$$

- Dans ce cas $err(m)$ est une variance locale au nœud m .
- Le critère qui arrête la progression de l'ad est $err(m) \leq \theta$. θ est donc un seuil en-dessous duquel on estime que la variance de la région relative au nœud m est suffisamment basse.

ad pour la régression (suite)

- Pour choisir la variable séparatrice X^j , on prend celle qui minimise :

$$err(m, X^j) = \frac{1}{\mathbf{N}^m} \sum_{k=1}^{q_j} \sum_{i=1}^n (y_i - g_k^m)^2 ind(\mathbf{x}_i, m, X^{j,k})$$

- Le **pseudo-code** donné précédemment dans le cas de la catégorisation s'adapte entièrement au problème de régression en remplaçant :
 - ▶ $ent(p^m)$ par $err(m)$ dans le pseudo-code `ArbreGeneration`.
 - ▶ $ent(p^m, X^j)$ par $err(m, X^j)$ dans le pseudo-code `DivisionAttribut`.
- Le paramètre θ est comme précédemment un paramètre de la complexité de l'ad.

ad pour la régression (suite)

- Il nous faut également spécifier un **critère pour décider de f^m** c-à-d la division à utiliser au nœud m si celui-ci est de variance (ou "impureté") encore trop forte.
- Prenons une variable X^j qui induit une séparation en q_j branches $\{X^{j,1}, \dots, X^{j,q_j}\}$ et introduisons $\forall k = 1, \dots, q_j$:

$$ind(\mathbf{x}, m, X^{j,k}) = \begin{cases} 1 & \text{si } \mathbf{x} \in \mathbf{X}^m \wedge \mathbf{x} \in X^{j,k} \\ 0 & \text{sinon} \end{cases}$$

- Soit g_k^m la tendance centrale des objets de la branche $X^{j,k}$ de m :

$$g_k^m = \frac{\sum_{i=1}^n y_i ind(\mathbf{x}_i, m, X^{j,k})}{\sum_{i=1}^n ind(\mathbf{x}_i, m, X^{j,k})} = \frac{1}{\mathbf{N}_k^m} \sum_{i=1}^n y_i ind(\mathbf{x}_i, m, X^{j,k})$$

Elagage de l'ad

- D'autres critères permettent d'améliorer les performances de l'ad : on parle d'**élagage** de l'ad. Ce process peut s'effectuer au cours de la construction de l'ad (**pré-élagage**) ou après la construction de l'ad (**post-élagage**).
- Exemple de pré-élagage :
 - ▶ Si le pourcentage de données au nœud m est en-dessous d'un seuil α on ne sépare pas le nœud : prendre une décision de division sur trop peu d'éléments augmente la variance du modèle et donc potentiellement des erreurs en généralisation.
- Principe du post-élagage :
 - ▶ On construit l'ad jusqu'à avoir des feuilles complètement pures ($\theta = 0$) sur \mathbb{E} .
 - ▶ On tente de détecter des sous-arbres qui causent du sur-apprentissage et on les enlève de l'ad.
 - ▶ Pour chaque sous-arbre enlevé, on le remplace par une feuille étiquetée avec la classe majoritaire (classification) ou la tendance centrale (régression).

Extraction de règles à partir d'un ad

- Un ad fait de la **sélection de variable** : dans un ad il se peut que certaines variables X^j ne soient pas du tout utilisées.
- Les variables de division proches du nœud racine sont globalement plus importantes.
- Contrairement aux svm³, les ad sont **facilement interprétables**.
- Tout **chemin** du nœud racine à une feuille est une **conjonction de plusieurs tests**.
- Pour l'exemple précédent : **Si** ($x_{i2} < 0.54$) **et** ($x_{i1} < 0.48$) **alors** ($X_i \in C_1$).
- On a donc pour un ad un ensemble de **"IF-THEN" règles** qui permet de faire de l'**extraction de connaissances**.
- Les ad sont en revanche des modèles à **forte variance** : de petits changements dans \mathbb{E} peuvent entraîner de nombreux changements dans l'ad.

3. et autres méthodes comme les réseaux de neurones.

Rappel du Sommaire

- 5 Décider en comité ("Ensemble Learning")
 - Introduction
 - Bagging
 - Les forêts aléatoires ("random forest")
 - Boosting

Rappel du Sommaire

- 1 Introduction
- 2 Les méthodes linéaires et leurs pénalisations (ridge, lasso, ...)
- 3 Les machines à vecteurs supports ("Support Vector Machines")
- 4 Les arbres de décisions ("Decision Trees")
- 5 Décider en comité ("Ensemble Learning")

Beaucoup de méthodes en AA

- Nous avons vu plusieurs types de familles de méthodes d'apprentissage : les méthodes paramétriques, les méthodes non paramétriques. . .
- Parmi les méthodes paramétriques, nous avons vu plusieurs familles d'hypothèses : les modèles linéaires, avec ou sans expansions de base (ou noyaux), les arbres de décisions. . .
- Il existe bien évidemment beaucoup d'autres méthodes !
- Pourquoi ? Parce qu'en général, il n'existe pas un type de modèle qui donne toujours les meilleures performances pour tout type de problèmes d'apprentissage ("No free lunch theorem").
- L'idée générale des **méthodes d'ensembles** est de combiner plusieurs régresseurs/classifieurs afin d'améliorer l'apprentissage.
- Mais pourquoi décider en comité ?

Le théorème du jury de Condorcet

- Un jury doit décider collectivement sur une question dont les réponses sont 0 ou 1.
- Supposons que la bonne réponse soit 1, qu'un votant quelconque a une probabilité p de donner la bonne réponse et que chaque votant est indépendant des autres.
- Le mode de scrutin est le **vote majoritaire**.
- Quel serait le nombre de votants N qu'il faudrait faire participer au jury pour avoir une grande probabilité P que la majorité donne la bonne réponse 1 ?
- Tout dépend de p :
 - ▶ Si $p > 1/2$ alors ajouter des votants dans le jury permet d'augmenter la probabilité P que la décision majoritaire soit 1. De plus, si $p > 1/2$ alors $P \rightarrow 1$ lorsque $N \rightarrow \infty$.
 - ▶ Si $p \leq 1/2$ alors ajouter des votants dans le jury fait décroître P et dans ce cas le jury optimal est composé d'un seul individu ($N = 1$).

