Classification supervisée

L. Rouvière
laurent.rouviere@univ-rennes2.fr

Novembre 2019

Présentation

— *Objectifs* : comprendre les aspects formels et pratiques de la classification supervisée.

— *Pré-requis* : théorie des probabilités, modélisation statistique, régression. R, niveau avancé.

— *Enseignant* : Laurent Rouvière *laurent.rouviere@univ-rennes2.fr*
  — Recherche : statistique non paramétrique, apprentissage statistique
  — Enseignements : statistique et probabilités (Université, école d'ingénieur et de commerce, formation continue).
  — Consulting : energie, finance, marketing.

Programme

— 14h *CM* + 6h ou 8h *TP* + 3 ou 4h *évaluation*.
— *Matériel* : slides + Notebook R. Disponible à l'url : `https://lrouviere.github.io/classif_sup/`
— *4 parties* :
  1. Cadre mathématique de la classification supervisée : 4h.
  2. Analyse discriminante linéaire : 4h.
  3. Arbres : 4h.
  4. Introduction aux forêts aléatoires : 2h.
— *Compléments* : exercices (à travailler seul).
— *Pré-requis* : bases de la théorie de l'estimation, modèle linéaire, analyses factorielles (ACP).

Table des matières

Première partie

Le problème de la classification supervisée

1 Quelques exemples

Prévision de pics d'ozone

— On a mesuré pendant 366 jours la *concentration maximale* en ozone (`V4`) ;
— On dispose également d'autres *variables météorologiques* (température, nébulosité, vent...).

```
> Ozone[1:5,]
  V1 V2 V3 V4   V5 V6 V7 V8    V9  V10 V11   V12 V13
1  1  1  4  3 5480  8 20 NA    NA 5000 -15 30.56 200
2  1  2  5  3 5660  6 NA 38    NA   NA -14    NA 300
3  1  3  6  3 5710  4 28 40    NA 2693 -25 47.66 250
4  1  4  7  5 5700  3 37 45    NA  590 -24 55.04 100
5  1  5  1  5 5760  3 51 54 45.32 1450  25 57.02  60
```

Question

Peut-on prédire la concentration maximale en ozone du lendemain à partir des prévisions météorologiques ?

Détection de clients à risque

— Une chaine de magasins a mis en place une carte de crédit.
— Elle dispose d'un *historique de 145 clients* dont 40 ont connu des défauts de paiement.
— Elle connait également d'*autres caractéristiques* sur ses clients (sexe, taux d'endettement, revenus mensuels, dépenses effectuées sur certaines gammes de produit...)

Question

Comment prédire si un nouveau client connaîtra des défauts de paiement ?

Iris de Fisher

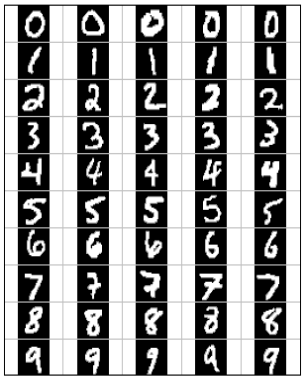
— On a mesuré sur 150 iris de *3 espèces différentes* (Setosa, Versicolor, Virginica) les quantités suivantes :
  — Longueur et largeur des pétales
  — Longueur et largeur des sépales

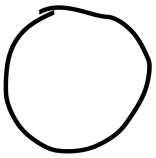
```
> summary(iris)
  Sepal.Length    Sepal.Width     Petal.Length    Petal.Width          Species
 Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100   setosa    :50
 1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300   versicolor:50
 Median :5.800   Median :3.000   Median :4.350   Median :1.300   virginica :50
 Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199
 3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800
 Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500
```

Question

Comment identifier l'espèce d'un iris à partir de ces 4 caractéristiques ?

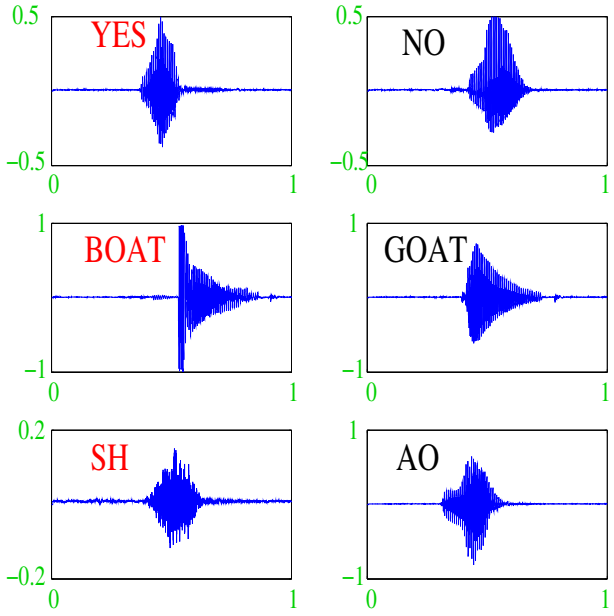
Reconnaissance de l'écriture





Qu'est-ce qui est écrit ? 0, 1, 2... ?

Reconnaissance de la parole



Détection de spam

— Sur 4 601 mails, on a pu identifier 1813 spams.

— On a également mesuré sur chacun de ces mails la présence ou absence de *57* mots.

```
> spam[1:5,c(1:8,58)]
  make address  all num3d  our over remove internet type
1 0.00    0.64 0.64     0 0.32 0.00   0.00     0.00 spam
2 0.21    0.28 0.50     0 0.14 0.28   0.21     0.07 spam
3 0.06    0.00 0.71     0 1.23 0.19   0.19     0.12 spam
4 0.00    0.00 0.00     0 0.63 0.00   0.31     0.63 spam
5 0.00    0.00 0.00     0 0.63 0.00   0.31     0.63 spam
```

Question

Peut-on construire à partir de ces données une méthode de détection automatique de spam ?

Variable à expliquer et variables explicatives

— Les exemples précédents appartiennent à une *même famille de problèmes*.
— Il s'agit d'*expliquer une variable* (notée Y) par p variables (notées $X_1, \dots, X_p$).

Y	X
maxO3	vent, pluie, maxO3v...
bon/mauvais payeur	sexe, revenus...
espèces de l'iris	longueur et largeur des pétales et sépales
spam ou pas spam	présence/absence de certains mots
Chiffre	Images
Mot	Courbes

Des problématiques diverses

— *Apprentissage supervisé* : expliquer/prédire une sortie $y \in \mathcal{Y}$ à partir d'entrées $x \in \mathcal{X}$;

— *Apprentissage non supervisé* : établir une typologie des observations ;

— *Règles d'association* : mesurer le lien entre différents produits ;

— *Systèmes de recommendation* : identifier les produits susceptibles d'intéresser des consommateurs.

Nombreuses applications

finance, économie, marketing, biologie, médecine...

Dans ce cours

On va se focaliser sur le problème d'apprentissage supervisé avec une sortie qualitative.

2 Cadre mathématique

Formalisation mathématique

— Les *données* : $(x_1, y_1) \dots, (x_n, y_n)$ où $x_i \in \mathbb{R}^p$ et $y_i \in \mathcal{Y} = \{1, \dots, K\}$.
— *Modélisation* : ces données sont vues comme des réalisations de variables aléatoires i.i.d $(X_1, Y_1), \dots, (X_n, Y_n)$ de loi inconnue.
— *Objectif* : trouver une fonction $g : \mathbb{R}^p \to \mathcal{Y}$ telle que

$$g(x_i) \approx y_i, \quad \forall i = 1, \dots, n.$$

Définition

On appelle *règle de classification* toute fonction $g : \mathbb{R}^p \to \mathcal{Y}$ qui, à une entrée $x \in \mathbb{R}^p$, renvoie une prévision $g(x) \in \mathcal{Y}$.

2.1 L'erreur de classification

Règle optimale

— Il existe un grand nombre de façons de construire des règles g.
— Nécessité de se donner des *critères de performance*.

Définition

Etant donnée une règle de classification $g : \mathbb{R}^p \to \mathcal{Y}$, on appelle *probabilité d'erreur* ou erreur de classification de g le réel

$$L(g) = \mathbf{E}[\mathbf{1}_{g(X)\neq Y}] = \mathbf{P}(g(X) \neq Y).$$

Objectif

Pour ce critère de performance, le problème sera donc de construire une règle telle que sa *probabilité d'erreur* soit la plus petite possible.

Règle de Bayes

— Problème facile d'un point de vue *théorique*...

Théorème

La règle de Bayes $g^\star : \mathbb{R}^p \to \mathcal{Y}$ définie par

$$g^\star(x) = \underset{k\in\mathcal{Y}}{\operatorname{argmax}}\, \mathbf{P}(Y = k|X = x)$$

est optimale au sens où $L(g^\star) \leq L(g)$ pour toute règle g.

Remarque

Cette règle est *naturelle* : elle consiste à affecter un nouvel individu dans le groupe k qui maximise $\mathbf{P}(Y = k|X = x)$.

2.2 La courbe ROC

Fonction de score

— On se place ici dans un cadre de *classification binaire* ($\mathcal{Y} = \{-1, 1\}$).

— Mais... plutôt que de chercher une règle de prévision $g : \mathcal{X} \to \{-1, 1\}$, on *cherche une fonction* $S : \mathcal{X} \to \mathbb{R}$ telle que

— Une telle fonction est appelée fonction de score : plutôt que de prédire directement le groupe d'un nouvel individu $x \in \mathcal{X}$, on lui donne une *note* $S(x)$

- élevée si il a des "chances" d'être dans le groupe 1 ;
- faible si il a des "chances" d'être dans le groupe -1 ;

Courbe ROC et AUC

— On utilise souvent la *courbe ROC* pour visualiser la performance d'un score :

$$\begin{cases} x(s) = \alpha(s) = 1 - sp(s) = \mathbf{P}(S(X) > s|Y = -1) \\ y(s) = 1 - \beta(s) = se(s) = \mathbf{P}(S(X) \geq s|Y = 1) \end{cases}$$

— On déduit de ce critère un *risque* pour les scores en considérant l'*aire sous la courbe ROC (AUC)* :

$$\mathcal{R}(S) = \text{AUC}(S).$$

Propriété

— $0.5 \leq \text{AUC}(S) \leq 1$.

— Plus l'AUC est *grand*, *meilleur* est le score.

Score parfait et score aléatoire

Remarque

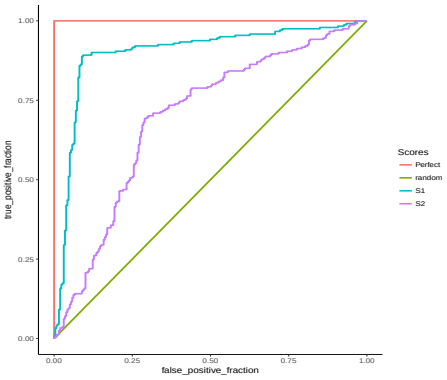
Pour n'importe quel score S on a $x(-\infty) = y(-\infty) = 1$ et $x(+\infty) = y(+\infty) = 0$. $\Longrightarrow$ la courbe ROC vit dans le carré $[0,1]^2$.

— Un *score parfait* va vérifier $\alpha(s^\star) = \beta(s^\star) = 0$ pour une certaine valeur $s^\star$ $\Longrightarrow$ sa courbe ROC passe donc par le point $(0,1)$.

— Un *score aléatoire* (le pire score) est un score qui note indépendamment de Y $\Longrightarrow$ il vérifie donc $x(s) = y(s)$ pour tout s $\Longrightarrow$ sa courbe ROC est donc la première bissectrice.

AUC

AUC(score parfait)=1 et AUC(score aléatoire)=0.5.




```
> library(pROC)
> df1 %>% group_by(Scores) %>% summarize(auc(D,M))
## # A tibble: 4 x 2
##   Scores  `auc(D, M)`
##   <chr>         <dbl>
## 1 Perfect       1
## 2 random        0.5
## 3 S1            0.896
## 4 S2            0.699
```

— D'un point de vue *théorique*, la règle de Bayes et le score optimal se déduisent des probabilités $\mathbf{P}(Y = k|X = x)$.

— Problème plus *difficile* d'un point de vue *pratique*...

Travail statistique

— Les probabilités $\mathbf{P}(Y = k|X = x)$ sont inconnues.

— Le job du statisticien sera de

1. Estimer les probabilités $\mathbf{P}(Y = k|X = x)$ à l'aide de l'échantillon $\mathcal{D}_n = (X_1, Y_1), \ldots, (X_n, Y_n)$.
2. En déduire une règle de classification $\hat{g}_n(.) = \hat{g}_n(., \mathcal{D}_n)$ telle que $L(\hat{g}_n) \approx L(g^\star)$.

Exemple

Si pour une *nouvelle valeur* x, on a

$$\widehat{\mathbf{P}}(Y = 1|X = x) = 0.2,\ \widehat{\mathbf{P}}(Y = 2|X = x) = 0.35,\ \widehat{\mathbf{P}}(Y = 3|X = x) = 0.45$$

alors on *prédira* $\widehat{Y} = \hat{g}(x) = 3$.

2.3 Un exemple : l'algorithme des plus proches voisins

Un exemple : la règle des plus proches voisins

— Etant donné un entier $k \leq n$, elle consiste à affecter un nouvel individu x dans le *groupe majoritaire de ses plus proches voisins* :

$$\hat{g}_n(x) = \underset{k \in \mathcal{Y}}{\operatorname{argmax}} \sum_{i \in \text{kppv}(x)} \mathbf{1}_{Y_i = k}$$

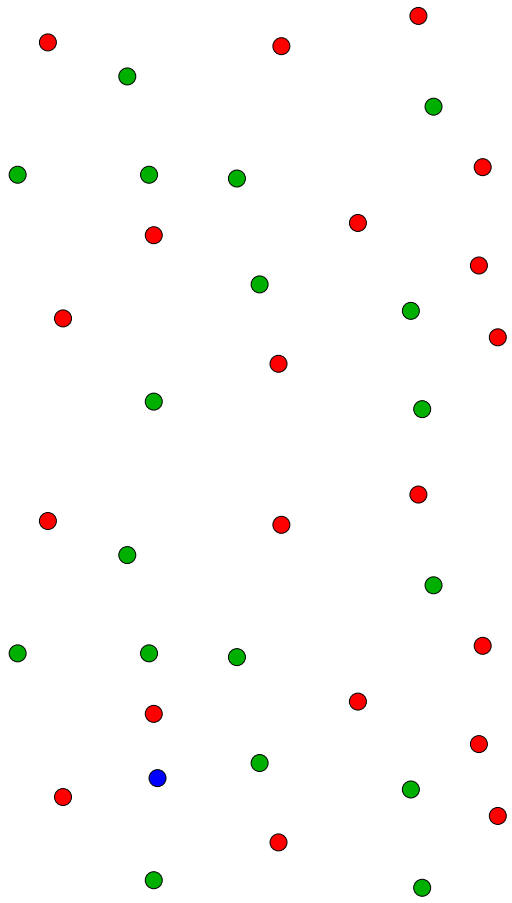
où $\text{kppv}(x) = \{i : X_i \text{ fait partie des kppv de } x \text{ parmi } \{X_1, \ldots, X_n\}\}$.

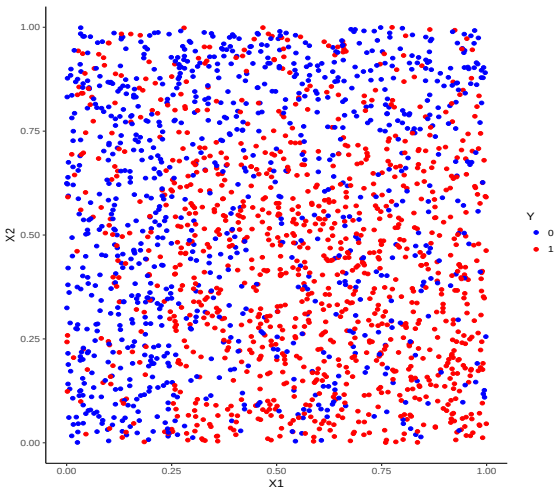
Remarque importante

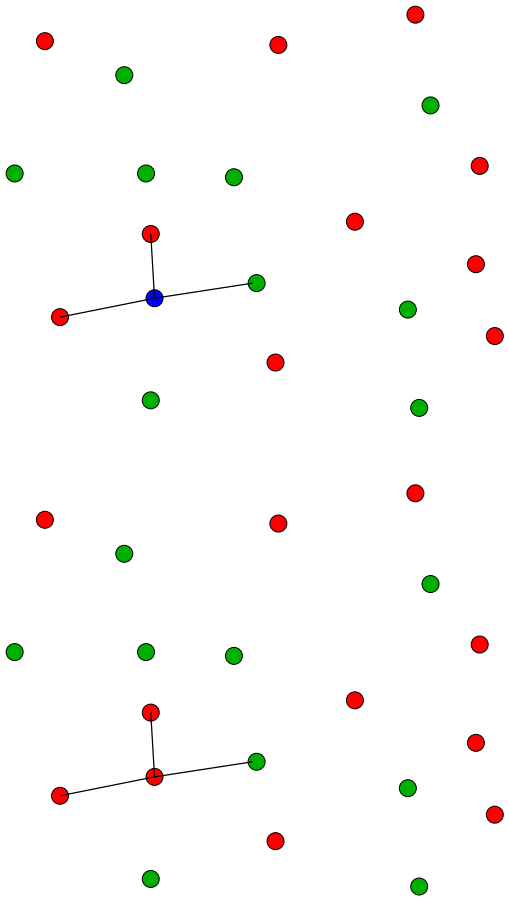
Le paramètre k est crucial pour la qualité de l'estimation :

1. k grand : estimateur « constant », variance faible, biais fort ;
2. k petit : « sur-ajustement », variance forte, biais faible.

Exemple : règle des 3-ppv







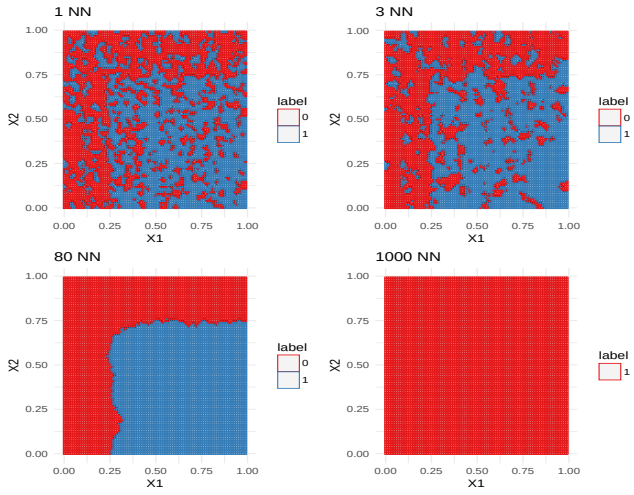
Un exemple

— On cherche à expliquer une variable Y *binaire* par 2 variables X_1 et X_2 quantitatives. On dispose de $n = 2000$ observations.

Représentation des règles des $k$ppv

Conclusion

On visualise bien l'importance du choix de k (parce qu'on est en 2d...)



3 Estimation de l'erreur

Rappels

— n observations $(X_1, Y_1), \ldots, (X_n, Y_n)$ i.i.d à valeurs dans $\mathbb{R}^p \times \mathcal{Y}$.

Objectif

On cherche une règle de prévision $g_n(x) = g_n(x, \mathcal{D}_n)$ qui soit "proche" de l'oracle $g^\star$ défini par

$$g^\star \in \operatorname*{argmin}_g L(g)$$

où $L(g) = \mathbf{P}(g(X) \neq Y)$.

Question

Etant donné un algorithme g_n, *que vaut son erreur*

$$L(g_n) = \mathbf{P}(g_n(X) \neq Y) \ ?$$

Risque empirique

— La loi de (X, Y) étant *inconnue* en pratique, il est *impossible de calculer* $L(g_n) = \mathbf{P}(g_n(X) \neq Y) = \mathbf{E}[\mathbf{1}_{g_n(X) \neq Y}]$.

— **Première approche** : $L(g_n)$ étant une espérance, on peut l'estimer (LGN) par sa *version empirique*

$$L_n(g_n) = \frac{1}{n} \sum_{i=1}^{n} \mathbf{1}_{g_n(X_i) \neq Y_i}.$$

Problème

— L'échantillon $\mathcal{D}_n$ a *déjà été utilisé* pour construire l'algorithme de prévision $g_n \Longrightarrow$ La LGN ne peut donc s'appliquer !

— *Conséquence :* $L_n(g_n)$ conduit souvent à une sous-estimation de $L(g_n)$.

Une solution

Utiliser des méthodes de type validation croisée ou bootstrap.

Apprentissage - Validation ou validation hold out

— L'approche consiste à séparer l'échantillon $\mathcal{D}_n$ en :

1. un *échantillon d'apprentissage* $\mathcal{D}_{n,app}$ pour construire la règle g_n ;
2. un *échantillon de validation* ou *test* $\mathcal{D}_{n,test}$ pour estimer le risque de g_n.

Algorithme

Entrées. $\mathcal{D}_n$: données, $\{\mathcal{A}, \mathcal{V}\}$: partition de $\{1, \ldots, n\}$.

1. Construire l'algorithme de prédiction sur $\mathcal{D}_{n,app} = \{(X_i, Y_i) : i \in \mathcal{A}\}$, on le note $g_{n,app}$;
2. Calculer $\widehat{L}_n(g_{n,app}) = \frac{1}{|\mathcal{V}|} \sum_{i \in \mathcal{V}} \mathbf{1}_{g_{n,app}(X_i) \neq Y_i}$.