"Wisdom of the crowd ! But what crowd ?"

- Le principe sous-jacent au théorème du jury de Condorcet a été vérifié dans d'autres contextes : la réponse collective de plusieurs individus amateurs peut être meilleure que celle d'un seul expert.
- J.M. Surowiecki, Phd et journaliste économiste, a écrit en 2004 un livre célèbre "The Wisdom of Crowds : Why the Many Are Smarter Than the Few and How Collective Wisdom Shapes Business, Economies, Societies and Nations".
- Cependant, pas toute foule est sage : les bulles spéculatives par exemple ("mais pourquoi tu cours ... ben parce que tu cours !").
- Selon Surowiecki, une foule est **sage** si elle vérifie les principes de :
 - ▶ **diversité** : les opinions doivent être diverses ;
 - ▶ **indépendance** : une opinion ne doit pas dépendre d'autres opinions ;
 - ▶ **décentralisation** : une opinion ne doit pas être soumise à une autorité supérieure ;
 - ▶ **agrégation** : les opinions individuelles peuvent être agrégées en une opinion collective.

Jury de Condorcet et méthodes d'ensemble

- Ce résultat peut s'appliquer aux problèmes d'apprentissage automatique et donne l'intuition de base de nombreuses méthodes d'ensemble :
 - ▶ Supposons que pour un problème de catégorisation binaire, nous disposons de plusieurs classifieurs indépendants les uns des autres et que chacun d'entre eux possède un taux d'erreur inférieur à 50% ("weak classifier").
 - ▶ Alors, un vote majoritaire de ces classifieurs donnerait (selon le théorème du jury de Condorcet) un taux d'erreur collectif plus petit que les taux d'erreurs individuels.
- Dans le cas d'un problème de régression, nous pouvons intuitivement transposer ce raisonnement en prenant une **moyenne** (ou une tendance centrale) de plusieurs régresseurs.
- Remarquons qu'il existe cependant deux limites à ce résultat :
 - ▶ L'hypothèse que les votants sont mutuellement indépendants.
 - ▶ Le résultat concerne une décision entre uniquement deux alternatives.

Différents ingrédients d'une méthode d'ensemble

- On peut identifier différents éléments des méthodes d'ensemble (permettant notamment de faire la distinction entre elles) [Rokach, 2010] :
 - ▶ Un ensemble d'entraînement $\mathbb{E} = \{(\mathbf{x}_i, y_i)\}$.
 - ▶ Un (ou plusieurs) modèle d'apprentissage de base permettant d'obtenir des fonctions de prédiction \hat{f} étant donné un ensemble d'entraînement et une (ou plusieurs) méthode d'inférence (le cas échéant).
 - ▶ Un **générateur de diversité** permettant d'obtenir des fonctions de prédictions diverses.
 - ▶ Un mécanisme d'**agrégation** permettant de combiner les résultats donnés par différentes fonctions de prédiction en un seul.
- Ces différents ingrédients font écho aux différents principes cités précédemment.

Génération de fonctions de prédiction diverses

- D'un point de vue générale, il faut disposer de plusieurs fonctions dont les prédictions sont variées car, intuitivement, il n'y a pas d'intérêt à agréger des fonctions prédisant à peu près la même chose !
- Pour s'assurer de la diversité, plusieurs approches existent [Alpaydin, 2010] :
 - ▶ Utiliser différentes familles d'hypothèses.
 - ▶ Utiliser plusieurs hyperparamètres.
 - ▶ Utiliser plusieurs espaces de description.
 - ▶ Utiliser plusieurs ensembles d'entraînement.

Génération de fonctions de prédiction diverses (suite)

- Utiliser **plusieurs ensembles d'entraînement**, par exemple :
 - ▶ Echantillonner avec remplacement et selon une distribution uniforme, plusieurs sous-ensemble d'objets à partir de \mathbb{E} et apprendre une fonction de prédiction sur chacun de ces échantillons. C'est l'idée du bootstrap déjà discuté au slide 93.
 - ▶ Echantillonner itérativement un sous-ensemble d'objets mais selon une distribution sur \mathbb{E} qui change à chaque itération. Il s'agit d'une approche séquentielle et la probabilité de tirer aléatoirement un objet de \mathbb{E} augmente si la prédiction pour cet objet du modèle courant est mauvaise.
- C'est deux dernières approches basées sur l'échantillonnage de sous-ensemble d'entraînement sont les fondements de deux méthodes classiques que sont :
 - ▶ le **Bagging** ("Bootstrap + Aggregating"),
 - ▶ et **AdaBoost** ("Adaptive Boosting").

Génération de fonctions de prédiction diverses (suite)

- Utiliser **différentes familles d'hypothèses**, par exemple :
 - ▶ Utiliser des méthodes paramétriques et non-paramétriques comme les svm (paramétrique) et les k -ppv (non paramétrique).
 - ▶ Utiliser des méthodes paramétriques différentes comme les svm (méthode linéaire) et les classifieurs bayésien naïfs. . .
- Utiliser une méthode mais avec **plusieurs hyperparamètres**, par exemple :
 - ▶ Les svm avec plusieurs noyaux ("**Multiple Kernel Learning**").
 - ▶ Les k -ppv avec plusieurs valeurs k . . .
- Utiliser **plusieurs espaces de description**, par exemple :
 - ▶ Problèmes multi-vues comme la catégorisation du genre d'une vidéo. Dans ce cas, on peut se baser sur les images, le son, la voix . . .
 - ▶ Sous-espace aléatoire ("**random subspace**") : si \mathbb{X} est l'espace de description, alors on utilise une même méthode mais sur plusieurs sous-ensembles des dimensions de \mathbb{X} , choisis aléatoirement.
 - ▶ Notons que l'utilisation des svm avec plusieurs noyaux peut également être cité dans cette rubrique.