Commentaires

Nécessite d'avoir un nombre suffisant d'observations dans

1. $\mathcal{D}_{n,app}$ pour bien *ajuster l'algorithme de prévision* ;
2. $\mathcal{D}_{n,test}$ pour bien *estimer l'erreur de l'algorithme.*

Validation croisée K-blocs

— **Principe** : répéter l'algorithme apprentissage/validation sur *différentes partitions.*

Algorithme - CV

Entrées. $\mathcal{D}_n$: données, K un entier qui divise n ;

1. Construire une partition $\{\mathcal{I}_1, \ldots, \mathcal{I}_K\}$ de $\{1, \ldots, n\}$;
2. Pour $k = 1, \ldots, K$
   (a) $\mathcal{I}_{app} = \{1, \ldots, n\} \backslash \mathcal{I}_k$ et $\mathcal{I}_{test} = \mathcal{I}_k$;
   (b) Construire l'algorithme de prédiction sur $\mathcal{D}_{n,app} = \{(X_i, Y_i) : i \in \mathcal{I}_{app}\}$, on le note $g_{n,k}$;
   (c) En déduire $g_n(X_i) = g_{n,k}(X_i)$ pour $i \in \mathcal{I}_{test}$;

Retourner

$$\widehat{L}_n(g_n) = \frac{1}{n} \sum_{i=1}^{n} \mathbf{1}_{g_n(X_i) \neq Y_i}.$$

Commentaires

— Plus adapté que la technique apprentissage/validation lorsqu'on a *peu d'observations.*
— Le *choix de* K doit être fait par l'utilisateur (souvent $K = 10$).

Leave one out

— Lorsque $K = n$, on parle de validation croisée *leave one out* ;
— Le risque est alors estimé par

$$\widehat{L}_n(g_n) = \frac{1}{n} \sum_{i=1}^{n} \mathbf{1}_{g_n^i(X_i) \neq Y_i}$$

où g_n^i désigne l'algorithme de prévision construit sur $\mathcal{D}_n$ *amputé de la* i*-ème observation.*

Exemple

— On estime les probabilités d'erreur pour les règles de 1, 10, 20 et 95 ppv.

```
> dim(donnees)
[1] 500   3
> head(donnees)
         X1        X2 Y
1 0.1328209 0.3843303 0
2 0.5311487 0.4381712 0
3 0.9785594 0.4039880 1
4 0.5988594 0.3720228 1
5 0.3109124 0.4178881 0
6 0.5964184 0.1043874 1
```

1. On sépare les données en 2
```
> set.seed(1234)
> ind.app <- sample(500,300)
> dapp <- donnees[ind.app,]
> dtest <- donnees[-ind.app,]
```

2. On ajuste les 4 modèles sur les données d'*apprentissage uniquement* et on calcule les prévisions pour les données test.
```
> library(class)
> m1 <- knn(train=dapp[,1:2],test=dtest[,1:2],cl=dapp$Y,k=1)
> m10 <- knn(train=dapp[,1:2],test=dtest[,1:2],cl=dapp$Y,k=10)
> m20 <- knn(train=dapp[,1:2],test=dtest[,1:2],cl=dapp$Y,k=20)
> m95 <- knn(train=dapp[,1:2],test=dtest[,1:2],cl=dapp$Y,k=95)
```

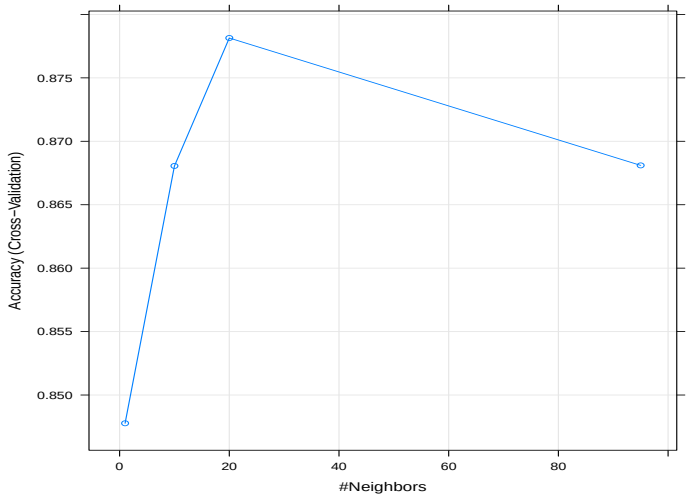
3. On compare les *prévisions* aux *valeurs observées* pour en déduire les estimations de la probabilité d'erreur :
```
> mean(m1!=dtest$Y)
[1] 0.155
> mean(m10!=dtest$Y)
[1] 0.12
> mean(m20!=dtest$Y)
[1] 0.135
> mean(m95!=dtest$Y)
[1] 0.16
```

Package Caret

— Il existe des packages (tels que *caret*) dédiés à l'estimation de critères d'erreur (et/ou au calibrage de paramètres) :
```
> library(caret)
> ctrl <- trainControl(method="cv")
> gr <- data.frame(k=c(1,10,20,95))
> a <- train(Y~.,data=donnees,method="knn",tuneGrid=gr,trControl=ctrl)
> a
k-Nearest Neighbors
500 samples
  2 predictor
  2 classes: '0', '1'
No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 450, 449, 451, 449, 451, 450, ...
Resampling results across tuning parameters:
  k   Accuracy   Kappa
   1  0.8477719  0.6901196
  10  0.8680576  0.7329527
  20  0.8781425  0.7542075
  95  0.8681000  0.7300775

Accuracy was used to select the optimal model using  the largest value.
The final value used for the model was k = 20.
```

Package Caret

```
> plot(a)
```



4 Le sur-apprentissage

— La plupart des algorithmes de prévision *dépendent d'un paramètre* λ.

— Ce paramètre représente souvent une mesure de la *complexité du modèle*.

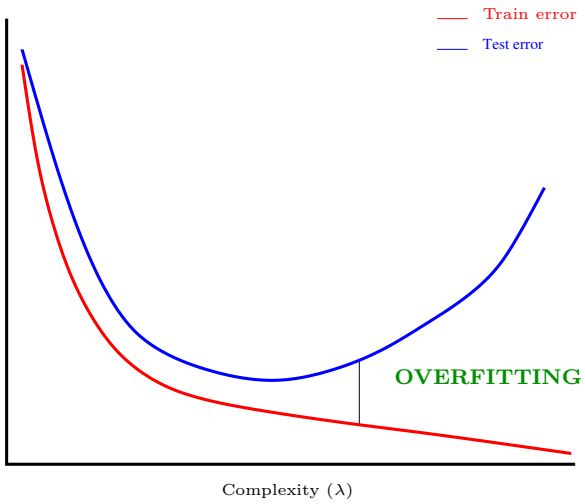
Complexité

— λ petit $\Longrightarrow$ modèle restrictif $\Longrightarrow$ mauvais ajustement $\Longrightarrow$ biais $\nearrow$, variance $\searrow$

— λ grand $\Longrightarrow$ modèle flexible (complexe) $\Longrightarrow$ sur-apprentissage (ou overfitting) $\Longrightarrow$ biais $\searrow$, variance $\nearrow$

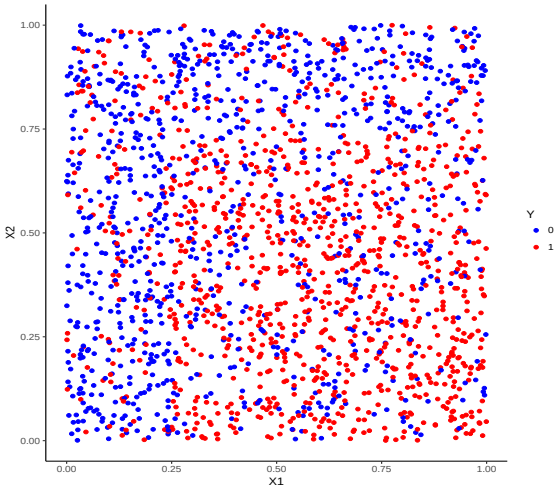
Overfitting

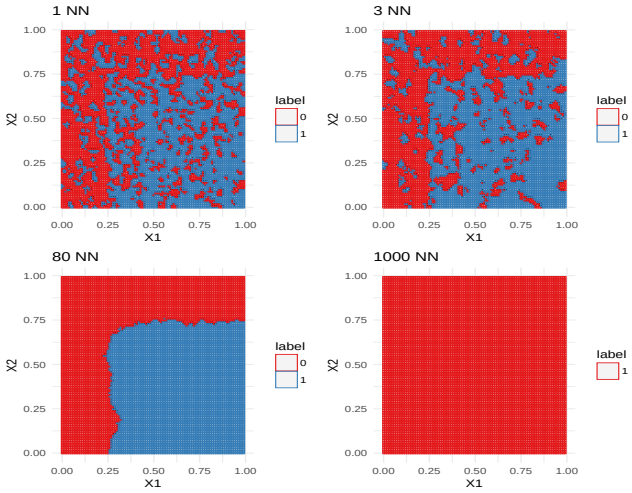
Très bon ajustement sur les données d'apprentissage (i.e. $g(X_i) = Y_i$) mais faible performance prédictive sur des nouveaux individus.



Un exemple

— On cherche à expliquer une variable Y *binaire* par 2 variables X_1 et X_2 quantitatives. On dispose de $n = 2000$ observations.


1.00
0.75
0.50
0.25
0.00
X2
0.00
0.25
0.50
0.75
1.00
X1
Y
0
1


1 NN
3 NN
80 NN
1000 NN
X1
X2
label
0
1

Overfitting pour les k-ppv

Conclusion

On sur-apprend pour les petites valeurs de k.

Deuxième partie

L'analyse discriminante

— *Modèle de référence* permettant d'expliquer une variable qualitative Y par plusieurs variables $X_1, \ldots, X_p$.

— Approche *modèle* mais aussi *géométrique* pour caractériser cette méthode.

— Références : [Saporta, 2011] et [Hastie et al., 2009].

Notations

— n-échantillon i.i.d $\mathcal{D}_n = (X_1, Y_1), \ldots, (X_n, Y_n)$ avec X_i à valeurs dans $\mathbb{R}^p$ et Y_i dans $\mathcal{Y} = \{1, \ldots, K\}$.
— On veut *estimer* $\mathbf{P}(Y = k|X = x), k = 1, \ldots, K$.

Notations

On note :
— $f_k(x), k = 1, \ldots, K$ les densités des lois de $X|Y = k$;
— $f(x)$ la densité de X.
— $\pi_k = \mathbf{P}(Y = k)$ les probabilités a priori d'appartenance aux groupes.

Formule de Bayes

Théorème de Bayes

Les probabilités *a posteriori* d'appartenance aux groupes $1, \ldots, K$ sont données par

$$\mathbf{P}(Y = k|X = x) = \frac{\pi_k f_k(x)}{\sum_{\ell=1}^{K} \pi_\ell f_\ell(x)}.$$

Conséquence

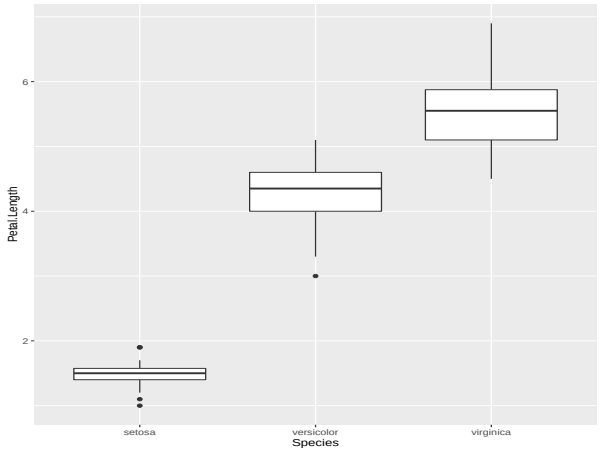
Une bonne estimation des densités de $X|Y = k$ nous donnera une bonne estimation des probabilités $\mathbf{P}(Y = k|X = x)$.

1 Le modèle d'analyse discriminante linéaire

1.1 Une seule variable explicative

— On commence d'abord par expliquer l'espèce des iris par la longueur des pétales *uniquement*.
— On peut visualer ce problème à l'aide d'un boxplot.

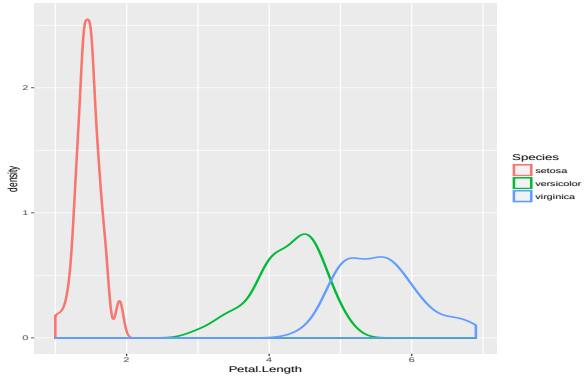
```
> ggplot(iris)+aes(x=Species,y=Petal.Length)+geom_boxplot()+theme_bw()
```



Représentation sous forme de densités

— La fonction geom_density permet de représenter des estimateurs des densités conditionnelles des lois conditionnelles de $X|Y = j$, $j = 1, 2, 3$.

```
> ggplot(iris)+aes(x=Petal.Length,color=Species)+geom_density(size=1)
```



Un modèle

— Les trois densités conditionnelles du graphe précédent ressemblent à des densités *gaussiennes*.

— Si on désigne par X la variable (quantitative) *Petal.Length* et par Y la variable (qualitative) *Species*, on peut être tenté de supposer que les lois de X sachant $Y = k$ sont des lois gaussiennes $\mathcal{N}(\mu_k, \sigma^2)$, $k = 1, 2, 3$.

— La densité de X *sachant* $Y = k$ s'écrit alors

$$f_{X|Y=k}(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(x-\mu_k)^2}{2\sigma^2}\right).$$

Estimation des paramètres inconnus

— Pour calculer les *probabilités a posteriori* $\mathbf{P}(Y = k|X = x)$ il faut estimer les paramètres inconnus du modèle :
  — Les paramètres μ_k et σ^2 des lois gaussiennes ;
  — Les probabilités a priori $\pi_k = \mathbf{P}(Y = k)$.

Les estimateurs

Ces quantités sont naturellement estimées à partir des données $(X_1, Y_1), \dots, (X_n, Y_n)$ selon

$$\widehat{\mu}_k = \frac{1}{n_k} \sum_{i:Y_i=k} X_i, \quad \widehat{\sigma^2} = \frac{1}{n-2} \sum_{k=1}^{K} \sum_{i:Y_i=k} (X_i - \widehat{\mu}_k)^2$$

$$\widehat{\pi}_k = \frac{n_k}{n} \quad \text{avec} \quad n_k = \sum_{i=1}^{n} \mathbf{1}_{\{Y_i=k\}}$$

Exemple sur R

```
> model <- lda(Species~Petal.Length,data=iris)
> model
Call:
lda(Species ~ Petal.Length, data = iris)

Prior probabilities of groups:
    setosa versicolor  virginica
 0.3333333  0.3333333  0.3333333

Group means:
           Petal.Length
setosa            1.462
versicolor        4.260
virginica         5.552

Coefficients of linear discriminants:
                LD1
Petal.Length 2.323774
```

Prévisions

— **predict** permet de *prédire* l'espèce de nouveaux iris uniquement à partir de leur longueur de pétales

```
> don_pred
    Sepal.Length Sepal.Width Petal.Length Petal.Width
           5.0         3.6          1.4         0.2
           5.5         2.4          3.7         1.0
           7.1         3.0          5.9         2.1
           6.7         3.3          5.7         2.5

> predict(model,newdata=don_pred)
$class
[1] setosa     versicolor virginica  virginica
Levels: setosa versicolor virginica
$posterior
         setosa   versicolor    virginica
    1.000000e+00 2.589892e-10 6.170197e-21
    3.123152e-06 9.997752e-01 2.217125e-04
    1.113402e-23 9.723296e-04 9.990277e-01
    9.198362e-22 3.913109e-03 9.960869e-01
```

1.2 LDA : cas général

— On souhaite maintenant *expliquer* l'espèce des iris par les 4 variables explicatives `Sepal.Length`, `Sepal.Width`, `Petal.Length`, `Petal.Width`. On notera X_1, X_2, X_3, X_4 ces 4 variables et $X = (X_1, X_2, X_3, X_4)$.

— La méthodologie est *identique* au cas précédent :

1. On modélise les lois conditionnelles de $X|Y = k$ par des lois gaussiennes *multivariées*.
2. On utilise la *formule de Bayes* pour en déduire la loi de $Y|X = x$.

LDA : cas général

— La loi de $X|Y = k$ est *modélisée par une loi normale multivariée* $\mathcal{N}(\mu_k, \Sigma)$ où $\mu_k \in \mathbb{R}^p$ et Σ est une matrice $p \times p$ définie positive. La densité de $X|Y = k$ est alors donnée par :

$$f_{X|Y=k}(x) = \frac{1}{(2\pi \det(\Sigma))^{p/2}} \exp\left(-\frac{1}{2}(x-\mu_k)^t \Sigma^{-1}(x-\mu_k)\right).$$

— La loi conditionnelle de $Y|X = x$ se déduit de la *formule de Bayes*

$$\mathbf{P}(Y = k|X = x) = \frac{\pi_k f_{X|Y=k}(x)}{f(x)}$$

où $f(x)$, la densité de X, se déduit des densités conditionnelles $f_{X|Y=k}(x)$ et des probabilités a priori $\pi_k = \mathbf{P}(Y = k)$.

Estimations

— Ici encore il faut *estimer les paramètres inconnus* du modèle :
  — les vecteurs $\mu_k, k = 1, \ldots, K$ et la matrice de variance-covariance Σ des lois gaussiennes ;
  — les probabilités a priori $\pi_k = \mathbf{P}(Y = k)$.

Les estimateurs

Ces quantités sont naturellement estimées à partir des données $(X_1, Y_1), \ldots, (X_n, Y_n)$ selon

$$\widehat{\mu}_k = \frac{1}{n_k} \sum_{i:Y_i=k} X_i, \quad \widehat{\Sigma} = \frac{1}{n-K} \sum_{k=1}^{K} \sum_{i:Y_i=k} (X_i - \widehat{\mu}_k)(X_i - \widehat{\mu}_k)^t$$

$$\widehat{\pi}_k = \frac{n_k}{n} \quad \text{avec} \quad n_k = \sum_{i=1}^{n} \mathbf{1}_{\{Y_i=k\}}.$$

Exemple sur R

```
> model_complet<- lda(Species~.,data=iris)
> model_complet
Call:
lda(Species ~ ., data = iris)

Prior probabilities of groups:
    setosa versicolor  virginica
 0.3333333  0.3333333  0.3333333

Group means:
           Sepal.Length Sepal.Width Petal.Length Petal.Width
setosa            5.006       3.428        1.462       0.246
versicolor        5.936       2.770        4.260       1.326
virginica         6.588       2.974        5.552       2.026
```

Prévisions

— La fonction *predict* permet de *prédire* le groupe de nouveaux individus :

```
> don_pred
    Sepal.Length Sepal.Width Petal.Length Petal.Width
           5.0         3.6          1.4         0.2
           5.5         2.4          3.7         1.0
           7.1         3.0          5.9         2.1
           6.7         3.3          5.7         2.5
```

```
> predict(model_complet,newdata=don_pred)
$class
[1] setosa     versicolor virginica  virginica
Levels: setosa versicolor virginica

$posterior
          setosa   versicolor    virginica
5   1.000000e+00 1.637387e-22 1.082605e-42
82  9.648075e-16 9.999997e-01 3.266704e-07
103 1.231264e-42 2.592826e-05 9.999741e-01
145 4.048249e-46 2.524984e-07 9.999997e-01
```

Règle de classification

— La manière la plus naturelle de classer une nouvelle observation $x \in \mathbb{R}^p$ est de *choisir le groupe qui maximise*

$$\mathbf{P}(Y = k|X = x).$$

— Comparons les valeurs de ces probabilités pour les groupes k et ℓ :

$$\begin{aligned}\log \frac{\mathbf{P}(Y = k|X = x)}{\mathbf{P}(Y = \ell|X = x)} &= \log \frac{f_k(x)}{f_\ell(x)} + \log \frac{\pi_k}{\pi_\ell} \\ &= \log \frac{\pi_k}{\pi_\ell} - \frac{1}{2}(\mu_k + \mu_\ell)^t \Sigma^{-1}(\mu_k - \mu_\ell) \\ &\quad + x^t \Sigma^{-1}(\mu_k - \mu_\ell) \qquad (1)\end{aligned}$$

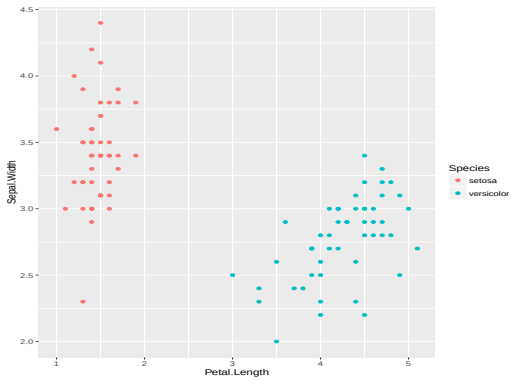
Conclusion

La frontière entre les classes k et ℓ est linéaire en x !

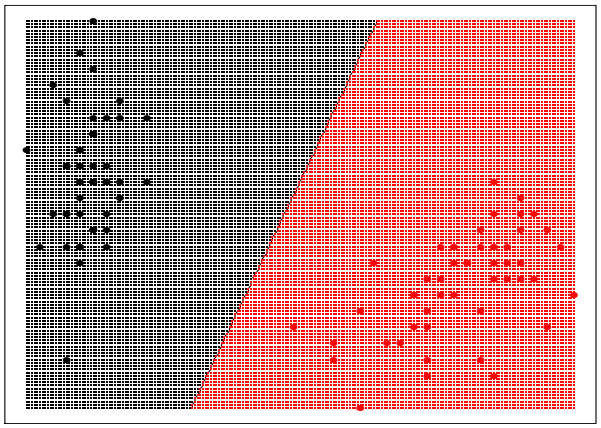
Exemple

— *Frontière* LDA entre "Setosa" et "Versicolor" avec 2 variables

```
> iris1 <- iris %>% filter(Species%in%c("setosa","versicolor")) %>%
       select(Petal.Length,Sepal.Width,Species)
> ggplot(iris1)+aes(x=Petal.Length,y=Sepal.Width,color=Species)+geom_point()
```



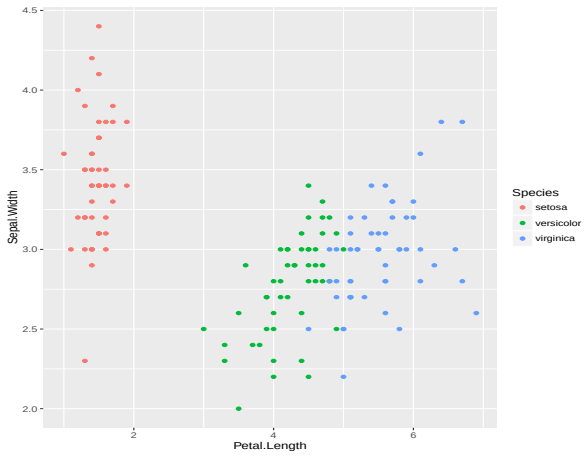
Frontière deux classes



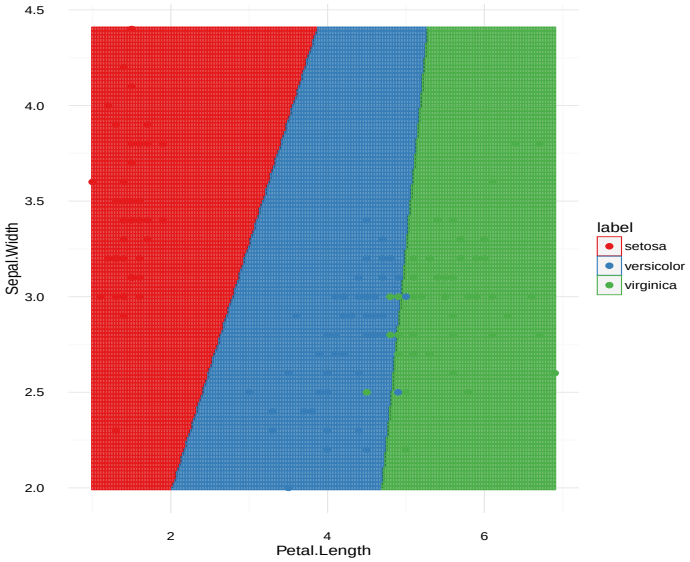
Exemple - 3 classes

— On fait de même pour les 3 espèces (*3 classes*).

```
> ggplot(iris)+aes(x=Petal.Length,y=Sepal.Width,color=Species)+geom_point()
```



Frontière trois classes



Fonctions linéaires discriminantes

Définition

On appelle *fonctions linéaires discriminantes* les fonctions

$$\delta_k(x) = x^t\Sigma^{-1}\mu_k - \frac{1}{2}\mu_k^t\Sigma^{-1}\mu_k + \log \pi_k, \quad k = 1, \ldots, K.$$

Propriété

D'après (1),

$$\underset{k}{\operatorname{argmax}}\, \mathbf{P}(Y = k|X = x) = \underset{k}{\operatorname{argmax}}\, \delta_k(x).$$

Conclusion

Choisir le groupe qui *maximise les probabilités a posteriori* revient à choisir le groupe qui *maximise les fonctions linéaires discriminantes.*

2 Réduction de dimension

— LDA a été présentée comme une méthode de classification *restreinte à un modèle gaussien.*

— La popularité de cette approche est également (surtout ?) due à une *vision géométrique* de cette méthode.

— L'analyse discriminante linéaire s'interprète également comme une méthode de *réduction de dimension* (démarche similaire à l'ACP).

— C'est également un outil de *visualisation de données.*

Introduction

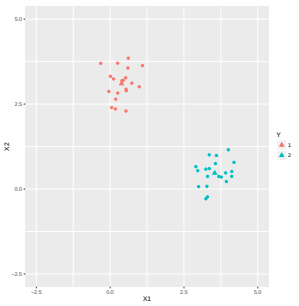
— *Données* : $(x_1, y_1), \dots, (x_n, y_n)$ où $x_i \in \mathbb{R}^p$ et $y_i \in \{1, \dots, K\}$.
— *Problème* : expliquer les sorties y_i par les entrées x_i.
— Traditionnellement l'*analyse discriminante* se présente selon deux aspects :
1. objectif prédictif (partie précédente) : il s'agit de prédire le groupe d'un nouvel individu $x \in \mathbb{R}^p$;
2. objectif descriptif (cette partie) : il s'agit de trouver des sous-espaces de faibles dimensions tels que les observations projetées sur ces sous-espaces soient *au mieux* séparées.

2.1 Recherche d'axes discriminants

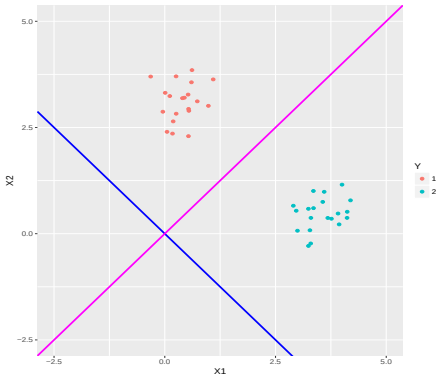
Notations

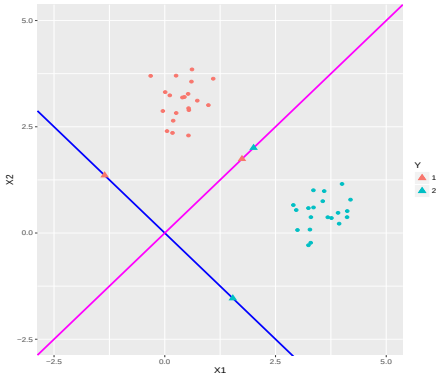
— *Données* : $(x_1, y_1), \dots, (x_n, y_n)$ où $x_i \in \mathbb{R}^p$ et $y_i \in \{1, \dots, K\}$.
— g le centre de gravité des données $g = \frac{1}{n} \sum_{i=1}^{n} x_i$.
— g_k le centre de gravité du groupe k :

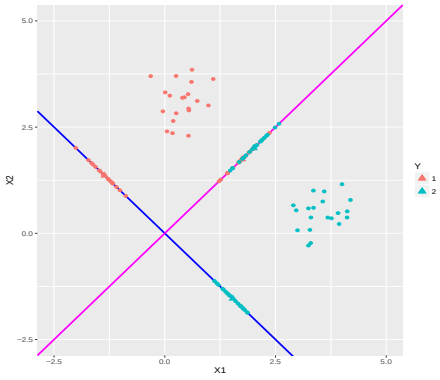
$$g_k = \frac{1}{n_k} \sum_{i : y_i = k} x_i.$$



Le problème







Le problème

Trouver un sous espace de dimension 1 tel que les observations projetées sur ce sous espace soient au mieux séparées.

L'approche de Fisher

Axe discriminant

Chercher une combinaison linéaire $a_1X_1 + \ldots + a_pX_p$ telle que

1. les centres de gravité de chaque groupe projetés sur cet axe soient au mieux séparés ;
2. la distance entre les observations projetées et leur centre de gravité projeté soit minimale.

Cette approche revient à

— *maximiser* la distance (ou variance) inter-classes ;

— *minimiser* la distance (ou variance) intra-classes.

Décomposition de la variance

— Variance *totale*

$$V = \frac{1}{n}\sum_{i=1}^{n}(X_i - g)(X_i - g)^t.$$

— Variance *inter-classes* (between)

$$B = \frac{1}{n}\sum_{k=1}^{K} n_k(g_k - g)(g_k - g)^t.$$

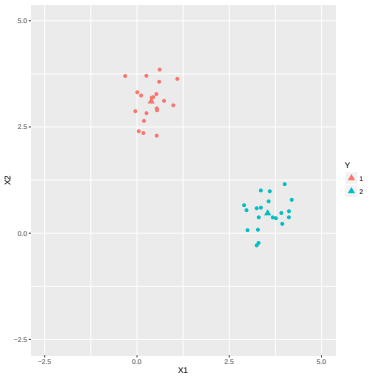
— Variance *intra-classes* (within)

$$W = \frac{1}{n}\sum_{k=1}^{K} n_k V_k \quad \text{avec} \quad V_k = \frac{1}{n_k}\sum_{i:Y_i=k}(X_i - g_k)(X_i - g_k)^t.$$

Propriété

$$V = B + W$$

Exemple



$$\begin{pmatrix} 2.63 & -2.04 \\ -2.04 & 1.90 \end{pmatrix} = \begin{pmatrix} 2.49 & -2.08 \\ -2.08 & 1.73 \end{pmatrix} + \begin{pmatrix} 0.14 & 0.04 \\ 0.03 & 0.17 \end{pmatrix}$$

Projection - Rappels

— Le *projeté* d'un vecteur u sur la droite engendrée par un vecteur v est

$$\pi_v(u) = \frac{\langle u, v\rangle}{\|v\|^2} v.$$

— Si v est de norme 1, alors

$$\|\pi_v(u)\|^2 = v^t u u^t v.$$

Variances projetées

Soit $a \in \mathbb{R}^p$ de *norme 1* :

— Variance totale sur vect(a) :

$$V(a) = \frac{1}{n}\sum_{i=1}^{n} \|\pi_a(X_i) - \pi_a(g)\|^2 = a^t V a.$$

V(a)	B(a)	W(a)
0.218	0.034	0.184
4.308	4.187	0.121

— Variance inter sur vect(a) :

$$B(a) = \frac{1}{n}\sum_{k=1}^{K} n_k \|\pi_a(g_k) - \pi_a(g)\|^2 = a^t B a.$$

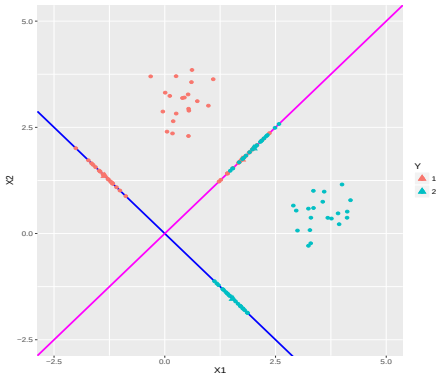
— Variance intra sur vect(a) :

$$W(a) = \frac{1}{n}\sum_{k=1}^{K} \sum_{i:Y_i=K} \|\pi_a(X_i) - \pi_a(g_k)\|^2 = a^t W a.$$

Propriété

$$V(a) = B(a) + W(a).$$

Exemple



Axe discriminant

Un axe a est *discriminant* si

— Variance *inter* projetée grande $\Longrightarrow$ $B(a)$ grande

— Variance *intra* projetée petite $\Longrightarrow$ $W(a)$ petite.

Coefficient de Rayleigh

Fisher propose d'utiliser comme mesure de la qualité d'un axe de discrimination le coefficient de Rayleigh

$$J(a) = \frac{a^t B a}{a^t W a}.$$

Première variable discriminante

Le problème d'optimisation

Le problème consiste à trouver $a \in \mathbb{R}^p$ qui maximise le coefficient de Rayleigh

$$\frac{a^t B a}{a^t W a},$$

ou de façon équivalente

$$\max_a a^t B a \quad \text{sous la contrainte} \quad a^t W a = 1.$$

Solution

Elle est donnée par un *vecteur propre* associé à la plus *grande valeur propre* de $W^{-1}B$.

V(a)	B(a)	W(a)	Rayleigh
0.218	0.034	0.184	0.185
4.308	4.187	0.121	34.603
4.325	4.208	0.117	35.966

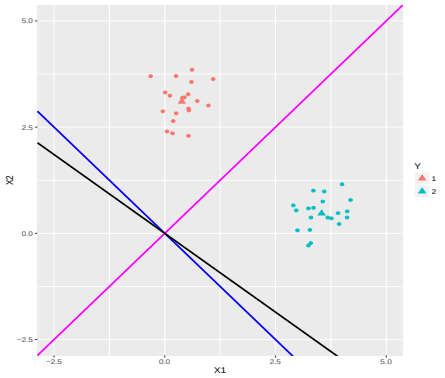
Exemple

```
> mod
Call:
lda(Y ~ ., data = D)

Prior probabilities of groups:
  1   2
0.5 0.5

Group means:
         X1        X2
1 0.3850758 3.1009709
2 3.5410917 0.4692031

Coefficients of linear discriminants:
         LD1
X1  2.284995
X2 -1.694860
```

Sorties R

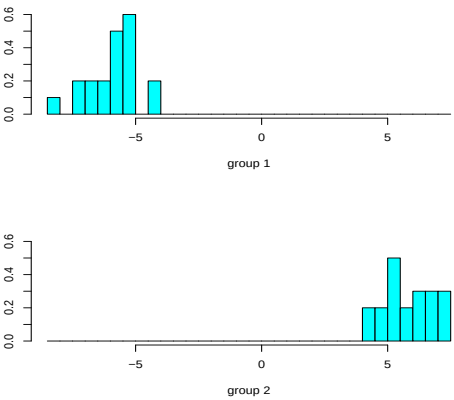
On a $a_1 = 2.284995$ et $a_2 = -1.694860$.

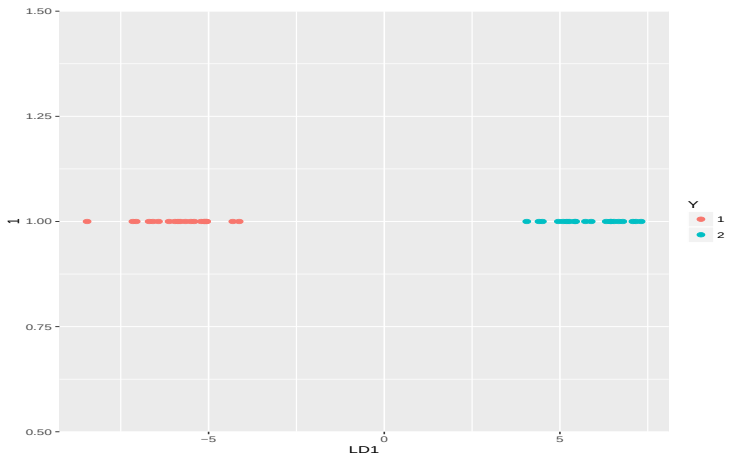
Exemple



plot.lda

```
> plot(mod)
```





— On peut également représenter les *projections* des individus sur le *premier axe discriminant*

```
> score1 <- predict(mod)$x
> donnees1 <- data.frame(score1,Y=D$Y)
> ggplot(donnees1)+aes(x=LD1,y=1,col=Y)+geom_point(size=2)
```

Autres axes

— Les $a_1, \ldots, a_p$ s'appellent *coordonnées discriminantes*.

— La variable $a_1X_1 + \ldots + a_pX_p$ s'appelle *première variable discriminante* (ou première variable *canonique*).

— Les centres $g_1, \ldots, g_K$ appartiennent à un espace de *dimension* $K-1$. Si $K \geq 3$, on peut poursuivre les projections (comme pour l'ACP).

Calcul des autres variables discriminantes

— On cherche a_2 orthogonal à a_1 (par rapport à W) qui maximise $\frac{a_2^t B a_2}{a_2^t W a_2}$.

— La solution est donnée par le vecteur propre associé à la deuxième plus grande valeur propre de $W^{-1}B$.

Remarque

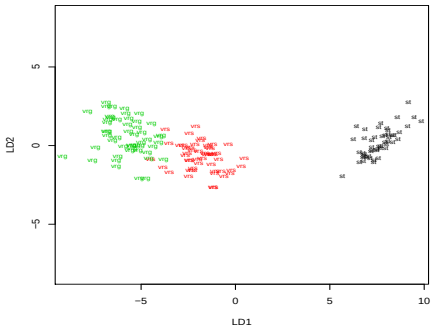
La matrice $W^{-1}B$ possède au plus $K-1$ *valeurs propres non nulles*, on peut donc avoir au maximum $K-1$ variables discriminantes.

Les iris de Fisher

```
> mod1 <- lda(Species~.,data=iris)
> mod1
Prior probabilities of groups:
    setosa versicolor  virginica
 0.3333333  0.3333333  0.3333333
Group means:
           Sepal.Length Sepal.Width Petal.Length Petal.Width
setosa            5.006       3.428        1.462       0.246
versicolor        5.936       2.770        4.260       1.326
virginica         6.588       2.974        5.552       2.026

Coefficients of linear discriminants:
                    LD1         LD2
Sepal.Length  0.8293776  0.02410215
Sepal.Width   1.5344731  2.16452123
Petal.Length -2.2012117 -0.93192121
Petal.Width  -2.8104603  2.83918785

Proportion of trace:
   LD1    LD2
0.9912 0.0088
```

Représentation des individus sur les deux premiers axes

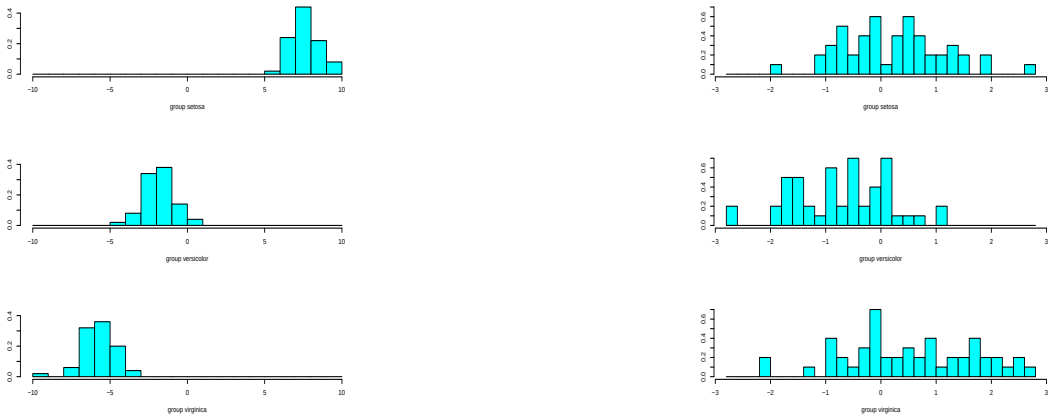
```
> plot(mod1)
```



Comparaison des axes discriminants

Le premier axe est (clairement) plus discriminant que le second.

Représentation des groupes par axes



Interprétation

On visualise à nouveau que le premier axe est beaucoup plus discriminant que le second.

Performances des variables canoniques

— On a

$$\frac{a_k^t B a_k}{a_k^t W a_k} = \lambda_k$$

où λ_k est la k-ème valeur propre de $W^{-1}B$

Une mesure de performance

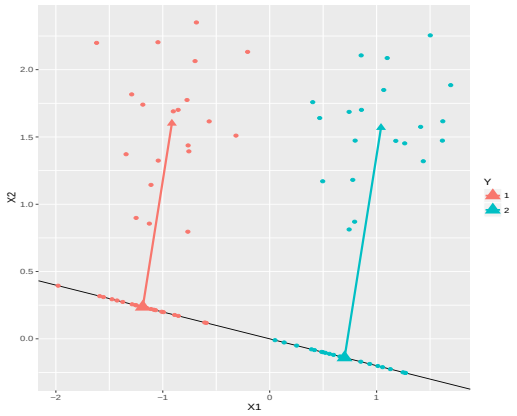
On peut donc mesurer la performance de la k-ème variable canonique par

$$\frac{\lambda_k}{\sum_{j=1}^{K-1} \lambda_j}.$$

```
Proportion of trace:
   LD1    LD2
0.9912 0.0088
```

2.2 Classification

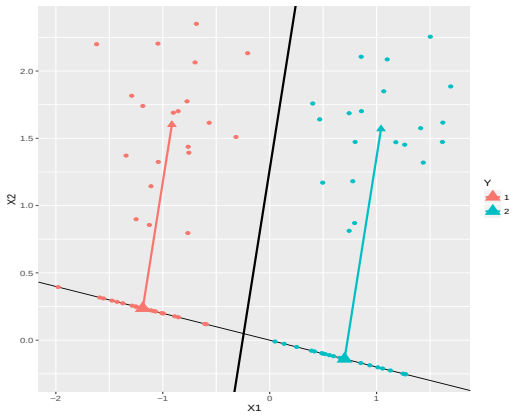
Le problème de la classification



Question

Comment classer un nouveau point $x = (x_1, x_2)$?

Une idée naturelle



Réponse

Utiliser l'axe orthogonal à l'axe discriminant passant par le point équidistant des projetés des centres de gravité.

Règle de Mahalanobis

Règle géométrique

On classe x dans le groupe 1 si

$$\|\pi_a(x) - \pi_a(g_1)\| \leq \|\pi_a(x) - \pi_a(g_2)\|.$$

Propriété

La règle géométrique est équivalente à classer x dans le groupe qui minimise la *distance de Mahalanobis*

$$d(x, g_k) = (x - g_k)^t W^{-1}(x - g_k).$$

— La propriété se généralise à un nombre de groupes K quelconque.

Lien LDA descriptive/Prédictive

LDA prédictive

On affecte un nouvel individu x au groupe k qui maximise

— la probabilité a posteriori : $\mathbf{P}(Y = k|x = x)$

— la fonction linéaire discriminante

$$\delta_k(x) = x^t \Sigma^{-1} \mu_k - \frac{1}{2} \mu_k^t \Sigma^{-1} \mu_k + \log \pi_k$$

— la distance de Mahalanobis "corrigée"

$$-\frac{1}{2}(x - \mu_k)^t \Sigma^{-1}(x - \mu_k) + \log \pi_k$$

LDA géométrique

On affecte un nouvel individu x au groupe k qui minimise la *distance de Mahalanobis*

$$d(x, g_k) = (x - g_k)^t W^{-1}(x - g_k).$$

Remarque (importante)

Dans le cas où on estime

— μ_k par g_k

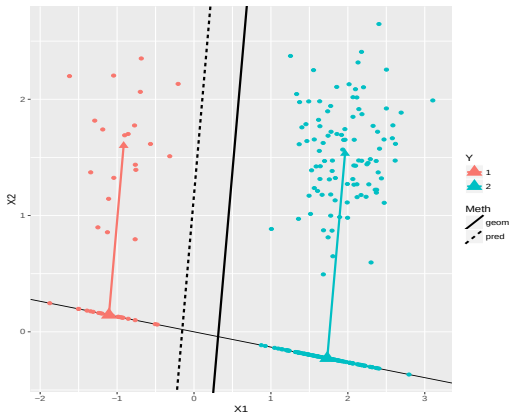
— Σ par W,

et que $\pi_k = 1/K, k = 1, \ldots, K$ les règles *prédictives et géométriques coïncident.*

Conséquence

La règle géométrique correspond à la règle probabiliste lorsque les probabilités a priori de chaque groupe sont identiques.