Rappel du Sommaire

- 5 Décider en comité ("Ensemble Learning")
 - Introduction
 - Bagging
 - Les forêts aléatoires ("random forest")
 - Boosting

Introduction

- Méthode proposée par Breiman [Breiman, 1996] en 1996.
- Le bagging consiste à :
 - 1 créer plusieurs échantillons bootstrap à partir de \mathbb{E} ,
 - 2 inférer une fonction de prédiction d'un même modèle d'apprentissage de base sur chaque échantillon bootstrap,
 - 3 agréger les prédictions données par chaque fonction :
 - ★ par un **vote majoritaire** si c'est un problème de catégorisation,
 - ★ par une **moyenne** si c'est un problème de régression.
- La méthode d'apprentissage de base utilisée en 2 doit être de **forte variance** : un "petit" changement de l'ensemble d'apprentissage doit générer un "grand" changement dans la fonction de prédiction estimée (sinon manque de **diversité** et le bagging n'apporte pas grand chose).
- Le modèle utilisé est ainsi en général les arbres décisionnels. Mais d'autres techniques reposant sur des familles hypothèses complexes tels que les réseaux de neurones, peuvent être utilisées.

Analyse théorique du bagging en régression (suite)

- Supposons désormais que nous raisonnions avec un couple aléatoire (X, Y) de loi $P(X, Y)$. L'inégalité précédente conduit à la relation :

$$\mathbb{E}_{X,Y} \left(\mathbb{E}_{\mathbb{E}} \left([Y - f_{\mathbb{E}}(X)]^2 \right) \right) \geq \mathbb{E}_{X,Y} \left((Y - \mathbb{E}_{\mathbb{E}}(f_{\mathbb{E}}(X)))^2 \right)$$

- Ceci est équivalent à :

$$\mathbb{E}_{\mathbb{E}} \left(\mathbb{E}_{X,Y} \left([Y - f_{\mathbb{E}}(X)]^2 \right) \right) \geq \mathbb{E}_{X,Y} \left(\underbrace{(Y - \mathbb{E}_{\mathbb{E}}(f_{\mathbb{E}}(X)))^2}_{f_{bag}(X)} \right)$$

- L'espérance de la perte quadratique de f_{bag} est plus petite que l'espérance sous \mathbb{E} de l'espérance de la perte quadratique de $f_{\mathbb{E}}$.
- Ce résultat montre que la **moyenne de plusieurs fonctions de prédiction** apprises sur différents échantillons fait en **moyenne** moins d'erreur qu'une seule fonction de prédiction apprise sur un échantillon.

Analyse théorique du bagging en régression

- Notons $f_{\mathbb{E}}$ une fonction de prédiction inférée d'un ensemble d'entraînement $\mathbb{E} = \{(\mathbf{x}_i, y_i)\}_{i=1,\dots,n}$ dont les éléments sont des réalisations i.i.d. d'une loi jointe inconnue $P(X, Y)$ où $Y \in \mathbb{R}$.
- Dans ce cas, la fonction de prédiction du bagging est :

$$f_{bag}(\mathbf{x}) = \mathbb{E}_{\mathbb{E}}(f_{\mathbb{E}}(\mathbf{x}))$$

- Soit (\mathbf{x}, y) un couple quelconque donné, l'espérance de l'erreur quadratique sous \mathbb{E} de $f_{\mathbb{E}}$ pour ce couple vaut :

$$\mathbb{E}_{\mathbb{E}} \left([y - f_{\mathbb{E}}(\mathbf{x})]^2 \right) = y^2 - 2y\mathbb{E}_{\mathbb{E}}(f_{\mathbb{E}}(\mathbf{x})) + \mathbb{E}_{\mathbb{E}} \left([f_{\mathbb{E}}(\mathbf{x})]^2 \right)$$

Comme pour toute v.a. Z , $\mathbb{E}(Z^2) \geq (\mathbb{E}(Z))^2$, on voit alors que $\mathbb{E}_{\mathbb{E}} \left([f_{\mathbb{E}}(\mathbf{x})]^2 \right) \geq (\mathbb{E}_{\mathbb{E}}(f_{\mathbb{E}}(\mathbf{x})))^2$ et donc :

$$\mathbb{E}_{\mathbb{E}} \left([y - f_{\mathbb{E}}(\mathbf{x})]^2 \right) \geq (y - \mathbb{E}_{\mathbb{E}}(f_{\mathbb{E}}(\mathbf{x})))^2$$

Analyse théorique du bagging en régression (suite)

- Le gain potentiel que l'on peut obtenir avec le bagging dépend de l'écart entre ces deux éléments :

$$\mathbb{E}_{\mathbb{E}} \left([f_{\mathbb{E}}(\mathbf{x})]^2 \right) \geq (\mathbb{E}_{\mathbb{E}}(f_{\mathbb{E}}(\mathbf{x})))^2$$

- Si la variance de $f_{\mathbb{E}}(\mathbf{x})$ est très forte cela veut dire que $\mathbb{E}_{\mathbb{E}} \left([f_{\mathbb{E}}(\mathbf{x})]^2 \right) - (\mathbb{E}_{\mathbb{E}}(f_{\mathbb{E}}(\mathbf{x})))^2$ est grand et donc le gain est important.
- Il est donc préférable d'utiliser une méthode de base de forte variance afin d'espérer observer une amélioration forte due au bagging.
- On remarquera que si la variance est nulle alors $\mathbb{E}_{\mathbb{E}} \left([f_{\mathbb{E}}(\mathbf{x})]^2 \right) = (\mathbb{E}_{\mathbb{E}}(f_{\mathbb{E}}(\mathbf{x})))^2$ et l'inégalité du slide précédent devient également une égalité (pas de gain).