Exemple



Remarque

La règle géométrique "favorise" les groupes à faibles effectifs.

Quelques tests

— LDA peut-être accompagnée de quelques *tests statistiques*.

— Par exemple : $H_0 : \mu_1 = \ldots = \mu_K = 0$.

— Λ de Wilks :

$$\Lambda = \frac{|W|}{|V|} = \frac{|W|}{|W+B|}$$

suit la *loi de Wilks* de paramètres $(p, n-K, K-1)$ sous H_0.

— Lawley-Hotelling : $\mathrm{tr}(W^{-1}B)$ suit la loi de T_0^2 généralisé de *Hotelling* sous H_0 (approximable par un $\chi^2_{p(K-1)}$).

Exemple

— Sous R, la fonction manova permet de mettre en œuvre ces tests.

```
> D  <- as.matrix(iris[,1:4])
> mod <- manova(D~iris$Species)
> summary(mod,test="Wilks")
             Df    Wilks approx F num Df den Df    Pr(>F)
iris$Species   2 0.023439   199.15      8    288 < 2.2e-16 ***
Residuals    147
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
> summary(mod,test="Hotelling-Lawley")
             Df Hotelling-Lawley approx F num Df den Df    Pr(>F)
iris$Species   2           32.477   580.53      8    286 < 2.2e-16 ***
Residuals    147
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

3 Analyse discriminante quadratique et régularisation

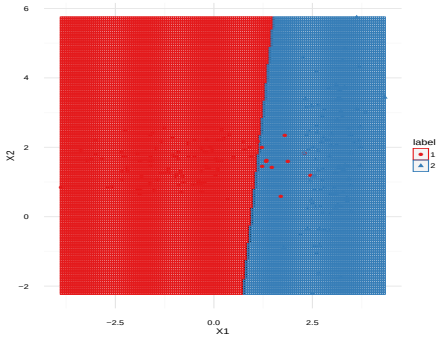
3.1 Analyse discriminante quadratique

Rappels LDA

1. Suppose $X|Y = k \sim \mathcal{N}(\mu_k, \Sigma)$;
2. Estime μ_k et Σ par *maximum de vraisemblance* ;
3. Bayes pour obtenir les *probabilités a posteriori*

$$\mathbf{P}(Y = k|X = x) = \frac{\pi_k f_k(x)}{\sum_{\ell=1}^{K} \pi_\ell f_\ell(x)}.$$

Exemple



Remarques

— LDA peut être *mise en défaut* lorsque les *matrices de variance-covariance* sont *différentes*.

— L'analyse discriminante quadratique propose d'utiliser des matrices de variance-covariance différentes pour chaque groupe.

QDA

Modèle QDA

$X|Y = k \sim \mathcal{N}(\mu_k, \Sigma_k)$.

Estimation

— Les paramètres μ_k et $\pi_k = \mathbf{P}(Y = k)$ sont estimés de la même façon que pour l'analyse discriminante linéaire.

— Les matrices de variance-covariance Σ_k sont « naturellement » estimées selon

$$\widehat{\Sigma}_k = \frac{1}{n_k} \sum_{i:Y_i=k} (X_i - \widehat{\mu}_k)(X_i - \widehat{\mu}_k)^t.$$

Formule de Bayes

$$\mathbf{P}(Y = k|X = x) = \frac{\pi_k f_k(x)}{\sum_{\ell=1}^{K} \pi_\ell f_\ell(x)}.$$

Fonctions linéaires discriminantes

— Même principe que pour LDA mais *FLD différentes*.

Définition

On appelle *fonctions linéaires discriminantes* (pour QDA) les fonctions

$$\delta_k(x) = -\frac{1}{2} \log |\Sigma_k| - \frac{1}{2}(x - \mu_k)^t \Sigma_k^{-1} (x - \mu_k) + \log \pi_k, \quad k = 1, \ldots, K.$$

Propriété

$$\underset{k}{\operatorname{argmax}}\, \mathbf{P}(Y = k|X = x) = \underset{k}{\operatorname{argmax}}\, \delta_k(x).$$

Conclusion

Choisir le groupe qui *maximise les probabilités a posteriori* revient à choisir le groupe qui *maximise les fonctions linéaires discriminantes*.

Frontières pour QDA

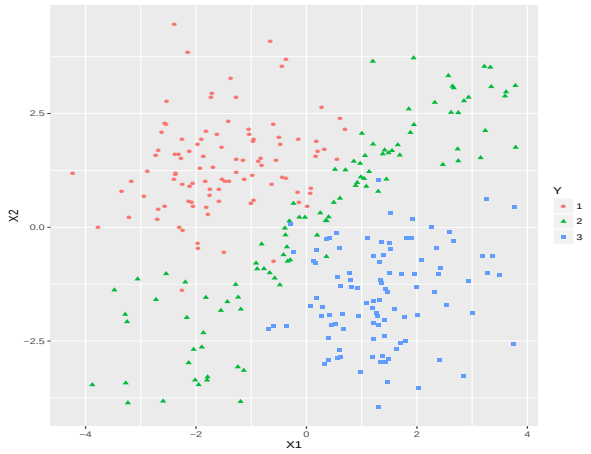
— Les *frontières* entre les groupes k et ℓ

$$\{x \text{ tq } \delta_k(x) = \delta_\ell(x)\}$$

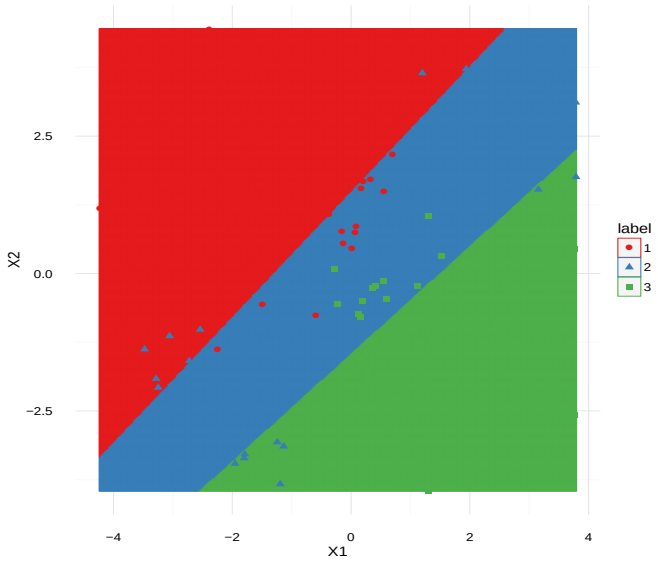
sont ici *quadratiques* en x (linéaires pour LDA).

Exemple

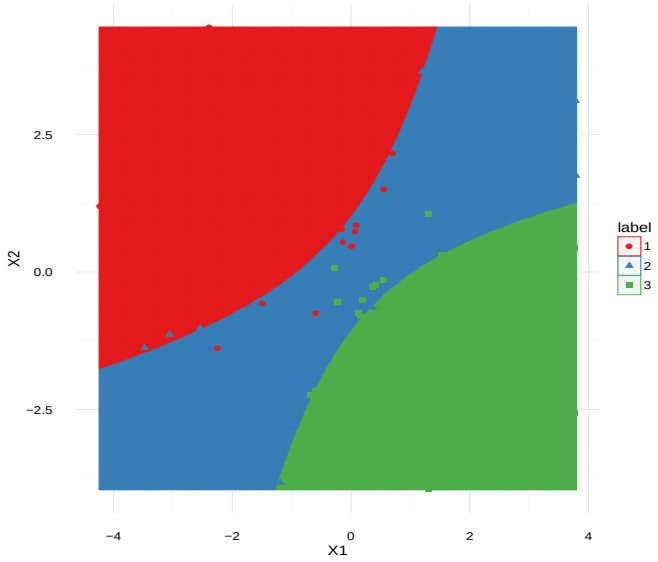
— On compare LDA et QDA sur les données du graphe ci-dessous



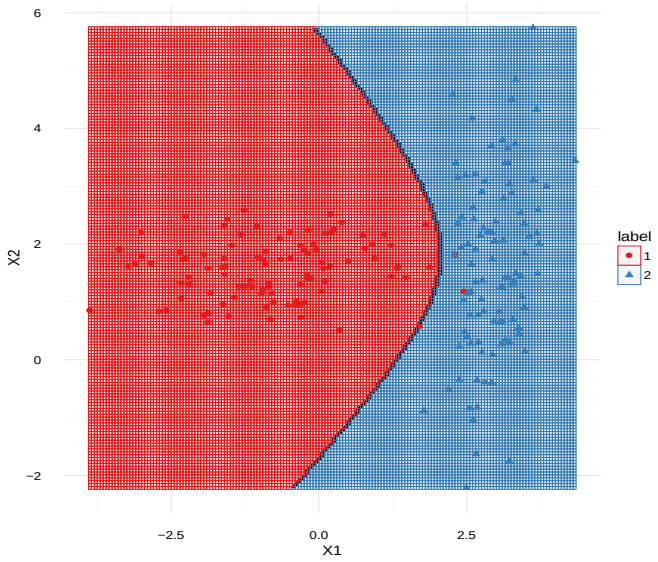
Frontières LDA



Frontières QDA



Autre exemple



LDA vs QDA

— QDA est *plus flexible* que LDA : LDA est en quelques sortes « imbriquée » dans QDA.

— QDA permet donc a priori de modéliser une *gamme plus large* de phénomènes.

— Mais... Le *prix à payer* se situe au niveau de l'estimation :
  — $(K-1)\times(p+1)$ paramètres pour LDA ;
  — $(K-1)\times(p(p+3)/2+1)$ pour QDA.

Conclusion

QDA est plus complexe $\Longrightarrow$ plus de paramètres à estimer $\Longrightarrow$ estimateurs moins précis.

3.2 Régularisation

Régularisation 1

— [Friedman, 1989] propose de *combiner* LDA et QDA.

— On reste dans le *modèle gaussien* mais

— les matrices de *variance-covariance* des lois $X|Y = k$ sont estimées par

$$(1 - \lambda)\widehat{\Sigma}_k + \lambda\widehat{\Sigma}.$$

Role de λ

— $\lambda = 0 \Longrightarrow$ QDA ;
— $\lambda = 1 \Longrightarrow$ LDA.
— $\lambda \in [0, 1]$ est un paramètre à calibrer.

Régularisation 2

— [Friedman, 1989] (toujours...) propose de *régulariser la matrice de variance-covariance* pour LDA :

$$(1 - \gamma)\widehat{\Sigma} + \gamma\hat{\sigma}^2 I_p.$$

Role de γ

— $\gamma = 0 \Longrightarrow$ LDA ;
— $\gamma \in [0, 1]$ est un paramètre à calibrer.

Le coin R

— La fonction rda du package *klaR* permet de combiner les deux pénalités en estimant les matrices de variance-covariance selon

$$(1 - \gamma)\widehat{\Sigma}_k(\lambda) + \gamma\frac{\text{trace}(\widehat{\Sigma}_k(\lambda))}{p} I_p.$$

avec

$$\widehat{\Sigma}_k(\lambda) = (1 - \lambda)\widehat{\Sigma}_k + \lambda\widehat{\Sigma}.$$

Roles de γ ***et*** λ

— $\gamma = 0, \lambda = 0 \Longrightarrow$ QDA ;
— $\gamma = 0, \lambda = 1 \Longrightarrow$ LDA ;
— Le problème est de bien choisir λ et γ.

Exemple

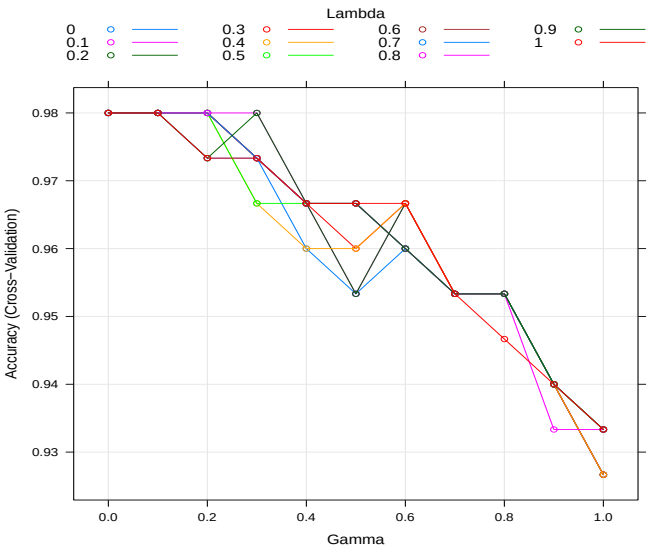
— La fonction *rda* propose de sélectionner automatiquement ces paramètres

```
> set.seed(1234)
> rda(Species~.,data=iris)
Call:
rda(formula = Species ~ ., data = iris)

Regularization parameters:
     gamma     lambda
0.09303661 0.85993116

Prior probabilities of groups:
    setosa versicolor  virginica
 0.3333333  0.3333333  0.3333333

Misclassification rate:
       apparent: 2 %
cross-validated: 2 %
```

Sélection avec caret

— On peut bien entendu également utiliser la fonction *train* du package caret.

```
> ctrl <- trainControl(method="cv")
> gr <- expand.grid(data.frame(gamma=seq(0,1,by=0.1),lambda=seq(0,1,by=0.1)))
> set.seed(12345)
> train(Species~.,data=iris,method="rda",tuneGrid=gr,trControl=ctrl)
Regularized Discriminant Analysis

150 samples
  4 predictor
  3 classes: 'setosa', 'versicolor', 'virginica'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 135, 135, 135, 135, 135, 135, ...
Resampling results across tuning parameters:

  gamma  lambda  Accuracy   Kappa
  0.0    0.0     0.9800000  0.97
  0.0    0.1     0.9800000  0.97
  0.0    0.2     0.9800000  0.97
  0.0    0.3     0.9800000  0.97
  0.0    0.4     0.9800000  0.97
  0.0    0.5     0.9800000  0.97
  0.0    0.6     0.9800000  0.97
  0.0    0.7     0.9800000  0.97
  0.0    0.8     0.9800000  0.97
  0.0    0.9     0.9800000  0.97
  0.0    1.0     0.9800000  0.97
  0.1    0.0     0.9800000  0.97
  0.1    0.1     0.9800000  0.97
  0.1    0.2     0.9800000  0.97
  0.1    0.3     0.9800000  0.97
  0.1    0.4     0.9800000  0.97
  0.1    0.5     0.9800000  0.97
  0.1    0.6     0.9800000  0.97
  0.1    0.7     0.9800000  0.97
  0.1    0.8     0.9800000  0.97
  0.1    0.9     0.9800000  0.97
  0.1    1.0     0.9800000  0.97
  0.2    0.0     0.9733333  0.96
  0.2    0.1     0.9800000  0.97
  0.2    0.2     0.9800000  0.97

Accuracy was used to select the optimal model using  the largest value.
The final values used for the model were gamma = 0 and lambda = 1.
```



Sélection de variables

— Tout comme pour les modèles linéaire et logistique, on peut chercher à *sélectionner des variables* pour un modèle d'analyse discriminante linéaire (les objectifs sont identiques).

— L'approche est similaire, on se donne un *critère de choix de modèle* (par exemple estimation de la probabilité d'erreur) et on utilise des *techniques pas à pas*.

— Sur R, les fonctions stepClass et train des packages *klaR* et *caret* permettent de faire de la sélection de variables.

Sélection avec stepClass

```
> stepclass(Species~.,data=iris,method="lda",direction="both")
 'stepwise classification', using 10-fold cross-validated
correctness rate of method lda'.
150 observations of 4 variables in 3 classes; direction: both
stop criterion: improvement less than 5%.
correctness rate: 0.96;  in: "Petal.Width";  variables (1): Petal.Width

 hr.elapsed min.elapsed sec.elapsed
      0.000       0.000       0.194
method      : lda
final model : Species ~ Petal.Width
<environment: 0x12d5b5e38>
correctness rate = 0.96
```

Sélection avec train

```
> slda <- train(Species ~ ., data = iris,
              method = "stepLDA",
              trControl = trainControl(method = "cv"))
> slda$finalModel
method      : lda
final model : y ~ Petal.Width
<environment: 0x12c509298>
correctness rate = 0.96
```

Bilan

— L'analyse discriminante est une méthode *simple* permettant de répondre au problème de classification supervisée.

— Elle est implémentée dans tous les *logiciels statistiques*.

— Elle peut se révéler performante même lorsque les "hypothèses modèles" ne sont pas vérifiées (justifié par l'approche géométrique).

— Plutôt utilisée pour des variables explicatives quantitatives à la base mais *peut s'adapter à des variables qualitatives* :

1. codage disjonctif des variables qualitatives ;
2. faire une analyse discriminante sur les axes d'une analyse des correspondances multiples (ACM) $\Longrightarrow$ *méthode DISQUAL* (voir [Saporta, 2011]).

Troisième partie

Régression logistique

1 Présentation du modèle

Pathologie concernant les artères coronaires

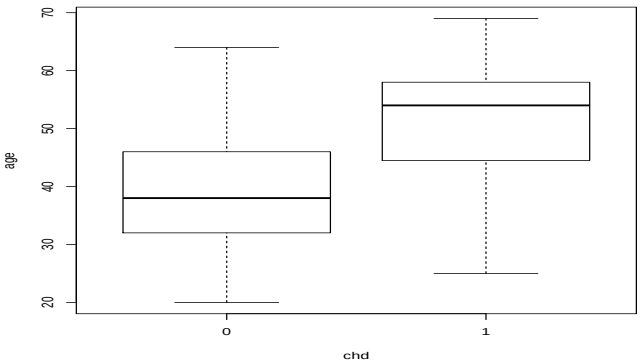
— *Problème* : étudier la présence d'une pathologie concernant les artères coronaires en fonction de l'âge des individus.

— *Données :* on dispose d'un échantillon de taille 100 sur lequel on a mesuré les variables :
  — chd qui vaut 1 si la pathologie est présente, 0 sinon ;
  — age qui correspond à l'âge de l'individu.