Analyse théorique du bagging en catégorisation

- $\mathbb{Y} = \{C_1, \dots, C_q\}$ est désormais discret. On suppose que (X, Y) est un couple aléatoire avec $P(X, Y)$, $P(Y|X)$, $P(X)$ les fonctions de probabilité jointe, conditionnelle et marginale respectivement.
- Supposons que l'on ait estimé sur k échantillons bootstrap $\{\mathbb{E}^j\}_{j=1, \dots, k}$ issus de \mathbb{E} , k classifieurs, $\{f_{\mathbb{E}^j}\}_{j=1, \dots, k}$, d'un même modèle.
- Soit (\mathbf{x}, y) un couple quelconque, notons par $P_{bs}(C_l|\mathbf{x})$ ⁴ la fréquence relative de la classe C_l parmi les k classifieurs :

$$P_{bs}(C_l|\mathbf{x}) = \frac{1}{k} \sum_{j=1}^k \text{ind}(f_{\mathbb{E}^j}(\mathbf{x}) = C_l)$$

où pour rappel, ind est la fonction indicatrice.

- Le **classifieur bagging** (vote majoritaire) est défini par :

$$f_{bag}(\mathbf{x}) = \arg \max_{C_{l'} \in \mathbb{Y}} P_{bs}(C_{l'}|\mathbf{x})$$

4. bs signifie bootstrap

Analyse théorique du bagging en catégorisation (suite)

- Reprenons notre distribution P_{bs} donnée par k échantillons bootstrap.
- Pour un \mathbf{x} donné, la probabilité pour que le classifieur bagging $f_{bag}(\mathbf{x}) = \arg \max_{C_{l'} \in \mathbb{Y}} P_{bs}(C_{l'}|\mathbf{x})$ le catégorise correctement vaut :

$$\sum_{C_l \in \mathbb{Y}} \text{ind}(f_{bag}(\mathbf{x}) = C_l) P(C_l|\mathbf{x})$$

- Ce qui est important ici c'est l'**ordre des classes qui est donné selon** P_{bs} et non pas les valeurs des probabilités elle-mêmes. En effet, si C_l est la vraie classe de \mathbf{x} (donc $C_l = \arg \max P(C_{l'}|\mathbf{x})$) alors avoir $P_{bs}(C_l|\mathbf{x}) = 0.9$ ou $P_{bs}(C_l|\mathbf{x}) = 0.5$ n'est pas important à condition que $C_l = \arg \max P_{bs}(C_{l'}|\mathbf{x})$.

Analyse théorique du bagging en catégorisation (suite)

- Rappelons le classifieur bayésien vu en slide 56 :

$$f^*(\mathbf{x}) = \arg \max_{C_{l'} \in \mathbb{Y}} P(C_{l'}|\mathbf{x})$$

Ce-dernier est optimal dans le cas d'une matrice de perte uniforme ce qui est le cas implicitement ici.

- Dans le cas du classifieur bayésien nous avons alors [Breiman, 1996] :

$$\text{acc}(f^*) = \int_{\mathbf{x} \in \mathbb{X}} \max_{C_{l'} \in \mathbb{Y}} P(C_{l'}|\mathbf{x}) P(\mathbf{x}) d\mathbf{x}$$

$\text{acc}(f^*)$ est interprétée ici comme étant la probabilité que f^* fasse de bonnes classifications.

- Peut-on atteindre ce résultat optimal ?

Analyse théorique du bagging en catégorisation (suite)

- On dit que f_{bag} est "order-correct" pour \mathbf{x} si [Breiman, 1996] :

$$f_{bag}(\mathbf{x}) = \arg \max_{C_{l'} \in \mathbb{Y}} P(C_{l'}|\mathbf{x})$$

- Si f_{bag} est "order-correct" pour \mathbf{x} alors :

$$\sum_{C_l \in \mathbb{Y}} \text{ind}(f_{bag}(\mathbf{x}) = C_l) P(C_l|\mathbf{x}) = \max_{C_{l'} \in \mathbb{Y}} P(C_{l'}|\mathbf{x})$$

- Soit alors \mathbb{C} l'ensemble des \mathbf{x} pour lequel f_{bag} est "order-correct" et notons $\overline{\mathbb{C}} = \mathbb{X} \setminus \mathbb{C}$ son complémentaire. Nous avons alors :

$$\begin{aligned} \text{acc}(f_{bag}) &= \int_{\mathbf{x} \in \mathbb{C}} \max_{C_{l'} \in \mathbb{Y}} P(C_{l'}|\mathbf{x}) P(\mathbf{x}) d\mathbf{x} \\ &+ \int_{\mathbf{x} \in \overline{\mathbb{C}}} \left(\sum_{C_l \in \mathbb{Y}} \text{ind}(f_{bag}(\mathbf{x}) = C_l) P(C_l|\mathbf{x}) \right) P(\mathbf{x}) d\mathbf{x} \end{aligned}$$

Analyse théorique du bagging en catégorisation (suite)

- Dans l'expression précédente, c'est bien le fait d'avoir $\arg \max P_{bs}(Y|X) = \arg \max P(Y|X)$ (f_{bag} "order-correct") qui permet d'atteindre le résultat optimal et non pas nécessairement le fait d'avoir $P_{bs}(Y|X) = P(Y|X)$.
- En effet, si f_{bag} est "order-correct" pour tout $\mathbf{x} \in \mathbb{X}$ alors $acc(f_{bag}) = acc(f^*)$ (mais plus facile à dire qu'à faire!).
- Un bon classifieur bagging est donc celui qui est "order-correct" pour $|\mathbb{C}|$ grand par rapport à $|\mathbb{C}|$.
- Ce résultat permet aussi de montrer que, contrairement au problème de régression, des classifieurs bootstrap $\{\hat{f}_{\mathbb{E}^j}\}_{j=1,\dots,k}$ peu performants donneront un classifieur bagging d'encore moins bonne qualité.
- Par ailleurs, ici aussi, la forte variance de la méthode d'apprentissage de base est requise : si les $\{\hat{f}_{\mathbb{E}^j}\}_{j=1,\dots,k}$ sont quasi-identiques alors \mathbb{C} est stable et pas d'amélioration due au bagging alors que si $\{\hat{f}_{\mathbb{E}^j}\}_{j=1,\dots,k}$ sont variables on aura tendance à augmenter \mathbb{C} .