```
> artere[1:5,]
   age agrp chd
1.  20    1   0
2.  23    1   0
3.  24    1   0
4.  25    1   0
5.  25    1   1
```

Boxplot

```
> plot(age~chd,data=artere)
```



Interprétation

Il semble que la maladie soit plus présente chez les personnes agées.

Début de modélisation

Question

Comment *expliquer* la relation entre la maladie et l'âge ?

— On désigne par
  — Y la variable qui prend pour valeur 1 si l'individu est atteint, 0 sinon.
  — X la variable qui correspond à l'âge de l'individu.

Le problème

Quantifier la relation entre Y et X à partir des données $(x_1, y_1), \ldots, (x_n, y_n)$ de taille $n = 100$.

Première idée

— On se base sur le *modèle linéaire.*

— On suppose que les deux variables Y et X sont liées par une relation de la forme

$$Y = \beta_0 + \beta_1 X + \varepsilon \tag{2}$$

où $\beta_0 \in \mathbb{R}$ et $\beta_1 \in \mathbb{R}$ sont les *paramètres inconnus* du modèle et ε est une variable aléatoire de loi $\mathcal{N}(0, \sigma^2)$.

Problème

La variable Y est ici qualitative, l'écriture (2) n'a donc aucun sens. $\Longrightarrow$ *mauvaise idée*

Loi conditionnelle

— Chercher à expliquer Y par X revient à chercher de l'information sur la *loi de probabilité de* Y *sachant* X.

— En effet, le modèle de régression linéaire peut se réécrire en caractérisant la loi de $Y|X = x$ par la loi $\mathcal{N}(\beta_0 + \beta_1 x, \sigma^2)$.

Idée

— Etendre cette caractérisation à notre contexte (où la variable à expliquer est binaire).
— Une loi candidate naturelle pour la variable $Y|X = x$ est la loi de Bernoulli.

Loi de Bernoulli

— On va ainsi caractériser la loi de $Y|X = x$ par la loi de Bernoulli.
— Cette loi dépend d'un *paramètre*

$$p(x) = \mathbf{P}(Y = 1|X = x).$$

— Sachant $X = x$, on a donc

$$Y = \begin{cases} 1 & \text{avec probabilité } p(x) \\ 0 & \text{avec probabilité } 1 - p(x) \end{cases}$$

La modélisation

Il reste maintenant à caractériser la probabilité $p(x)$.

Première idée

— Là encore, on peut se baser sur le *modèle linéaire* et proposer

$$p(x) = \beta_0 + \beta_1 x.$$

— Cette écriture n'est pas satisfaisante. En effet

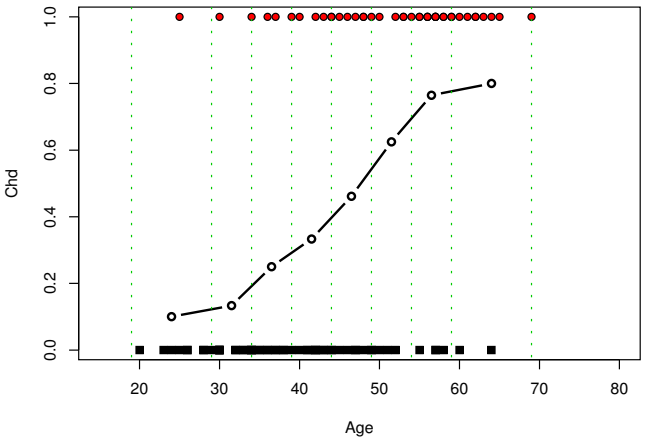
— $p(x) \in [0, 1]$ tandis que $\beta_0 + \beta_1 x \in \mathbb{R}$.

— *Idée :* trouver une transformation φ de $p(x)$ telle que $\varphi(p(x))$ prenne ses valeurs dans $\mathbb{R}$.

Visualisation : découpage en classes d'âge

Age	n	Absent	Présent	Proportion
$[19, 29[$	10	9	1	.10
$[29, 34[$	15	13	2	.13
$[34, 39[$	12	9	3	.25
$[39, 44[$	15	10	5	.33
$[44, 49[$	13	7	6	.46
$[49, 54[$	8	3	5	.625
$[54, 59[$	17	4	13	.76
$[59, 69[$	10	2	8	.8

Représentation graphique



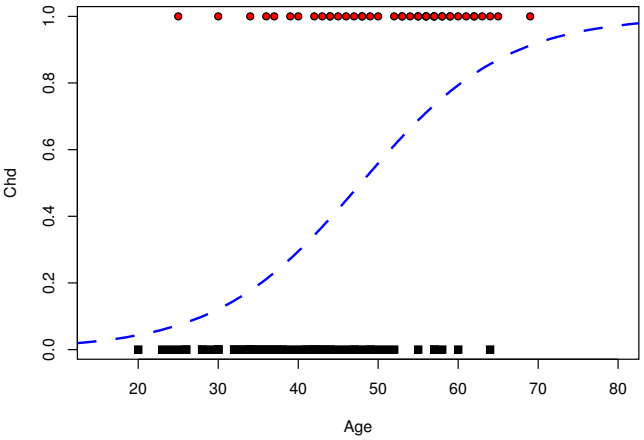
Pour aller plus loin

On souhaiterait trouver une *fonction*

— un peu plus régulière

— qui utilise toutes les données

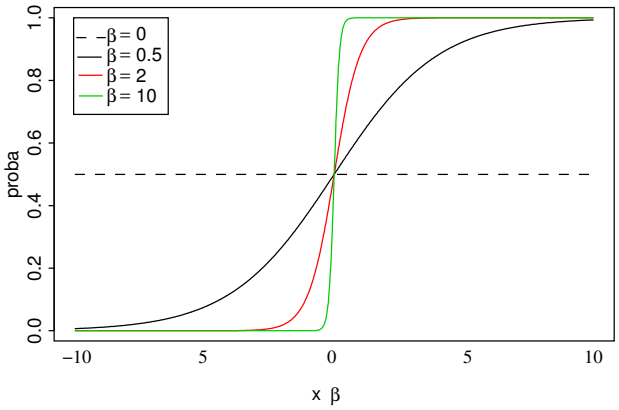
pour obtenir par exemple



Equation d'une courbe en S

Une façon d'obtenir une courbe en S est de considérer

$$x \mapsto \frac{\exp(x'\beta)}{1+\exp(x'\beta)}$$



Le modèle de régression logistique

— Il propose de *modéliser la probabilité* $p(x)$ selon

$$p(x) = \frac{\exp(\beta_0 + \beta_1 x)}{1 + \exp(\beta_0 + \beta_1 x)}.$$

— On peut réécrire

$$\text{logit}\, p(x) = \log\left(\frac{p(x)}{1 - p(x)}\right) = \beta_0 + \beta_1 x.$$

Le modèle de régression logistique

Le modèle de régression logistique consiste donc à caractériser la loi de $Y|X = x$ par une loi de Bernoulli de paramètre $p(x)$ tel que

$$\text{logit}\, p(x) = \log\left(\frac{p(x)}{1 - p(x)}\right) = \beta_0 + \beta_1 x.$$

Exemple sur R

```
> model <- glm(chd~age,data=artere,family=binomial)
> model

Call:  glm(formula = chd ~ age, family = binomial, data = artere)

Coefficients:
(Intercept)          age
    -5.3095       0.1109

Degrees of Freedom: 99 Total (i.e. Null);  98 Residual
Null Deviance:     136.7
Residual Deviance: 107.4  AIC: 111.4
```

— La fonction *glm* renvoie les estimations de β_0 et β_1.

— On peut ainsi avoir une estimation de la *probabilité d'avoir une maladie pour un individu de 30 ans* :

$$\hat{p}(x = 30) = \frac{\exp(-5.3095 + 0.1109 * 30)}{1 + \exp(-5.3095 + 0.1109 * 30)} \approx 0.12.$$

Modèles GLM

— La fonction sur R qui permet d'ajuster le *modèle logistique* est la fonction glm.

— Le modèle de régression logistique, tout comme le modèle linéaire, appartient à la famille des *modèles linéaires généralisés*.

— C'est pourquoi il faut spécifier l'argument *family=binomial* lorsque l'on veut faire une régression logistique.

Maladie cardiovasculaire

— Dans la quasi-totalité des cas pratiques on cherche à expliquer une variables Y par p *variables explicatives*.

— *Exemple* : on cherche à expliquer la présence/absence d'une maladie cardiovasculaire (`chd`) par *9 variables*. On dispose de $n = 462$ individus.

```
> data(SAheart,package="bestglm")
> head(SAheart)
  sbp tobacco  ldl adiposity famhist typea obesity alcohol age chd
1 160   12.00 5.73     23.11 Present    49   25.30   97.20  52   1
2 144    0.01 4.41     28.61  Absent    55   28.87    2.06  63   1
3 118    0.08 3.48     32.28 Present    52   29.14    3.81  46   0
4 170    7.50 6.41     38.03 Present    51   31.99   24.26  58   1
5 134   13.60 3.50     27.78 Present    60   25.99   57.34  49   1
```

Le modèle de régression logistique multiple

— On dispose d'une variable binaire Y et de p *variables explicatives* $X = (X_1, \ldots, X_p)$.

— On cherche toujours à modéliser *la loi de* $Y|X = x$. La seule chose qui change ici est que x est un vecteur de dimension p.

Le modèle de régression logistique multiple ([?])

La loi de $Y|X = x$ est caractérisée par une loi de Bernoulli de paramètre $p(x) = \mathbf{P}(Y = 1|X = x)$ tel que

$$\operatorname{logit} p(x) = \log\left(\frac{p(x)}{1-p(x)}\right) = \beta_0 + \beta_1 x_1 + \ldots + \beta_p x_p = x^t\beta$$

2 Estimation des paramètres

La vraisemblance

— La *vraisemblance* du modèle est une fonction des paramètres $\beta_1, \ldots, \beta_p$ définie par

$$L_n(\beta) = \prod_{i=1}^{n} \mathbf{P}(Y = y_i|X = x_i).$$

— Cette fonction "mesure" la *probabilité d'observer les données* que l'on a pour chaque valeur de β_0 et β_1.

— L'idée consiste à trouver les valeurs de β_0 et β_1 qui maximise cette probabilité (on parle *d'estimateurs du maximum de vraisemblance*).

Calcul de la vraisemblance

— Les variables aléatoires $Y_1, \ldots, Y_n$ étant *discrètes et indépendantes*, la vraisemblance du modèle logistique est définie par

$$\begin{aligned} L_n : \{0,1\}^n \times \mathbb{R}^p &\to \mathbb{R} \\ (y_1, \ldots, y_n, \beta) &\mapsto \prod_{i=1}^{n} \mathbf{P}(Y_i = y_i|X_i = x_i). \end{aligned}$$

— Pour simplifier, on notera $L_n(y_1, \ldots, y_n, \beta) = L_n(\beta)$ et $\mathcal{L}_n(\beta) = \log(L_n(\beta))$.

Propriété

$$\mathcal{L}_n(\beta) = \sum_{i=1}^{n} \{y_i x_i'\beta - \log(1 + \exp(x_i'\beta))\}.$$

Maximisation de la vraisemblance

— *Malheureusement* il n'existe pas de solutions explicites pour maximiser la vraisemblance.

— *Mais...* la vraisemblance possède généralement un maximum unique et il existe des algorithmes numériques itératifs qui permettent d'obtenir ce maximum :

  — algorithme de Newton-Raphson ;
  — algorithme du score de Fisher.

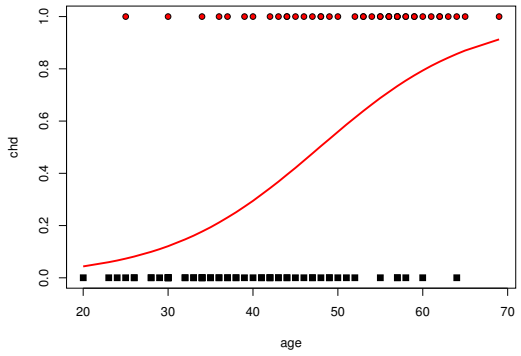
— On peut vérifier sur R si l'algorithme a bien *convergé*

```
> summary(model)
Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -5.30945    1.13365  -4.683 2.82e-06 ***
age          0.11092    0.02406   4.610 4.02e-06 ***
---
Number of Fisher Scoring iterations: 4
> model$converged
[1] TRUE
```

— Le *modèle ajusté* est donc

$$\hat{\mathbf{P}}(Y = 1|X = x) = \frac{\exp(-5.30945 + 0.11092x)}{1 + \exp(-5.30945 + 0.11092x)}.$$

Fonction estimée



Quand le coefficient β_j associé à la variable X_j est

— *positif* : X_j augmente $\rightarrow$ p augmente

— *négatif* : X_j augmente $\rightarrow$ p diminue

Ici, $\hat{\beta}_{age} = 0.11$, donc la probabilité augmente avec l'âge !

3 Propriétés des estimateurs

Comportement asymptotique des estimateurs

— Contrairement au modèle linéaire, *on ne connaît pas la loi des estimateurs* $\hat{\beta}_j$ pour le modèle logistique.

— Néanmoins, la théorie du maximum de vraisemblance ([**?**]) nous permet d'obtenir la *loi limite du vecteur aléatoire* $\hat{\beta}$:

$$\sqrt{n}(\hat{\beta} - \beta) \xrightarrow{\mathcal{L}} \mathcal{N}(0, \mathcal{I}(\beta)^{-1}).$$

Remarques

— $\mathcal{I}(\beta)$, matrice d'information de Fisher du modèle au point β ;

— Cette matrice est inconnue mais possibilité de "bien" l'estimer.

— En pratique, on fait l'approximation

$$\sqrt{n}(\hat{\beta} - \beta) \overset{\mathcal{L}}{\approx} \mathcal{N}(0, \hat{\mathcal{I}}(\hat{\beta})^{-1}).$$

Intervalles de confiance et tests

Loi de $\hat{\beta}_j$

On déduit du théorème précédent

$$\sqrt{n}\frac{\hat{\beta}_j - \beta_j}{\hat{\sigma}_j} \overset{\mathcal{L}}{\approx} \mathcal{N}(0,1),$$

où $\hat{\sigma}_j^2$ désigne le j^{e} terme de la diagonale de $\hat{\mathcal{I}}(\hat{\beta})$.

Applications :

— Intervalle de confiance de niveau $1-\alpha$ pour β_j :

$$\left[\hat{\beta}_j - q_{1-\alpha/2}\frac{\hat{\sigma}_j}{\sqrt{n}}; \hat{\beta}_j + q_{1-\alpha/2}\frac{\hat{\sigma}_j}{\sqrt{n}}\right].$$

— Tests : $H_0 : \beta_j = 0$ contre $H_1 : \beta_j \neq 0$.

L'exemple du chd

— Le *modèle* :

$$\log \frac{\mathbf{P}(chd = 1|age)}{1 - \mathbf{P}(chd = 1|age)} = \beta_0 + \beta_1 age.$$

— La *sortie* **R** :

```
> summary(model)$coefficients
             Estimate Std. Error   z value     Pr(>|z|)
(Intercept) -5.3094534 1.13365365 -4.683488 2.820338e-06
age          0.1109211 0.02405982  4.610224 4.022356e-06
```

au risque 5%, on rejette l'hypothèse $\beta_1 = 0$.

— *Intervalles de confiance* :

```
> confint(model)
Waiting for profiling to be done...
                 2.5 %     97.5 %
(Intercept) -7.72587162 -3.2461547
age          0.06693158  0.1620067
```

Test de nullité de q coefficients

— Tester la nullité d'un paramètre n'est *pas suffisant*.

1. Comment *tester la nullité de tous les paramètres* (à l'exception de la constante) ? Equivalent du test de Fisher en régression linéaire.
2. Comment tester l'*effet d'une variable explicative qualitative* ? Pour tester l'effet de la variable `marque`, on teste la nullité simultanée des coefficients du modèle associé à cette variable

Plusieurs paramètres

Nécessité de développer des procédures de tests permettant de tester des hypothèses du genre :

$$H_0 : \beta_1 = \ldots = \beta_q = 0 \quad \text{contre} \quad H_1 : \exists j \in \{1, \ldots, q\} : \beta_j \neq 0.$$

Test de Wald

— Il est basé sur le résultat :

$$(\hat{\beta}_n - \beta)'\hat{\Sigma}(\hat{\beta}_n - \beta) \xrightarrow{\mathcal{L}} \chi^2_p.$$

— On désigne par $\hat{\beta}_n^{(q)}$ les q premières composantes de $\hat{\beta}_n$ et $\hat{\Sigma}^{(q)}$ la matrice $q \times q$ comprenant les q premières lignes et colonnes de $\hat{\Sigma}$. On a alors :

$$(\hat{\beta}_n^{(q)} - \beta^{(q)})'\hat{\Sigma}^{(q)}(\hat{\beta}_n^{(q)} - \beta^{(q)}) \xrightarrow{\mathcal{L}} \chi^2_q.$$

— On déduit que *sous* H_0

$$\hat{\beta}^{(q)}\hat{\Sigma}^{(q)}\hat{\beta}^{(q)} \xrightarrow{\mathcal{L}} \chi^2_q.$$

— On rejette l'hypothèse nulle si la valeur observée de la statistique de test ci dessus est supérieure au quantile d'ordre $1-\alpha$ de la loi χ^2_q.

Test de déviance ou du rapport de vraisemblance

— *Idée :* on note $\hat{\beta}_{H_0}$ l'emv contraint sous H_0. Si H_0 est vraie, on doit avoir

$$\hat{\beta}_{H_0} \approx \hat{\beta}_n \quad \text{et} \quad \mathcal{L}_n(\hat{\beta}_{H_0}) \approx \mathcal{L}_n(\hat{\beta}_n).$$

— Plus précisément, on montre que *sous* H_0,

$$2(\mathcal{L}_n(\hat{\beta}_n) - \mathcal{L}_n(\hat{\beta}_{H_0})) \xrightarrow{\mathcal{L}} \chi^2_q.$$

— On rejette l'hypothèse nulle si la valeur observée de la statistique de test ci dessus est supérieure au quantile d'ordre $1-\alpha$ de la loi χ^2_q.

Test du score

— *Idée :* on note $\hat{\beta}_{H_0}$ l'emv contraint sous H_0. Si H_0 est vraie, on doit avoir $S(\hat{\beta}_{H_0}) = \nabla\mathcal{L}_n(\hat{\beta}_0) \approx 0$.

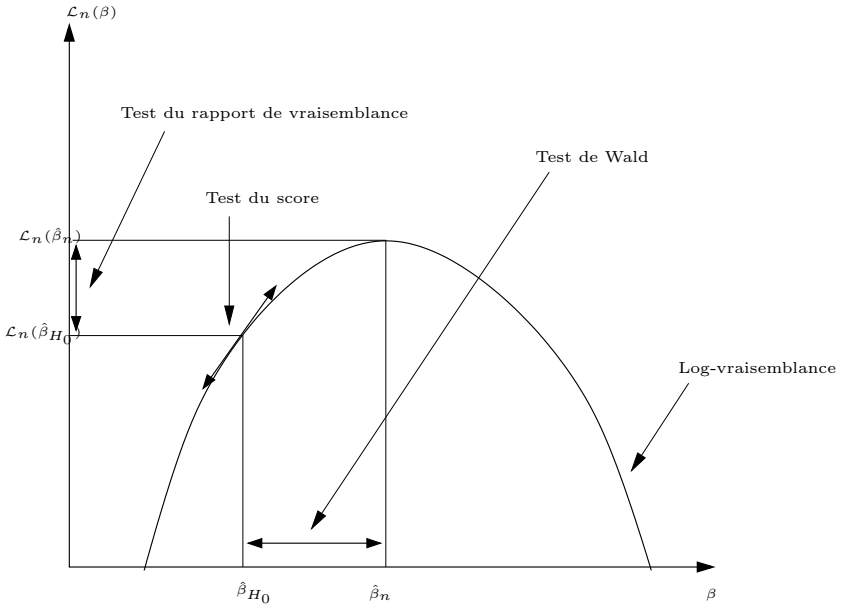
— Plus précisément, on montre que *sous* H_0,

$$S(\hat{\beta}_{H_0})'\hat{\Sigma}_{H_0}^{-1}S(\hat{\beta}_{H_0}) \xrightarrow{\mathcal{L}} \chi^2_q,$$

où $\hat{\Sigma}_{H_0} = \mathbb{X}W_{\hat{\beta}_{H_0}}\mathbb{X}$.

— On rejette l'hypothèse nulle si la valeur observée de la statistique de test ci dessus est supérieure au quantile d'ordre $1-\alpha$ de la loi χ^2_q.

Récaptitulatif



Exemple sous R

On peut utiliser la fonction *Anova* du package car :

1. *Test de Wald* :

```
> library(car)
> Anova(model,type=3,test.statistic="Wald")
            Df  Chisq Pr(>Chisq)
(Intercept)  1 0.3294     0.5660
age          1 0.0218     0.8826
marque       2 1.9307     0.3809
Residuals   29
```

2. *Test du rapport de vraisemblance* :

```
> Anova(model,type=3,test.statistic="LR")
       LR Chisq Df Pr(>Chisq)
age     0.02189  1     0.8824
marque  2.09562  2     0.3507
```

SAS

— Sous SAS, on utilise la *proc logistic*

```
proc logistic data=Tp1_panne descending;
class marque;
model panne= age marque;
run;
```

06:50 mardi, janvier 14, 2014 **1**

Le Système SAS

Procédure LOGISTIC

Estimations par l'analyse du maximum de vraisemblance						
Paramètre		**DDL**	**Valeur estimée**	**Erreur type**	**Khi-2 de Wald**	**Pr > Khi-2**
Intercept		1	-0.1471	0.6265	0.0551	0.8144
age		1	0.0139	0.0940	0.0218	0.8826
marque	**0**	1	0.6252	0.5344	1.3684	0.2421
marque	**1**	1	0.2058	0.4907	0.1758	0.6750

Estimations des rapports de cotes			
Effet	**Valeur estimée du point**	**Intervalle de confiance de Wald à 95 %**	
age	1.014	0.843	1.219
marque 0 vs 3	4.289	0.544	33.820
marque 1 vs 3	2.820	0.407	19.544

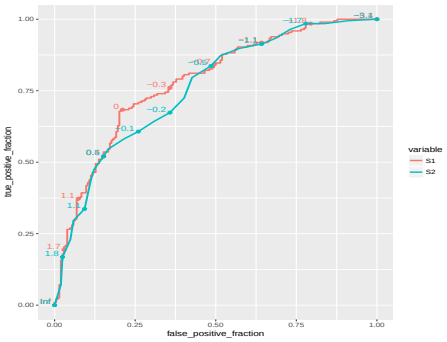
Le Système SAS

Procédure LOGISTIC

Statistiques d'ajustement du modèle		
Critère	**Constante uniquement**	**Constante et covariables**
AIC	47.717	51.502
SC	49.214	57.488
-2 Log	45.717	43.502

Test de l'hypothèse nulle globale : BETA=0			
Test	**Khi-2**	**DDL**	**Pr > Khi-2**
Rapport de vrais	2.2152	3	0.5290
Score	2.1630	3	0.5393
Wald	2.0333	3	0.5655

Analyse des effets Type 3			
Effet	**DDL**	**Khi-2 de Wald**	**Pr > Khi-2**
age	1	0.0218	0.8826
marque	2	1.9306	0.3809



4 Discrétisation des variables explicatives

— Les variables continues sont *souvent (tout le temps) discrétisées* avant une régression logistique.

```
> summary(dapp)
       X1                  X2             Y
 Min.   :-3.39606   Min.   :-3.080366   0:312
 1st Qu.:-0.69551   1st Qu.:-0.656398   1:288
 Median :-0.02684   Median :-0.014737
 Mean   :-0.02128   Mean   : 0.003285
 3rd Qu.: 0.62586   3rd Qu.: 0.633525
 Max.   : 3.19590   Max.   : 2.585601

> summary(dapp1)
             X11                    X22       Y
 [-10,-0.849]   :120   [-10,-0.849]   :112   0:312
 (-0.849,-0.285]:124   (-0.849,-0.285]:122   1:288
 (-0.285,0.193] :107   (-0.285,0.193] :116
 (0.193,0.761]  :128   (0.193,0.761]  :122
 (0.761,10]     :121   (0.761,10]     :128
```

Ajustement des modèles

On ajuste le modèle logistique sur les *données brutes*, puis sur les *variables discrétisées*.