Bagging et arbres de decision

- Nous venons de voir le bagging et ses différentes propriétés.
- En particulier, il est recommandé d'utiliser un modèle d'apprentissage de base qui soit de forte variance et de faible biais.
- C'est le cas des ad qui sont typiquement appliqués avec le bagging :
 - ▶ Si l'arbre est profond, celui-ci peut capturer les structures complexes des données et avoir un faible biais.
 - ▶ Les ad sont fortement variables (changer les données d'entraînement peut changer drastiquement un ad) et donc le principe de "moyenner" plusieurs arbres sous-jacent au bagging permet de réduire la variance (et donc améliorer en principe l'erreur en généralisation) tout en maintenant un faible biais.
- Les forêts aléatoires sont une extension substantielle du bagging+ad qui a été également proposé par Breiman [Breiman, 2001].
- L'idée principale est de modifier le bagging de sorte à avoir des ad décorrélés. Cette approche fait écho au principe d'**indépendance** exposé précédemment mais non encore traité jusqu'à présent.

Rappel du Sommaire

- 5 Décider en comité ("Ensemble Learning")
 - Introduction
 - Bagging
 - Les forêts aléatoires ("random forest")
 - Boosting

Réduire la variance ... encore et toujours !

- Pour mieux comprendre les fondements des forêts aléatoires, il est utile de rappeler quelques résultats en probabilité.
- Soit Z_1, \dots, Z_n , des v.a. identiquement distribuées de variance σ^2 .
- Soit $\bar{Z} = \frac{1}{n} \sum_{i=1}^n Z_i$.
- La variance de \bar{Z} vaut :

$$\rho\sigma^2 + \frac{1-\rho}{n}\sigma^2$$

où ρ est la corrélation supposée positive entre deux v.a..

- Si les Z_i sont de plus indépendants ($\rho = 0$), alors la variance de \bar{Z} est plus petites et se réduit à :

$$\frac{\sigma^2}{n}$$

Les forêts aléatoires

- Dans le bagging, du fait du tirage aléatoire avec remplacement du bootstrap, les échantillons ne sont pas indépendants. Donc, les fonctions de prédiction apprises sur ces échantillons ne le sont pas non plus! Nous sommes ainsi uniquement dans le contexte "identiquement distribué" et nous voulons aller vers la situation "**indépendant** et i.d."
- L'objectif des forêts aléatoires est donc de réduire la variance du bagging en réduisant la corrélation entre les ad.
- Pour ce faire, l'idée est d'ajouter de l'aléatoire dans l'induction d'un arbre en tirant au hasard un sous-ensemble de variables pour être candidat à la division.

Pseudo-code des forêts aléatoires

Input : \mathbb{E}, θ (seuil de pureté), k (nb d'échantillons bootstrap)

- 1 **Pour tout** $j = 1, \dots, k$ **faire**
- 2 Déterminer un échantillon bootstrap \mathbb{E}^j
- 3 Induire un ad \hat{f}^j à partir de \mathbb{E}^j en appliquant la procédure :
- 4 **Tant que** l'arbre n'est pas globalement pur
- 5 Sélectionner aléatoirement r attributs
- 6 Déterminer la meilleure division parmi ces r variables
- 6 Séparer le nœud selon la division précédente
- 7 **Fin Tant que**
- 8 **Fin Pour**
- 9 **Output** : $\{\hat{f}^j\}_{j=1, \dots, k}$

Les forêts aléatoires (suite)

- Plus spécifiquement : avant chaque division d'un nœud m , on choisit aléatoirement r ($r \leq p$) attributs comme candidats à la division.
- Intuitivement, si r est petit alors les arbres appris sur différents échantillons bootstrap sont de moins en moins corrélés et leur agrégation sera de variance plus petite.
- Les forêts aléatoires utilisent à la fois plusieurs ensembles d'entraînement et plusieurs espaces de description (principe du sous-espace aléatoire ou "**random subspace**").

Prédiction des forêts aléatoires

- A partir des n ad estimés $\{\hat{f}^j\}_{j=1, \dots, k}$ on calcule les prédictions de la façon suivante.
- Soit \mathbf{x} un objet quelconque représenté dans \mathbb{X} .
- S'il s'agit d'un problème de régression :

$$\hat{f}_{fa}(\mathbf{x}) = \frac{1}{k} \sum_{j=1}^k \hat{f}^j(\mathbf{x})$$

- S'il s'agit d'un problème de catégorisation :

$$\hat{f}_{fa}(\mathbf{x}) = \arg \max_{C_{l'} \in \mathbb{Y}} \sum_{j=1}^k \text{ind}(\hat{f}^j(\mathbf{x}) = C_{l'})$$

Rappel du Sommaire

- 5 Décider en comité ("Ensemble Learning")
 - Introduction
 - Bagging
 - Les forêts aléatoires ("random forest")
 - Boosting

AdaBoost

- AdaBoost est une méthode de boosting proposée par Freund et Schapire⁵ en 1996 [Freund et al., 1996].
- C'est un algorithme itératif et à chaque itération, une nouvelle fonction de prédiction est estimée mais de façon à palier aux erreurs des fonctions de prédictions précédentes.
- Comme les forêts aléatoires, l'algorithme repose sur :
 - ▶ un échantillonnage des données de \mathbb{E} ,
 - ▶ un modèle de base simple pour éviter le sur-apprentissage.
- Comme précédemment, les ad sont souvent utilisés avec le boosting.
- Mais contrairement aux forêts aléatoires :
 - ▶ adaboost combine **séquentiellement** les fonctions de prédiction et non *a posteriori* comme pour les forêts aléatoires,
 - ▶ adaboost modifie à chaque itération la distribution de probabilités sur \mathbb{E} pour le rééchantillonnage.

5. Prix Gödel 2003

Boosting

- Vers la fin des années 80, Kearns et Valiant pose le "hypothesis boosting problem" dans le cadre de l'apprentissage PAC : *"this problem asks whether an efficient learning algorithm (in the distribution-free model of Valiant) that outputs a hypothesis whose performance is only slightly better than random guessing implies the existence of an efficient algorithm that outputs a hypothesis of arbitrary accuracy"*.
- Schapire en 90 [Schapire, 1990] répond positivement au problème posé. Les méthodes dites de boosting se développent alors et elles visent à construire à partir de prédicteurs individuels "faible" ("weak learner"), un prédicteur collectif "fort" ("strong learner").
- Un weak learner peut être vu comme un prédicteur faiblement corrélé à la variable cible Y tandis qu'un strong learner est un prédicteur arbitrairement fortement corrélé à Y (càd qu'on peut le construire de façon à produire des prédictions de plus en plus corrélées avec Y).

AdaBoost (suite)

- Les points principaux qui font la spécificité de cette approche sont :
 - ▶ La distribution sur \mathbb{E} est uniforme initialement mais elle est modifiée au cours de l'algorithme. Les objets qui sont plus difficiles à prédire ont une probabilité d'être échantillonné qui augmente ("adaptive boosting").
 - ▶ De façon successive, les prédicteurs apprennent sur des échantillons qui sur-représentent les objets qui ont été mal prédits aux itérations suivantes.
 - ▶ Les prédicteurs ne sont donc pas mutuellement indépendants.
- Il existe plusieurs variantes d'adaboost :
 - ▶ adaboost (l'original) pour un pb de catégorisation binaire.
 - ▶ adaboost.M1 et adaboost.M2 pour un pb de catégorisation multiclassée.
 - ▶ adaboost.R2 pour un pb de régression.
 - ▶ Plusieurs autres variantes répondant à des contextes divers...
- Ci-dessous on présente adaboost pour la catégorisation binaire. On suppose $\mathbb{Y} = \{-1, 1\}$ et le weak learner $f : \mathbb{X} \rightarrow \{-1, 1\}$.

Pseudo-code d'AdaBoost "binaire"

```

1   Input :  $\mathbb{E}, T$  (nb d'itérations),  $f$  (un weak learner)
2    $t = 1$ 
3    $\mathbf{w}^t = (1, \dots, 1)/n$ 
4   Pour tout  $t = 1, \dots, T$  faire
5       Déterminer un échantillon  $\mathbb{E}^t$  selon  $\mathbf{w}^t$ 
6       Induire  $\hat{f}^t$  minimisant  $\sum_{i=1}^n w_i^t \text{ind}(f^t(\mathbf{x}_i) \neq y_i)$  sur  $\mathbb{E}^t$ 
7       Calculer  $\text{err}(\hat{f}^t) = \sum_{i=1}^n w_i^t \text{ind}(\hat{f}^t(\mathbf{x}_i) \neq y_i)$ 
8       Si  $\text{err}(\hat{f}^t) > 1/2$  faire
9            $T = t - 1$ 
10          break
11      Fin Si
12      Calculer  $\alpha_t = \frac{1}{2} \log\left(\frac{1 - \text{err}(\hat{f}^t)}{\text{err}(\hat{f}^t)}\right)$ 
13      Calculer  $w_i^{t+1} = w_i^t \exp(-\alpha_t y_i \hat{f}^t(\mathbf{x}_i)), \forall i = 1, \dots, n$ 
14      Calculer  $z^t = \sum_i w_i^{t+1}$ 
15      Normaliser  $\mathbf{w}^{t+1}$  en calculant  $\mathbf{w}^{t+1} = \mathbf{w}^{t+1}/z^t$ 
16  Fin Pour
17  Output :  $\{\hat{f}^t, \alpha_t\}_{t=1, \dots, T}$ 

```

AdaBoost "binaire" (échantillonnage adaptatif)

- Contrairement au bagging où l'échantillonnage est uniforme et i.i.d. d'une itération à l'autre, dans adaboost, l'échantillonnage est **adaptatif** et dépend des erreurs comises d'un weak learner au suivant.
- Breiman nomme ce type de stratégie par **ARCing** pour "Adaptive Resampling and Combining".
- A l'itération t la probabilité de sélectionner \mathbf{x}_i devient :

$$w_i^{t+1} = \frac{w_i^t \exp(-\alpha_t y_i \hat{f}^t(\mathbf{x}_i))}{z^t}$$

où $z^t = \sum_i w_i^t \exp(-\alpha_t y_i \hat{f}^t(\mathbf{x}_i))$.

- Rééchantillonnage **adaptatif** :

$$\exp(-\alpha_t y_i \hat{f}^t(\mathbf{x}_i)) \begin{cases} > 1 & \text{si } y_i \neq \hat{f}^t(\mathbf{x}_i) \\ < 1 & \text{si } y_i = \hat{f}^t(\mathbf{x}_i) \end{cases}$$

- Si \mathbf{x}_i est mal classifié par \hat{f}^t alors sa probabilité d'être tiré aléatoirement augmente à la prochaine itération.

AdaBoost "binaire" (prédiction)

- La prédiction est un **vote pondéré** des weak learners $\{\hat{f}^t\}_t$:

$$\hat{f}_{ab}(\mathbf{x}) = \text{sign}\left(\sum_{t=1}^T \alpha_t \hat{f}^t(\mathbf{x})\right)$$

- $\alpha_t = \frac{1}{2} \log\left(\frac{1 - \text{err}(\hat{f}^t)}{\text{err}(\hat{f}^t)}\right)$ est la fonction logit appliqué au taux de réussite $1 - \text{err}(\hat{f}^t) \in [0, 1]$ (sur \mathbb{E}^t).
- Ainsi moins \hat{f}^t fait d'erreur, plus grand est son coefficient α_t .
- En lignes 6-10 on stoppe l'induction si $\text{err}(\hat{f}^t) > 1/2$ puisque dans ce cas le learner est moins bon que le classifieur aléatoire (il n'est même plus weak). De plus, si $\text{err}(\hat{f}^t) > 1/2$ alors $\alpha_t < 0$ ce qu'on ne souhaite pas.