```
> m1
Coefficients:
(Intercept)           X1           X2
   -0.08301      0.93426      1.10734
> m2
Coefficients:
       (Intercept)  X11(-0.849,-0.285]   X11(-0.285,0.193]    X11(0.193,0.761]
           -3.0673              0.9280              0.9095              1.5994
    X11(0.761,10]  X22(-0.849,-0.285]   X22(-0.285,0.193]    X22(0.193,0.761]
            2.5822              1.3961              1.9471              1.8978
    X22(0.761,10]
            3.2447
```

Discussion

Pourquoi discrétiser ?

— Avantage : prise en compte d'effets non linéaires.

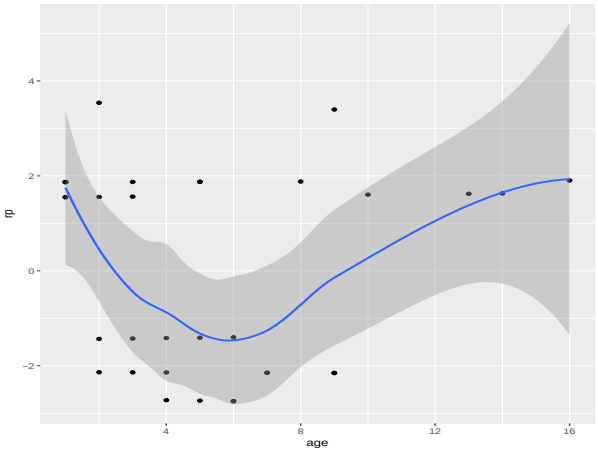
— Inconvénients :

  — *Perte d'information* et de *performance* si l'effet de la variable est linéaire.

  — *Pas de règle optimale* pour discrétiser : les approches classiques sont basées sur des analyses bivariées. Aucune garantie de leur validité dans un contexte multivarié.

  — *Augmentation de la complexité* du modèle (plus de paramètres à estimer).

```
> S1 <- predict(m1,newdata=dtest); S2 <- predict(m2,newdata=dtest1)
> tab.score <- data.frame(S1,S2,Y=as.numeric(dtest$Y)-1)
> tab.score1 <- melt(tab.score,"Y")
> library(plotROC)
> ggplot(tab.score1)+aes(d=Y,m=value,color=variable)+geom_roc()
```




```
> library(pROC)
> auc(dtest$Y,S1)
Area under the curve: 0.78
> auc(dtest$Y,S2)
Area under the curve: 0.7568
```

Une solution : les résidus partiels

— On considère le modèle logistique

$$\text{logit}\, p_\beta(x) = \beta_1 x_1 + \ldots, \beta_p x_p.$$

Définition

Les *résidus partiels* sont définis par :

$$r_{ij} = \frac{Y_i - p_{\hat{\beta}_n}(x_i)}{p_{\hat{\beta}_n}(x_i)(1 - p_{\hat{\beta}_n}(x_i))} + \hat{\beta}_j x_{ij}, \quad i = 1, \ldots n, j = 1 \ldots p.$$

Diagnostic

— L'analyse consiste à tracer pour toutes les variables j les T résidus $r_{ij}, i = 1, \ldots n$.

— Si le tracé est linéaire alors tout est "normal". Si par contre une tendance non linéaire se dégage, il faut remplacer la variable j par une fonction de celle ci donnant la même tendance que celle observée.

Un exemple : les données panne

```
> model <- glm(etat~.,data=panne,family=binomial)
> summary(model)
Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  0.47808    0.83301   0.574    0.566
age          0.01388    0.09398   0.148    0.883
marqueB     -0.41941    0.81428  -0.515    0.607
marqueC     -1.45608    1.05358  -1.382    0.167
```

Remarque

On accepte la nullité du coefficient age.

```
> rp <- residuals(model,type="partial")
> df <- data.frame(age=panne$age,rp=rp[,1])
> ggplot(df)+aes(x=age,y=rp)+geom_point()+geom_smooth()
```

Conclusion

Le graphe suggère d'ajouter la variable $\texttt{age}^2$ dans le modèle.

```
> model1 <- glm(etat~age+I(age^2),data=panne,family=binomial)
> summary(model1)
Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  4.18501    1.73860   2.407  0.01608 *
age         -2.03343    0.77401  -2.627  0.00861 **
I(age^2)     0.17601    0.07044   2.499  0.01247 *
```

Remarque

On rejette maintenant la nullité du coefficient age.

5 Sélection de modèle logistique

Limites

— Principalement *2 motifs d'insatisfaction* :

1. Précision d'estimation : les estimateurs des MCO pour la régression et du MV pour la logistique ont souvent un biais relativement faible mais une variance élevée (notamment lorsque le nombre de variables d est grand).

2. Interprétation : lorsque le nombre de variables d est grand, on ne connait pas les variables "importantes".

Objectifs

— Avec l'*augmentation du volume des données* ces dernières années, ces deux *inconvénients* sont de *plus en plus visibles*.

— Nécessité de développer des procédures de *sélection de sous-groupes de variables*.

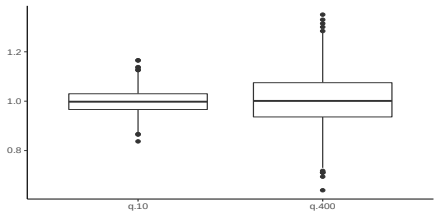
Un exemple

— On génère des données $(x_i, y_i), i = 1, \ldots, 500$ selon le modèle

$$\text{logit}\, p_\beta(x) = 1x_1 + 0x_2 + \ldots + 0x_{q+1}$$

où $X_1, \ldots, X_{q+1}$ sont i.i.d. de loi $\mathcal{N}(0, 1)$.

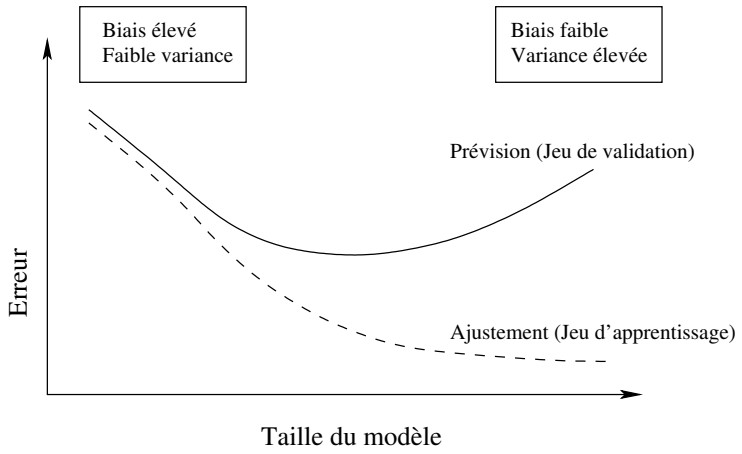
— On calcule l'*estimateur du MV de* β_1 sur 1000 répétitions. On trace les boxplot de ces estimateurs pour $q = 10$ et $q = 400$.



Conclusion

Plus de variance (donc moins de précisions) lorsque le nombre de variables inutiles augmente.

Taille de modèle



Idem erreur d'estimation (variance) / erreur d'approximation (biais).

Quelques questions

Comment comparer des modèles ?

— Nécessité de définir des critères mais...
— Il n'existe pas de critère universel permettant de définir la notion de meilleur modèle.
— Nous proposons quelques exemples de critères dans la suite.

Comment choisir les meilleurs variables parmi $X_1, \ldots, X_p$ ***?***

— Nécessité de proposer des algorithmes.
— Recherche exhaustive vs pas à pas.

5.1 Critères de choix de modèles

AIC-BIC

— *Idée* : mesurer la qualité d'ajustement en utilisant la vraisemblance.

Problème

Si $\mathcal{M}_1 \subset \mathcal{M}_2$ alors $\mathcal{L}_n(\hat{\beta}_1) \leq \mathcal{L}_n(\hat{\beta}_2)$ où $\hat{\beta}_j$ désigne l'emv du modèle $\mathcal{M}_j, j = 1, 2$.

— *Conséquence* : la vraisemblance sélectionnera toujours le modèle le plus complexe.

Solution

Pénaliser la vraisemblance par la complexité du modèle.

Définition

Soit $\mathcal{M}$ un modèle logistique à p paramètres. On note $\hat{\beta}_n$ l'emv des paramètres du modèle.
— L'*AIC (Akaike Information Criterion)* du modèle $\mathcal{M}$ est défini par

$$AIC(\mathcal{M}) = -2\mathcal{L}_n(\hat{\beta}_n) + 2p.$$

— Le *BIC (Bayesian Information Criterion)* du modèle $\mathcal{M}$ est défini par

$$BIC(\mathcal{M}) = -2\mathcal{L}_n(\hat{\beta}_n) + p \log n.$$

Conclusion

— Le modèle retenu sera celui qui minimise l'AIC ou le BIC.
— $\log n > 2$ (pour $n \geq 8$) BIC aura tendance a choisir des modèles plus parcimonieux que AIC.

Exemple

— On considère les mêmes modèles que precédemment :

```
> model1 <- glm(chd~tobacco+famhist,data=SAheart,family=binomial)
> model2 <- glm(chd~tobacco+famhist+adiposity+alcohol,data=SAheart,
                                                    family=binomial)
```

— On les compare en terme d'AIC et de BIC.

```
> c(AIC(model1),AIC(model2))
[1] 530.5759 523.4657
> c(BIC(model1),BIC(model2))
[1] 542.9826 544.1436
```

Conclusion

AIC sélectionne `model2` tandis que BIC sélectionne `model1`.

5.2 Sélection de variables

Motivations

— Dans la partie précédente, on a présenté des outils permettant de comparer des modèles *construits*.

— On se place dans un cadre différent : étant donné p variables explicatives $X_1, \ldots, X_p$, on cherche une procédure automatique permettant de trouver le *"meilleur" sous-groupe de variables* à mettre dans le modèle logistique.

Pourquoi ?

(Au moins) 2 raisons peuvent motiver cette démarche :

1. Descriptif : identifier les variables qui permettent d'expliquer la cible.
2. Statistique : la variance des estimateurs augmente avec le nombre de paramètres du modèle. Diminuer le nombre de variables permettra d'avoir des estimateurs plus précis.

Recherche exhaustive

— Une approche naturelle est de construire *tous* les modèles logistiques (2^p) et de retenir celui qui *optimise un critère donné* (AIC-BIC...).

— Les package leaps permet de faire cela pour la *régression linéaire*.

— Pour le *modèle logistique*, on peut utiliser le package bestglm.

```
> library(bestglm)
> model4 <- bestglm(dapp,family=binomial,IC="BIC")
Morgan-Tatar search since family is non-gaussian.
> model4$BestModel

Call:  glm(formula = y ~ ., family = family, data = Xi, weights = weights)

Coefficients:
   (Intercept)            ldl  famhistPresent            age
      -4.29645        0.18650        0.82172        0.05088

Degrees of Freedom: 249 Total (i.e. Null);  246 Residual
Null Deviance:     319.2
Residual Deviance: 267.5  AIC: 275.5
```

— On peut également visualiser les *variables retenues dans les meilleurs modèles* pour le critère donné

```
> model4$BestModels
    sbp tobacco   ldl adiposity famhist typea obesity alcohol  age Criterion
1 FALSE   FALSE  TRUE     FALSE    TRUE FALSE   FALSE   FALSE TRUE  284.0427
2 FALSE   FALSE  TRUE     FALSE   FALSE FALSE   FALSE   FALSE TRUE  286.0520
3 FALSE    TRUE  TRUE     FALSE    TRUE FALSE   FALSE   FALSE TRUE  286.7856
4 FALSE   FALSE FALSE     FALSE    TRUE FALSE   FALSE   FALSE TRUE  287.3270
5 FALSE   FALSE  TRUE     FALSE    TRUE  TRUE   FALSE   FALSE TRUE  287.9329
```

Remarque

Lorsque le nombre de variables p est trop grand, balayer tous les modèles peut se révéler très couteux en tant de calcul. On a alors recours à des méthodes pas à pas.

Méthode pas à pas

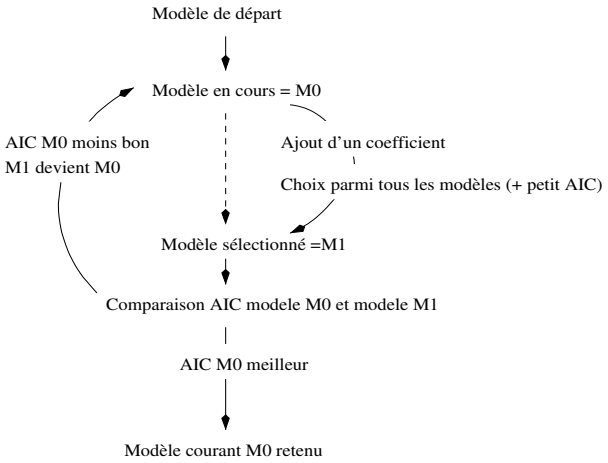
L'approche consiste à :

— construire un *modèle initial*

— Ajouter (*forward*) ou supprimer (*backward*) la variable qui optimise un critère donné (*BIC* ou *AIC*) par exemple.

— Répéter le processus jusqu'à un *critère d'arrêt*.

Technique ascendante utilisant l'AIC



Exemple sur R

— La fonction step permet de sélectionner des variables à l'aide de méthodes *pas à pas*.

```
> model_complet <- glm(chd~.,data=dapp,family=binomial)
> model_step <- step(model_complet,direction="backward",k=log(nrow(dapp)))
> model_step

Call:  glm(formula = chd ~ ldl + famhist + age, family = binomial,
                                                    data = dapp)