AdaBoost "binaire" (weak learner)

- adaboost nécessite un weak learner càd qu'il se base sur une classe d'hypothèses \mathbb{H} à fort biais, dont le taux d'erreur est à peine inférieur à celui d'un classifieur aléatoire.
- Si \mathbb{H} est trop complexe alors $\text{err}(\hat{f}^t)$ est faible et l'échantillonnage suivant basé sur \mathbf{w}^{t+1} contiendra peu d'objets avec de forte probabilité d'être tiré. Les échantillons obtenus représentent alors du bruit (l'apprentissage ne se focalise pas sur des exemples difficiles mais sur des exemples quelconques). C'est pour cela que le classifieur de base doit être faible.
- En pratique, on utilise des arbres de décisions basés sur une (voire deux) variables. \mathbb{H} est donc à forte variance mais également à fort biais. On parle de "decision stump".

AdaBoost "binaire" (borne supérieure de l'erreur empirique)

Théorème.

L'erreur d'entraînement est bornée supérieurement comme suit :

$$\frac{1}{n} \sum_{\mathbf{x}_i \in \mathbb{E}} \text{ind}(\hat{f}_{ab}(\mathbf{x}_i) \neq y_i) \leq \prod_{t=1}^T z^t$$

- adaboost peut-être vue comme la minimisation de la borne supérieure de l'erreur empirique [Schapire and Singer, 1999].
- Les z^t étant positifs on peut minimiser $\prod_{t=1}^T z^t$ en minimisant z^t à chaque itération t . Remarquons que :

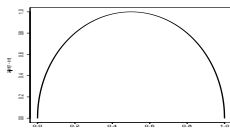
$$z^t = \sum_i w_i^{t+1} = \sum_i w_i^t \exp(-\alpha_t y_i \hat{f}^t(\mathbf{x}_i))$$

AdaBoost "binaire" (optimisation de la borne supérieure)

- Nous pouvons aussi optimiser z^t comme fonction de $\text{err}(\hat{f}^t)$ (erreur pondérée). A l'optimum on sait que $\alpha_t = \frac{1}{2} \log \left(\frac{1 - \text{err}(\hat{f}^t)}{\text{err}(\hat{f}^t)} \right)$. Si on injecte cette relation dans z^t il vient :

$$\begin{aligned} z^t &= \sum_i w_i^t \exp(-\alpha_t y_i \hat{f}^t(\mathbf{x}_i)) \\ &= \sum_{i: \hat{f}^t(\mathbf{x}_i) = y_i} w_i^t \exp(-\alpha_t) + \sum_{i: \hat{f}^t(\mathbf{x}_i) \neq y_i} w_i^t \exp(\alpha_t) \\ &= \exp(-\alpha_t)(1 - \text{err}(\hat{f}^t)) + \exp(\alpha_t) \text{err}(\hat{f}^t) \\ &= 2 \sqrt{\text{err}(\hat{f}^t)(1 - \text{err}(\hat{f}^t))} \end{aligned}$$

- On voit alors qu'il faut choisir à chaque t , \hat{f}^t qui minimise $\text{err}(\hat{f}^t) = \sum_{i=1}^n w_i^t \text{ind}(\hat{f}^t(\mathbf{x}_i) \neq y_i)$.



AdaBoost "binaire" (optimisation de la borne supérieure)

- Nous pouvons optimiser z^t comme fonction de α_t :

$$\begin{aligned} \frac{\partial z^t}{\partial \alpha_t} = 0 &\Leftrightarrow - \sum_i w_i^t y_i \hat{f}^t(\mathbf{x}_i) \exp(-\alpha_t y_i \hat{f}^t(\mathbf{x}_i)) = 0 \\ &\Leftrightarrow - \sum_{i: \hat{f}^t(\mathbf{x}_i) = y_i} w_i^t \exp(-\alpha_t) + \sum_{i: \hat{f}^t(\mathbf{x}_i) \neq y_i} w_i^t \exp(\alpha_t) = 0 \\ &\Leftrightarrow - \exp(-\alpha_t)(1 - \text{err}(\hat{f}^t)) + \exp(\alpha_t) \text{err}(\hat{f}^t) = 0 \\ &\Leftrightarrow \alpha_t = \frac{1}{2} \log \left(\frac{1 - \text{err}(\hat{f}^t)}{\text{err}(\hat{f}^t)} \right) \end{aligned}$$

- Ceci justifie la valeur de α_t donnée dans l'algorithme.

AdaBoost "binaire" (maximisation de la marge)

- La prédiction d'adaboost peut être interprétée tel un vote pondéré :

$$\hat{f}_{ab}(\mathbf{x}) = \sum_{t=1}^T \alpha_t \hat{f}^t(\mathbf{x}) \text{ et } \hat{y} = \text{sign}(\hat{f}_{ab}(\mathbf{x}))$$

- Comme c'est le signe de $\hat{f}(\mathbf{x})$ qui importe on peut, sans perte de généralité, rééchelonner les α_t de sorte à ce que $\sum_t \alpha_t = 1$.
- Bartlett et al [Bartlett et al., 1998] définissent alors la **marge** associée à une observation \mathbf{x}_i comme suit :

$$y_i \hat{f}_{ab}(\mathbf{x}_i) = y_i \sum_{t=1}^T \alpha_t \hat{f}^t(\mathbf{x}_i)$$

où $y_i \hat{f}_{ab}(\mathbf{x}_i) > 0$ si \mathbf{x}_i est bien classifié par \hat{f}_{ab} .

- La valeur $|\sum_{t=1}^T \alpha_t \hat{f}^t(\mathbf{x}_i)|$ est interprétée comme une mesure de la **confiance** en la prédiction donnée par \hat{f}_{ab} : plus $|\sum_{t=1}^T \alpha_t \hat{f}^t(\mathbf{x}_i)|$ est grand plus \hat{f}_{ab} est confiante en sa prédiction.








AdaBoost "binaire" (maximisation de la marge) (suite)

- L'idée de confiance est similaire au concept de marge telle que définie par Vapnik dans le cas des svm.
- Si on se positionne dans un cadre probabiliste, nous avons alors le résultat suivant pour adaboost :








$$P_{X,Y}(Yf_{ab}(X) \leq \theta) \leq 2^{-T} \prod_{t=1}^T \sqrt{\frac{err(f^t)^{1-\theta}(1-err(f^t))^{1+\theta}}{err(f^t)^{1-\theta}(1-err(f^t))^{1+\theta}}}$$

- Ainsi si on minimise $err(f^t)$ à chaque itération t , on maximise la marge. Ceci explique les bonnes performances en **généralisation** d'adaboost.







Quelques références II

-  Breiman, L., Friedman, J. H., Olshen, R. A., and Stone, C. J. (1984). Classification and Regression Trees. Chapman & Hall, New York, NY.
-  Clark, A., Fox, C., and Lappin, S. (2010). The Handbook of Computational Linguistics and Natural Language Processing. Blackwell Handbooks in Linguistics. Wiley.
-  Cleveland, W. S. (2001). Data science : An action plan for expanding the technical areas of the field of statistics.
-  Cornillon, P.-A. and Matzner-Lober, E. (2010). Régression avec R.
-  Cornuéjols, A. and Miclet, L. (2003). Apprentissage artificiel. Eyrolles.
-  Dietterich, T. G. and Bakiri, G. (1995). Solving multiclass learning problems via error-correcting output codes. Journal of artificial intelligence research, 2 :263–286.
-  Dreyfus, G. (2008). Apprentissage Statistique. Réseaux de neurones, Cartes topologiques, Machines à vecteurs supports. Eyrolles.

Quelques références I

-  Abiteboul, S., Bancelhon, F., Bourdoncle, F., Clemencon, S., De La Higuera, C., Saporta, G., and Fogelman Soulié, F. (2014). L'émergence d'une nouvelle filière de formation : " data scientists ". Interne, INRIA Saclay.
-  Alpaydin, E. (2010). Introduction to Machine Learning. MIT Press.
-  Bartlett, P., Freund, Y., Lee, W. S., and Schapire, R. E. (1998). Boosting the margin : A new explanation for the effectiveness of voting methods. The annals of statistics, 26(5) :1651–1686.
-  Beyer, K., Goldstein, J., Ramakrishnan, R., and Shaft, U. (1999). When is Nearest Neighbor Meaningful? In ICDT.
-  Bishop, C. M. (2006). Pattern Recognition and Machine Learning. Springer.
-  Breiman, L. (1996). Bagging predictors. Machine learning, 24(2) :123–140.
-  Breiman, L. (2001). Random forests. Machine learning, 45(1) :5–32.





Quelques références III

-  Efron, B., Hastie, T., Johnstone, I., Tibshirani, R., et al. (2004). Least angle regression. The Annals of statistics, 32(2) :407–499.
-  Freund, Y., Schapire, R., and Abe, N. (1999). A short introduction to boosting. Journal-Japanese Society For Artificial Intelligence, 14(771-780) :1612.
-  Freund, Y. and Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. J. Comput. Syst. Sci., 55(1) :119–139.
-  Freund, Y., Schapire, R. E., et al. (1996). Experiments with a new boosting algorithm. In icml, volume 96, pages 148–156.
-  Friedman, J., Hastie, T., and Tibshirani, R. (2010). Regularization paths for generalized linear models via coordinate descent. Journal of statistical software, 33(1) :1.
-  Friedman, J. H. (1989). Regularized discriminant analysis. Journal of the American statistical association, 84(405) :165–175.
-  Han, J. and Kamber, M. (2006). Data Mining Concepts and Techniques. Morgan Kaufmann.

Quelques références IV

-  Hastie, T., Rosset, S., Tibshirani, R., and Zhu, J. (2004).
The entire regularization path for the support vector machine.
[Journal of Machine Learning Research](#), 5(Oct) :1391–1415.
-  Hastie, T., Tibshirani, R., and Friedman, J. (2011).
[The Elements of Statistical Learning](#).
Springer.
-  James, G., Witten, D., Hastie, T., and Tibshirani, R. (2013).
[An introduction to statistical learning](#), volume 112.
Springer.
-  Jiang, J. (2012).
Information extraction from text.
In [Mining text data](#), pages 11–41. Springer.
-  Leskovec, J., Rajaraman, A., and Ullman, J. D. (2014).
[Mining of massive datasets](#).
Cambridge University Press.
-  Manning, C. D. and Schütze, H. (1999).
[Foundations of Statistical Natural Language Processing](#).
MIT Press.
-  Mitchell, T. (1997).
[Machine Learning](#).
McGraw Hill.

Quelques références VI

-  Vapnik, V. N. (1995).
[The nature of statistical learning theory](#).
Springer-Verlag New York, Inc., New York, NY, USA.
-  Weston, J., Watkins, C., et al. (1999).
Support vector machines for multi-class pattern recognition.
In [ESANN](#), volume 99, pages 219–224.
-  Zhou, G., Zhang, J., Su, J., Shen, D., and Tan, C. (2004).
Recognizing names in biomedical texts : a machine learning approach.
[Bioinformatics](#), 20(7) :1178–1190.
-  Zou, H. and Hastie, T. (2005).
Regularization and variable selection via the elastic net.
[Journal of the Royal Statistical Society : Series B \(Statistical Methodology\)](#), 67(2) :301–320.

Quelques références V

-  Quinlan, J. (1986).
Induction of decision trees.
[Machine Learning](#), 1(1) :81–106.
-  Quinlan, J. R. (1993).
[C4.5 : programs for machine learning](#).
Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.
-  Rokach, L. (2010).
Ensemble-based classifiers.
[Artificial Intelligence Review](#), 33(1-2) :1–39.
-  Schapire, R. E. (1990).
The strength of weak learnability.
[Machine learning](#), 5(2) :197–227.
-  Schapire, R. E. and Singer, Y. (1999).
Improved boosting algorithms using confidence-rated predictions.
[Machine learning](#), 37(3) :297–336.
-  Sun, Y., Deng, H., and Han, J. (2012).
Probabilistic models for text mining.
In [Mining text data](#), pages 259–295. Springer.
-  Valiant, L. G. (1984).
A theory of the learnable.
[Communications of the ACM](#), 27(11) :1134–1142.