Coefficients:
   (Intercept)            ldl  famhistPresent            age
      -4.29645        0.18650        0.82172        0.05088
```

Quatrième partie

Arbres

Présentation

— Les arbres sont des algorithmes de prédiction qui fonctionnent en *régression et en discrimination*.

— Il existe *différentes variantes* permettant de construire des prédicteurs par arbres.

— Nous nous focalisons dans cette partie sur la *méthode CART* [Breiman et al., 1984] qui est la plus utilisée. La méthode CHAID est proposée en *annexe*.

1 Arbres binaires

Notations

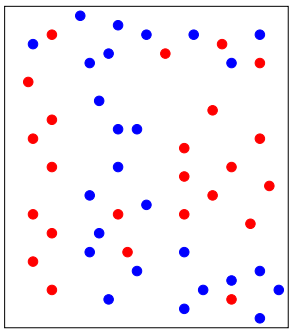
— On cherche à *expliquer une variable* Y par p *variables explicatives* $X_1, \ldots, X_p$.

— Y peut admettre un nombre quelconque de modalités et les variables $X_1, \ldots, X_p$ peuvent être *qualitatives et/ou quantitatives*.

— Néanmoins, pour simplifier on se place dans un premier temps en *discrimination binaire* : Y admet 2 modalités (-1 ou 1). On suppose de plus que l'on a simplement 2 variables explicatives quantitatives.

Représentation des données

— On dispose de n obervations $(X_1, Y_1), \ldots, (X_n, Y_n)$ où $X_i \in \mathbb{R}^2$ et $Y_i \in \{-1, 1\}$.



Approche par arbres

Trouver une partition des observations qui *sépare* "au mieux" les points rouges des points bleus.

Définitions

Arbre binaire

Un *arbre binaire de décision* CART est

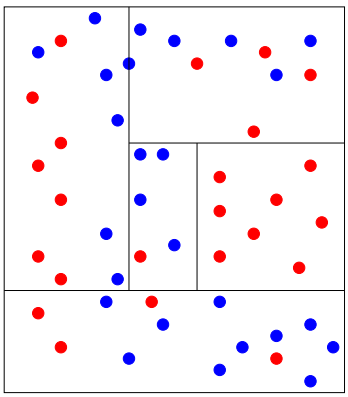
— un algorithme de *moyennage local* par partition (moyenne ou vote à la majorité sur les éléments de la partition),

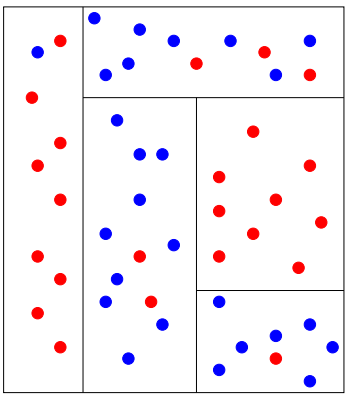
— dont la partition est construite par *divisions successives* au moyen *d'hyperplans orthogonaux aux axes* de $\mathbb{R}^p$, dépendant des données (X_i, Y_i).

Arbres binaires

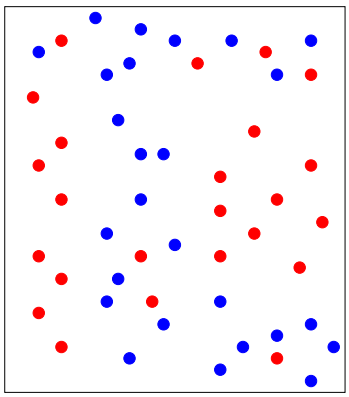
— La *méthode CART* propose de construire une partition basée sur des divisions *successives parallèles aux axes.*

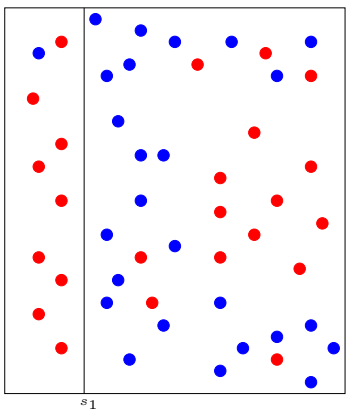
— 2 exemples de partition :

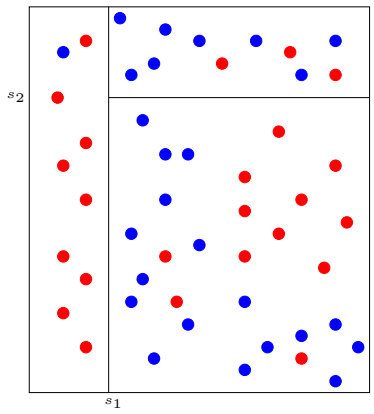


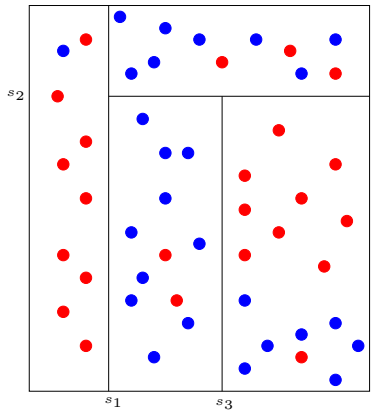


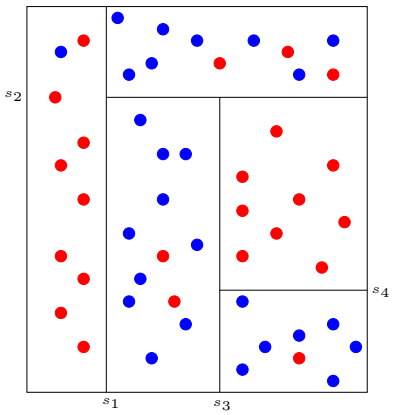
— A chaque étape, la méthode cherche une *nouvelle division* : une variable et un seuil de coupure.



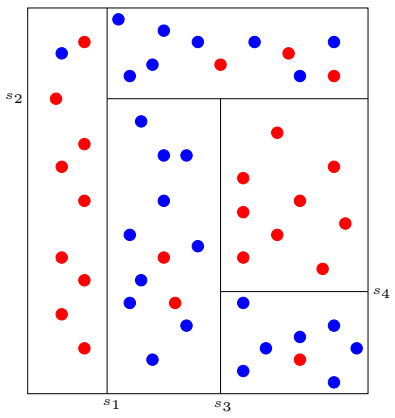


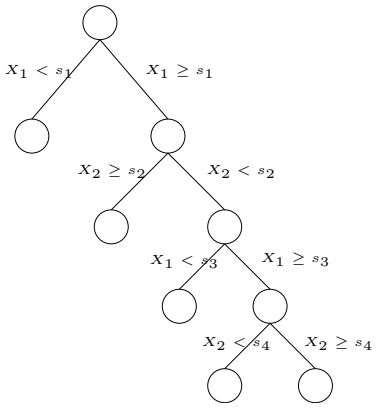


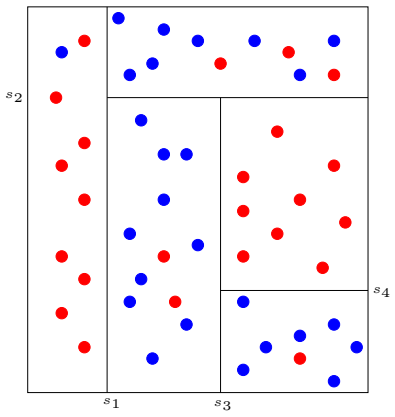


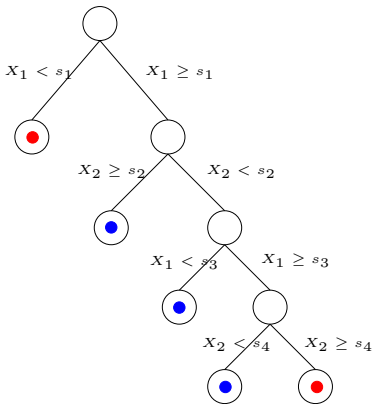


Représentation de l'arbre









Règle de classification

On effectue un vote à la majorité dans les nœuds terminaux de l'arbre.

Définitions

Définition

— Les éléments de la partition d'un arbre sont appelés les *nœuds terminaux* ou les *feuilles* de l'arbre.
— L'ensemble $\mathbb{R}^p$ constitue le *nœud racine*.
— Chaque division définit deux nœuds, les *nœuds fils à gauche et à droite*.

2 Choix des découpes

Questions

1. Comment choisir les découpes ?
2. Faut-il stopper les découpes ? Si oui, quand ?

— A chaque étape, on cherche un *couple* (j, s) qui split un noeud $\mathcal{N}$ en deux nœuds fils :

$$\mathcal{N}_1(j, s) = \{X \in \mathcal{N} | X_j \leq s\} \quad et \quad \mathcal{N}_2(j, s) = \{X \in \mathcal{N} | X_j > s\}.$$

— La sélection du couple (j, s) s'effectue en optimisant un critère qui mesure l'*(im)pureté* ou l'*hétérogénité* des deux nœuds fils.

Critère de découpe

— L'*impureté* $\mathcal{I}$ d'un nœud doit être :
  1. faible lorsque un nœud est homogène : les valeurs de Y dans le nœud sont *proches*.
  2. élevée lorsque un nœud est hétérogène : les valeurs de Y dans le nœud sont *dispersés*.

L'idée

Une fois $\mathcal{I}$ défini, on choisira le couple (j, s) qui *maximise le gain d'impureté* :

$$\Delta(\mathcal{I}) = \mathbf{P}(\mathcal{N})\mathcal{I}(\mathcal{N}) - (\mathbf{P}(\mathcal{N}_1)\mathcal{I}(\mathcal{N}_1(j, s)) + \mathbf{P}(\mathcal{N}_2)\mathcal{I}(\mathcal{N}_2(j, s)).$$

2.1 Cas de la régression

— Une mesure naturelle de l'*impureté* d'un nœud $\mathcal{N}$ en *régression* est la variance du nœud :

$$\mathcal{I}(\mathcal{N}) = \frac{1}{|\mathcal{N}|} \sum_{i: X_i \in \mathcal{N}} (Y_i - \bar{Y}_{\mathcal{N}})^2,$$

où $\bar{Y}_{\mathcal{N}}$ désigne la moyenne des Y_i dans $\mathcal{N}$.

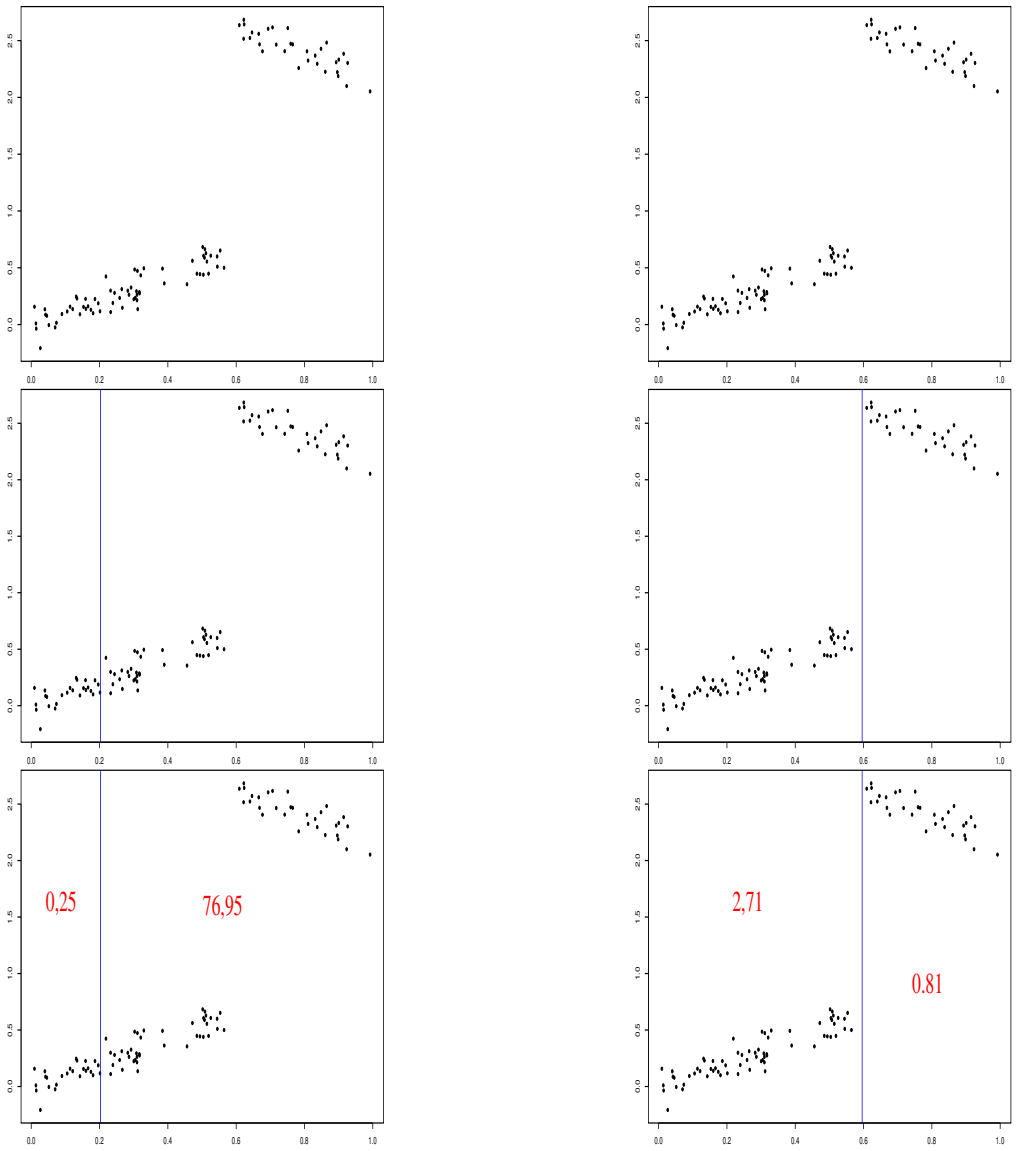
Découpe en régression

A chaque étape, on choisit le couple (j, s) qui minimise

$$\sum_{X_i \in \mathcal{N}_1(j,s)} (Y_i - \bar{Y}_1)^2 + \sum_{X_i \in \mathcal{N}_2(j,s)} (Y_i - \bar{Y}_2)^2$$

où $\bar{Y}_k = \frac{1}{|\mathcal{N}_k(j,s)|} \sum_{X_i \in \mathcal{N}_k(j,s)} Y_i, k = 1, 2.$

Exemple



Sélection

On choisira le seuil de droite.

2.2 Cas de la classification supervisée

— Les $Y_i, i = 1, \ldots, n$ sont à valeurs dans $\{1, \ldots, K\}$.

— On cherche une fonction $\mathcal{I}$ telle que $\mathcal{I}(\mathcal{N})$ soit

  — *petite* si un *label majoritaire* se distingue clairement dans $\mathcal{N}$;

  — *grande* sinon.

Impureté

L'impureté d'un nœud $\mathcal{N}$ en classification se mesure selon

$$\mathcal{I}(\mathcal{N}) = \sum_{j=1}^{K} f(p_j(\mathcal{N}))$$

où

— $p_j(\mathcal{N})$ représente la proportion d'observations de la classe j dans le nœud $\mathcal{N}$.

— f est une fonction (concave) $[0, 1] \to \mathbb{R}^+$ telle que $f(0) = f(1) = 0$.

Exemples de fonctions f

— Si $\mathcal{N}$ *est pur*, on veut $\mathcal{I}(\mathcal{N}) = 0 \Longrightarrow$ c'est pourquoi f doit vérifier $f(0) = f(1) = 0$.

— Les 2 mesures d'impureté les plus classiques sont :

1. *Gini* : $f(p) = p(1 - p)$;
2. *Information* : $f(p) = -p \log(p)$.

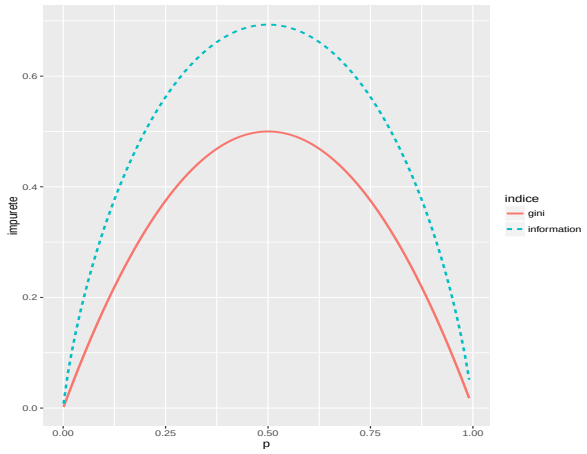
Cas binaire

Dans ce cas on a

1. $\mathcal{I}(\mathcal{N}) = 2p(1 - p)$ pour Gini
2. $\mathcal{I}(\mathcal{N}) = -p \log p - (1 - p) \log(1 - p)$ pour Information

où p désigne la proportion de 1 (ou -1) dans $\mathcal{N}$.

Impureté dans le cas binaire



Découpe en classification supervisée

— On rappelle que pour un nœud $\mathcal{N}$ donné et un couple (j, s), on note

$$\mathcal{N}_1(j,s) = \{X \in \mathcal{N} | X_j \leq s\} \quad et \quad \mathcal{N}_2(j,s) = \{X \in \mathcal{N} | X_j > s\}.$$

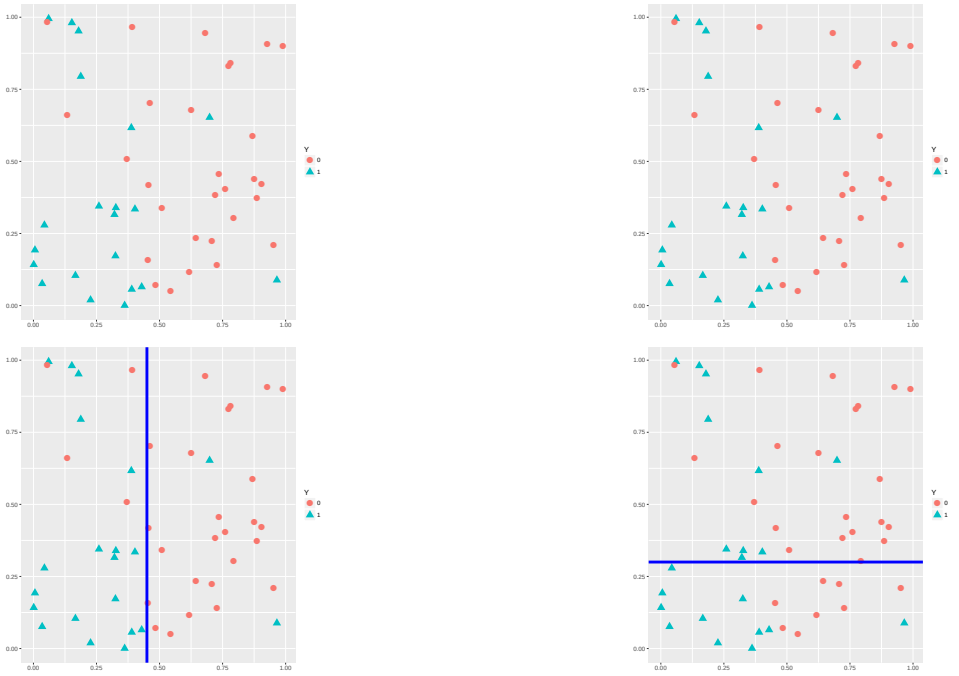
Choix de (j, s)

Pour une mesure d'impureté $\mathcal{I}$ donnée, on choisira le couple (j, s) qui maximise le gain d'impureté :

$$\Delta(\mathcal{I}) = \mathbf{P}(\mathcal{N})\mathcal{I}(\mathcal{N}) - (\mathbf{P}(\mathcal{N}_1)\mathcal{I}(\mathcal{N}_1(j,s)) + \mathbf{P}(\mathcal{N}_2)\mathcal{I}(\mathcal{N}_2(j,s)).$$

Exemple

$$\mathcal{I}(\mathcal{N}) = 0.4872$$



	$\mathcal{I}(\mathcal{N}_1)$	$\mathcal{I}(\mathcal{N}_2)$	$\Delta(\mathcal{I})$
Gauche	0.287	0.137	*0.281*
Droite	0.488	0.437	0.031

Conclusion

On choisira la découpe de gauche.

3 Elagage

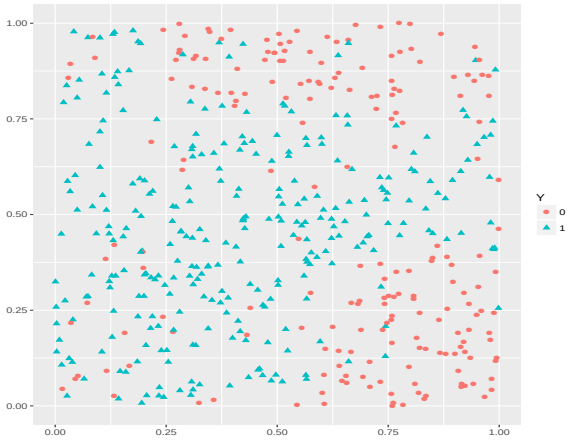
Questions

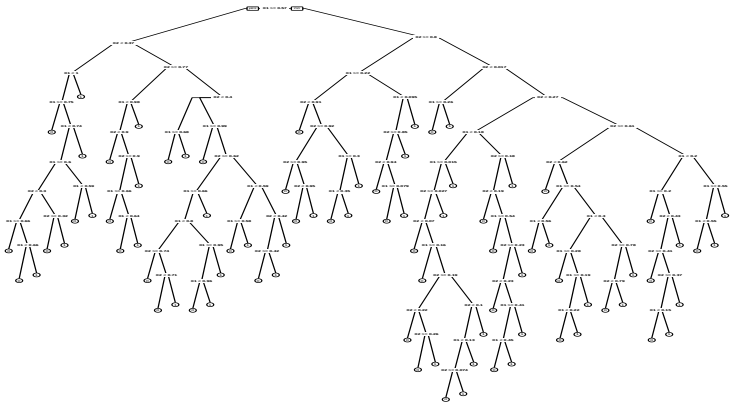
— Comment construire un *"bon" arbre* ?

— Construire l'arbre *maximal* ? (on découpe les nœuds jusqu'à ce qu'on ne puisse plus).

— Faut-il se donner un *critère d'arrêt* ?

— Faut-il construire un arbre grand et choisir un *sous-arbre* de ce dernier ?





Un exemple en discrimination

Arbre optimal ?

Intuitivement, on a envie de faire à peu près *5 classes*.

Arbre « maximal »

```
> library(rpart)
> library(rpart.plot)
> arbre1 <- rpart(Y~.,data=donnees,cp=0.0001,minsplit=2)
> prp(arbre1)
```

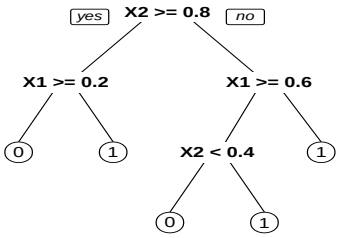
Un arbre plus petit

```
> arbre2 <- rpart(Y~.,data=donnees)
> prp(arbre2)
```

Comparaison des deux arbres

— On compare les performances des deux arbres en estimant leur *probabilité de mauvais classement* sur un échantillon test :

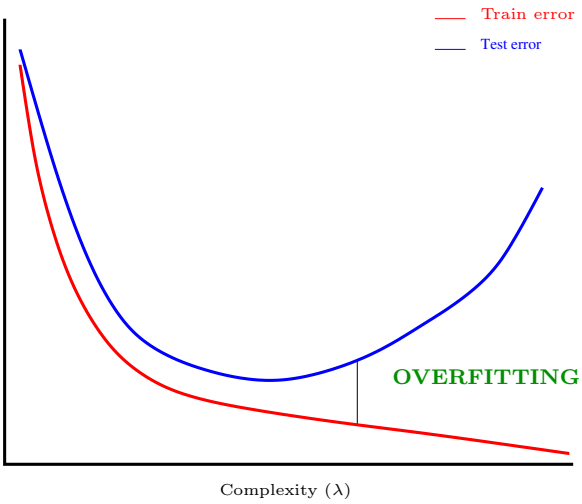
```
> prev1 <- predict(arbre1,newdata=dtest,type="class")
> prev2 <- predict(arbre2,newdata=dtest,type="class")
> round(mean(prev1!=dtest$Y),3)
[1] 0.157
> round(mean(prev2!=dtest$Y),3)
[1] 0.115
```



Conclusion

La performance *n'augmente pas forcément avec la profondeur.*

Sur-ajustement pour les arbres



Remarque

La *complexité* d'un arbre est mesurée par sa taille ou profondeur.

Biais et variance

La profondeur régule le compromis biais/variance :

1. Peu de découpes (arbres peu profonds) $\Longrightarrow$ arbres stables $\Longrightarrow$ peu de variance... mais... *beaucoup de biais.*
2. Beaucoup de découpes (arbres profonds) $\Longrightarrow$ arbres instables $\Longrightarrow$ peu de biais... mais... *beaucoup de variance (surapprentissage).*

Principe d'élagage [Breiman et al., 1984]

Plutôt que de choisir « quand couper » on raisonne en 3 temps :

1. On construit un *arbre maximal* (très profond) $\mathcal{T}_{max}$;
2. On sélectionne une *suite d'arbres emboités* :

$$\mathcal{T}_{max} = \mathcal{T}_0 \supset \mathcal{T}_1 \supset \ldots \supset \mathcal{T}_K.$$

3. On *sélectionne un arbre* dans cette sous-suite.

Arbres emboîtés

Construction de la suite de sous arbres

— Soit T un arbre à $|T|$ nœuds terminaux $\mathcal{N}_1, \ldots, \mathcal{N}_{|T|}$.
— Soit $R(\mathcal{N})$ le risque (l'erreur) dans le nœud $\mathcal{N}$:
  — *Régression* :

$$R(\mathcal{N}) = \frac{1}{|\mathcal{N}|} \sum_{i:X_i \in \mathcal{N}} (Y_i - \bar{Y}_{\mathcal{N}})^2.$$

  — *Classification binaire* :

$$R(\mathcal{N}) = \frac{1}{|\mathcal{N}|} \sum_{i:X_i \in \mathcal{N}} \mathbf{1}_{Y_i \neq Y_{\mathcal{N}}}.$$

Définition

Soit $\alpha > 0$, on pose

$$C_\alpha(T) = \sum_{m=1}^{|T|} N_m R(\mathcal{N}_m) + \alpha |T|.$$

Idée

— $C_\alpha(T)$ est un critère qui prend en compte l'adéquation d'un arbre et sa complexité.
— L'idée est de chercher un arbre T_α qui minimise $C_\alpha(T)$ pour une valeur de α bien choisie.

Remarque

— $\alpha = 0 \Longrightarrow T_\alpha = T_0 = T_{max}$.
— $\alpha = +\infty \Longrightarrow T_\alpha = T_{+\infty}$ =*arbre sans coupure.*
— α est appelé *paramètre de complexité* et $C_\alpha(T)$ le *cout* de l'arbre T.

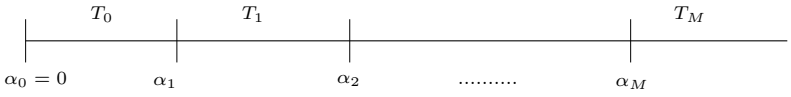
[

Théorème[Breiman et al., 1984]] Il existe une sous-suite finie $\alpha_0 = 0 < \alpha_1 < \ldots < \alpha_M$ avec $M < |T_{max}|$ et une suite associée d'*arbres emboités*

$$T_{max} = T_0 \supset T_1 \supset \ldots \supset T_M$$

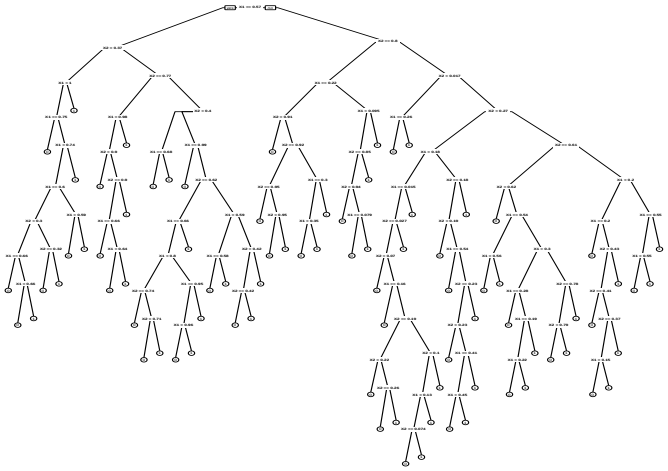
telles que $\forall \alpha \in [\alpha_m, \alpha_{m+1}[$

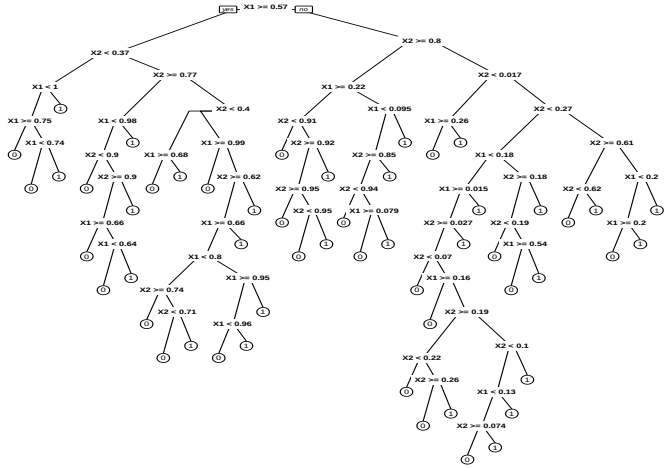
$$T_m = \underset{T}{\operatorname{argmin}}\, C_\alpha(T).$$

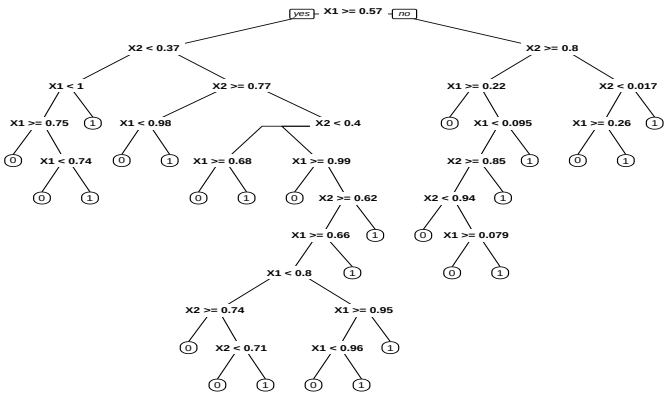


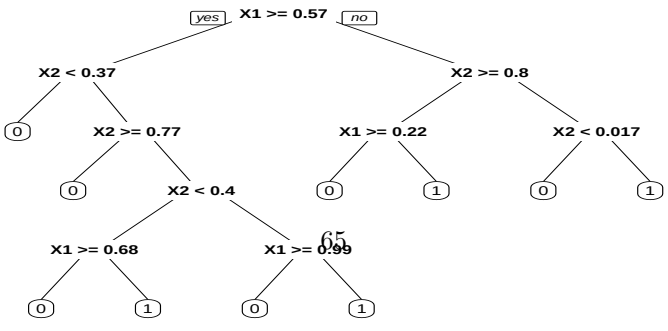
Conséquences

— On se ramène à une sous-suite finie d'arbres (emboités).
— Il reste à choisir un arbre (ou une valeur de α).




yes
X1 >= 0.57
no
X2 < 0.37
X2 >= 0.8
X1 < 1
X2 >= 0.77
X1 >= 0.22
X2 < 0.017
X1 >= 0.75
X1 < 0.98
X2 < 0.4
X2 < 0.91
X1 < 0.095
X1 >= 0.26
X2 < 0.27
X1 < 0.74
X2 < 0.9
X1 >= 0.68
X1 >= 0.99
X2 >= 0.92
X2 >= 0.85
X1 < 0.18
X2 >= 0.61
X2 >= 0.9
X2 >= 0.62
X2 >= 0.95
X2 < 0.94
X1 >= 0.015
X2 >= 0.18
X2 < 0.62
X1 < 0.2
X1 >= 0.66
X1 >= 0.66
X2 < 0.95
X1 >= 0.079
X2 >= 0.027
X2 < 0.19
X1 >= 0.2
X1 < 0.64
X1 < 0.8
X2 < 0.07
X1 >= 0.54
X1 >= 0.95
X2 >= 0.74
X1 >= 0.16
X2 < 0.71
X2 >= 0.19
X1 < 0.96
X2 < 0.1
X2 < 0.22
X2 >= 0.26
X1 < 0.13
X2 >= 0.074

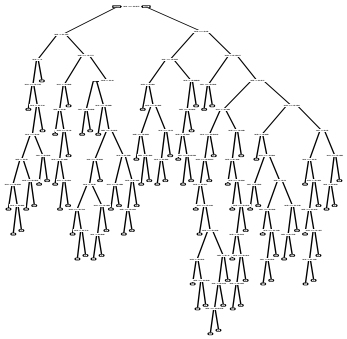

yes
X1 >= 0.57
no
X2 < 0.37
X2 >= 0.8
X1 < 1
X2 >= 0.77
X1 >= 0.22
X2 < 0.017
X1 >= 0.75
X1 < 0.98
X2 < 0.4
X1 < 0.095
X1 >= 0.26
X1 < 0.74
X1 >= 0.68
X1 >= 0.99
X2 >= 0.85
X2 < 0.94
X2 >= 0.62
X1 >= 0.66
X1 >= 0.079
X1 < 0.8
X2 >= 0.74
X1 >= 0.95
X2 < 0.71
X1 < 0.96

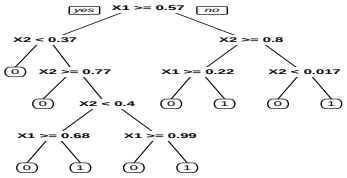

yes
X1 >= 0.57
no
X2 < 0.37
X2 >= 0.8
X2 >= 0.77
X1 >= 0.22
X2 < 0.017
X2 < 0.4
X1 >= 0.68
X1 >= 0.99

Exemple

```
> printcp(arbre)
Classification tree:
rpart(formula = Y ~ ., data = donnees, cp = 1e-04, minsplit = 2)
Variables actually used in tree construction:
[1] X1 X2
Root node error: 204/500 = 0.408
n= 500
          CP nsplit rel error  xerror     xstd
1  0.2941176      0  1.000000 1.00000 0.053870
2  0.1225490      1  0.705882 0.71569 0.049838
3  0.0931373      3  0.460784 0.49020 0.043844
4  0.0637255      4  0.367647 0.43627 0.041928
5  0.0122549      5  0.303922 0.34314 0.038034
6  0.0098039      7  0.279412 0.34314 0.038034
7  0.0049020      9  0.259804 0.36275 0.038923
8  0.0040107     25  0.181373 0.34804 0.038260
9  0.0036765     41  0.112745 0.39216 0.040184
10 0.0032680     49  0.083333 0.40196 0.040586
11 0.0024510     52  0.073529 0.41176 0.040980
12 0.0001000     82  0.000000 0.43137 0.041742
```

```
> arbre1 <- prune(arbre,cp=0.005)
> prp(arbre)
> prp(arbre1)
```





Choix d'un arbre

Il reste à sélectionner un arbre dans la suite

$$T_{max} = T_0 \supset T_1 \supset \ldots \supset T_M$$

.

Sélection d'un arbre

Choix d'un risque

La sélection de l'arbre final s'effectue en choisissant l'élément de la suite qui minimise le risque moyen $\mathbf{E}[R(Y, T_m(X)]$. Par exemple,

1. l'erreur quadratique $\mathbf{E}[(Y - T_m(X))^2]$ en *régression*;
2. la probabilité d'erreur $\mathbf{P}(Y \neq T_m(X))$ en *discrimination binaire*.

Ce risque (inconnu) est estimé par validation croisée.

Choix de l'arbre final

L'approche consiste à

1. *estimer le risque* pour chaque α_m.
2. choisir le α_m qui *minimise le risque estimé* $\Longrightarrow T_{\alpha_m}$.

Elagage/pruning - Algorithme

Algorithme

1. Calculer la suite $\alpha_0 = 0 < \alpha_1 < \ldots < \alpha_M$ et poser

$$\beta_1 = 0, \quad \beta_2 = \sqrt{\alpha_1 \alpha_2}, \quad \beta_3 = \sqrt{\alpha_2 \alpha_3}, \quad \ldots, \quad \beta_{M+1} = \infty.$$

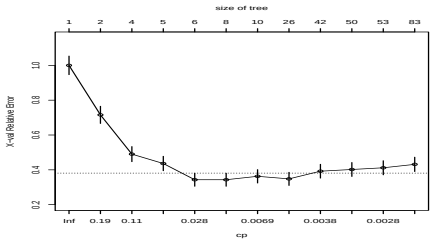
2. *Séparer les données* en K blocs $G_1, \ldots, G_k$ de taille k/n. Pour $i = 1, \ldots, k$:
   (a) Construire les arbres $T_{\beta_1}, \ldots, T_{\beta_{M+1}}$ sur l'ensemble des observations *privé du ième bloc.*
   (b) En déduire pour tout $j \in G_i$ et tout $m \leq M+1$, $\hat{Y}_j(\beta_m) = T_{\beta_m}(X_j)$.
3. Calculer $\mathcal{R}(m) = \frac{1}{n} \sum_{i=1}^{n} R(Y_i, \hat{Y}_i(\beta_m))$ pour $m = 1, \ldots, M+1$.
4. *Choisir* $\alpha_{m^\star}$ tel que $\beta_{m^\star+1} = \text{argmin}_{m \leq M+1} \mathcal{R}(m)$.

— Les estimations $\mathcal{R}(m)$ se trouvent dans la colonne *xerror* de la fonction printcp :

```
          CP nsplit rel error  xerror     xstd
1  0.2941176      0  1.000000 1.00000 0.053870
2  0.1225490      1  0.705882 0.71569 0.049838
3  0.0931373      3  0.460784 0.49020 0.043844
4  0.0637255      4  0.367647 0.43627 0.041928
5  0.0122549      5  0.303922 0.34314 0.038034
6  0.0098039      7  0.279412 0.34314 0.038034
7  0.0049020      9  0.259804 0.36275 0.038923
```

— On peut représenter les erreurs en fonction des α_m à l'aide de plotcp

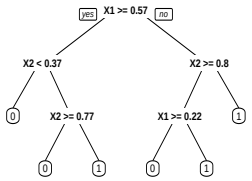
```
> plotcp(arbre3)
```



On choisira l'arbre à 5 coupures.

Tracé de l'arbre final

```
> alpha_opt <- arbre$cptable[which.min(arbre$cptable[,"xerror"]),"CP"]
> arbre_final <- prune(arbre,cp=alpha_opt)
> prp(arbre_final)
```



Règle de classification et score par arbre

— L'arbre final $\mathcal{T}$ renvoie une *partition* de $\mathbb{R}^p$ en $|\mathcal{T}|$ nœuds terminaux $\mathcal{N}_1, \ldots, \mathcal{N}_{|\mathcal{T}|}$.

— Règle de classification :

$$\hat{g}(x) = \begin{cases} 1 & \text{si } \sum_{i:X_i \in \mathcal{N}(x)} \mathbf{1}_{Y_i=1} \geq \sum_{i:X_i \in \mathcal{N}(x)} \mathbf{1}_{Y_i=0} \\ 0 & \text{sinon,} \end{cases}$$

où $\mathcal{N}(x)$ désigne le nœud terminal qui contient x.

— Score :

$$\hat{S}(x) = \hat{\mathbf{P}}(Y = 1 | X = x) = \frac{1}{n} \sum_{i:X_i \in \mathcal{N}(x)} \mathbf{1}_{Y_i=1}.$$

Fonction predict

— La fonction **predict** (**predict.rpart**) permet d'*estimer* la classe ou le score :

```
> x_new <- data.frame(X1=0.5,X2=0.85)
> predict(arbre_final,newdata=x_new)
    0   1
1 0.9 0.1
> predict(arbre_final,newdata=x_new,type="class")
1
0
Levels: 0 1
```

Bilan

— Méthode « simple » relativement facile à mettre en œuvre.

— Fonctionne en *régression* et en *discrimination*.

— Résultats *interprétables* (à condition que l'arbre ne soit pas trop profond).

— Un inconvénient : méthode connue pour être *instable*, sensible à de légères perturbations de l'échantillon.

— Cet inconvénient sera un avantage pour des *agrégations bootstrap* $\Longrightarrow$ forêts aléatoires.

4 Annexe : arbres Chaid

— CHAID : Chi2 Automatic Interaction Detection [Kass, 1980].

— *2 étapes* χ^2 dans le procédé de division d'un nœud :
  — regrouper les modalités peu discriminantes de chaque variable explicative X_j ;
  — choisir la variable à utiliser pour scinder le nœud.

χ^2 d'indépendance : rappel

— Soient X et Y deux variables aléatoires à valeurs dans E et F. On souhaite tester au niveau α les hypothèses H_0 : *"X et Y sont indépendantes"* contre H_1 : "X et Y ne sont pas indépendantes".

— On se donne $(E_1, \ldots, E_I)$ et $(F_1, \ldots, F_J)$ deux partitions de E et F.

— On dispose de n mesures du couple (X, Y) et on désigne par N_{ij} l'effectif observé dans la classe $E_i \times F_j$.

	F_1	$\ldots$	F_j	$\ldots$	F_J	Total
E_1	N_{11}	$\ldots$	N_{1j}	$\ldots$	N_{1J}	$N_{1\bullet}$
$\vdots$						$\vdots$
E_i	N_{i1}	$\ldots$	N_{ij}	$\ldots$	N_{iJ}	$N_{i\bullet}$
$\vdots$						$\vdots$
E_I	N_{I1}	$\ldots$	N_{Ij}	$\ldots$	N_{IJ}	$N_{I\bullet}$
Total	$N_{\bullet 1}$	$\ldots$	$N_{\bullet j}$	$\ldots$	$N_{\bullet J}$	n

Le test

Propriété

Sous H_0 la statistique

$$X_n = \sum_{i=1}^{I}\sum_{j=1}^{J} \frac{\left(\frac{N_{I\bullet}N_{\bullet J}}{n} - N_{ij}\right)^2}{\frac{N_{I\bullet}N_{\bullet J}}{n}}$$

converge en loi vers la loi $\chi^2_{(I-1)(J-1)}$.

Conséquence

- Au niveau α, on *rejettera l'hypothèse* H_0 si X_{obs} est supérieure au quantile d'ordre $1-\alpha$ de la loi du $\chi^2_{(I-1)(J-1)}$.
- Une *forte valeur de* X_{obs} (ou une *faible valeur de la probabilité critique*) signifiera un *lien fort* entre les deux variables.

Chaid : le principe

- On suppose dans un premier temps que toutes les variables explicatives $X_j, j = 1, \ldots, p$ sont qualitatives à M_j modalités.

Division d'un nœud

1. Regroupement des modalités peu discriminantes de chaque variable X_j ;
2. Choix de la variable X_j la plus discriminante
3. Le nœud est alors divisé en un nombre de nœuds fils égal au nombre de modalités créées à l'étape 1.

4.1 Regroupement des modalités

1. On se place dans un nœud $\mathcal{N}$ et on considère une variable X_j à M_j modalités ;
2. Les observations dans le nœud définissent *la table de contingence* suivante

	M_1	$\ldots$	M_j
1			
$\vdots$			
K			

3. $\forall (M_i, M_\ell) \in \{M_1, \ldots, M_j\}^2$, on calcule la *statistique du* χ^2 croisant Y et les modalités $(M_i, M_\ell) \Longrightarrow \chi^2(M_i, M_\ell)$ et $p(M_i, M_\ell)$ la probabilité critique associée.

   Remarque

   - 2 modalités *discriminantes* $\Longrightarrow$ dépendance *forte* dans le test avec $Y \Longrightarrow$ "Fort rejet" de $H_0 \Longrightarrow \chi^2$ *élevé* ou pc *faible* ;
   - Regrouper les *modalités peu discriminantes* revient donc à regrouper celles qui ont un χ^2 *faible* ou *une pc grande.*

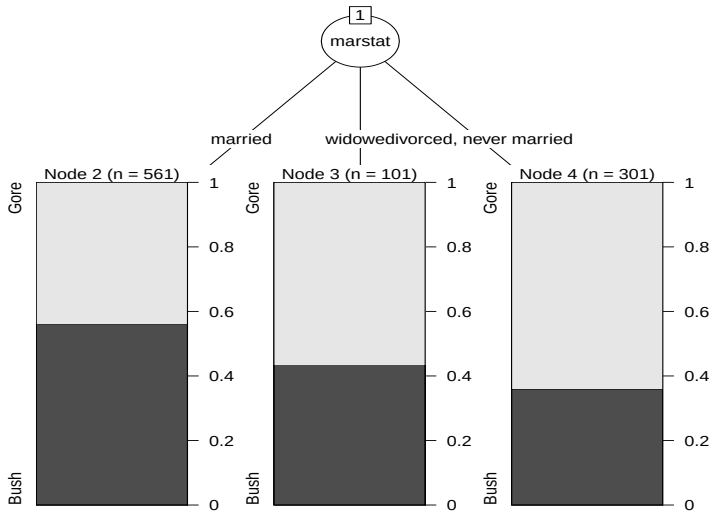
4. On choisit la *paire de modalités* qui minimise le χ^2 :

$$(\tilde{M}_i, \tilde{M}_\ell) = \underset{(M_i,M_\ell)\in\{M_1,\ldots,M_j\}^2}{\operatorname{argmin}} \chi^2(M_i, M_\ell) = \underset{(M_i,M_\ell)\in\{M_1,\ldots,M_j\}^2}{\operatorname{argmax}} p(M_i, M_\ell).$$

5. Si $p(\tilde{M}_i, \tilde{M}_\ell) > \alpha_2$ ($\alpha_2 \in]0,1[$ *fixé par l'utilisateur*) alors on regroupe les modalités $\tilde{M}_i$ et $\tilde{M}_\ell$ et on retourne à l'étape 2 avec le tableau à $M_j - 1$ modalités

	M_1	$\ldots$	$M_j - 1$
1			
$\vdots$			
K			

   Sinon, on stoppe les regroupements.



Exemple

— On considère la variable *marstat* :

```
> aa <- table(USvoteS$vote3,USvoteS$marstat)
> aa

      married widowed divorced never married
  Gore     246      57       82           111
  Bush     315      44       48            60
```

— On calcule les *probabilités critiques* pour les 6 croisements :

```
> res <- matrix(0,nrow=4,ncol=4)
> rownames(res) <- levels(USvoteS$marstat)
> colnames(res) <- levels(USvoteS$marstat)
> for (i in 1:3)
+   for (j in (i+1):4)
+     res[i,j] <- chisq.test(aa[,c(i,j)])$p.value
+
+
> res
              married widowed divorced never married
married             0  0.0194 7.64e-05      1.41e-06
widowed             0  0.0000 3.06e-01      1.65e-01
divorced            0  0.0000 0.00e+00      7.42e-01
never married       0  0.0000 0.00e+00      0.00e+00
```

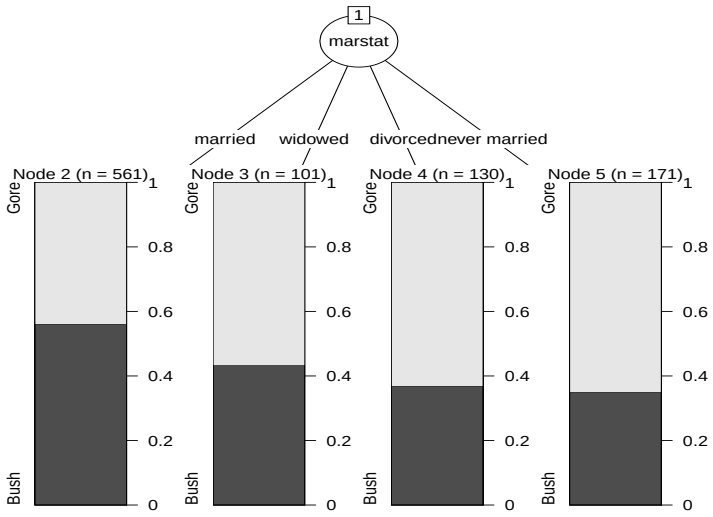
Exemple de regroupement

Les modalités divorced et never married sont regroupées (si $\alpha_2 < 0.742$).

— En effet

```
> ctrl <- chaid_control(minsplit = 20,alpha2=0.74)
> a1 <- chaid(vote3~marstat,data=USvoteS,control = ctrl)
> plot(a1)
```

```
> ctrl <- chaid_control(minsplit = 20,alpha2=0.75)
> a2 <- chaid(vote3~marstat,data=USvoteS,control = ctrl)
> plot(a2)
```



Variables continues et ordinales

— Variables ordinale : le traitement est *identique*. Seules les *modalités contiguës* peuvent être regroupées.

— Variables continues : traitées comme des variables ordinales. Penser à utiliser *as.ordered* sur R.

4.2 Division d'un nœud

Un autre χ^2 pour choisir la variable

— La phase *regroupement* effectuée, il faut choisir une variable parmi les p variables regroupées pour *diviser* le nœud.

— *Idée* : faire un χ^2 pour chaque variable :

	(X_1, M_1)	$\dots$	(X_1, M_{1j})	(X_2, M_1)	$\dots$	(X_2, M_{2j})	$\dots$
1							
$\vdots$							
K							

$\Longrightarrow$ p probabilités critiques $p(X_1), \dots, p(X_p)$ et

— X_j discriminante $\Longrightarrow$ rejet de H_0 $\Longrightarrow$ $p(X_j)$ petite.

— On *choisit la variable* j qui possède la plus *petite probabilité critique*.

Correction de Bonferroni

— Tendance à *favoriser* les variables ayant subi le *plus de regroupements* (erreur de type 1).

— Pour rééquilibrer, les probabilités critiques sont multipliées par le coefficient de Bonferroni :

$$p'(X_j) = b_j p(X_j)$$

où b_j correspond au nombre de manières les regrouper les M_j modalités initiales de X_j en $\tilde{M}_j$ modalités finales.

— Variable *qualitative* et ordinale :

$$b_j = \sum_{i=0}^{\tilde{M}_j - 1} (-1)^i \frac{(\tilde{M}_j - i)^{M_j}}{i!(\tilde{M}_j - i)!} \qquad b_j = \binom{M_j - 1}{\tilde{M}_j - 1}.$$

— On choisira la variable $j^\star$ qui *minimise* $p'(X_j)$...

— à condition que $p'(X_j)$ soit *plus petit qu'un certain seuil* α_4 fixé par l'utilisateur.

— Le nœud sera scindé en *autant de groupes* que X_j *possède de modalités* (après la phase de regroupement).

Critère d'arrêt

Un nœud ne sera pas divisé si :

— $p'(X_j) > \alpha_4$ pour *tout* $j = 1, \ldots, p$.

— le nœud est *pur* ou quasiment pur.

— le nœud contient *trop peu d'observations*...

Remarque

Sur R, on pourra regarder la fonction chaid.control :

```
chaid_control(alpha2 = 0.05, alpha3 = -1, alpha4 = 0.05,
              minsplit = 20, minbucket = 7, minprob = 0.01,
              stump = FALSE, maxheight = -1)
```

4.3 Choix des paramètres

— En plus des paramètres associés au critère d'arrêt, *deux paramètres sont à calibrer* pour construire l'arbre : les niveaux α_2 et α_4.

— Il en existe un troisième (α_3) qui concerne le remise en cause des *regroupements des modalités*.

Choix de α_4

Degrés d'exigence pour couper un nœud :

— petit : très exigeant $\Longrightarrow$ arbres peu profonds (beaucoup de biais et peu de variance) ;

— grand : peu exigeant $\Longrightarrow$ arbres profonds (beaucoup de variance et peu de biais).

Choix de α_2

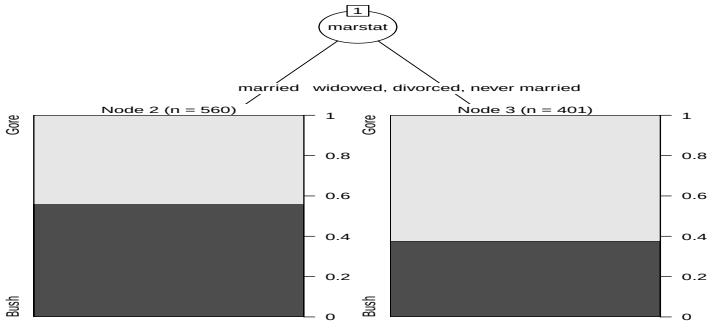
Degrés d'exigence pour regrouper des modalités :

— petit : peu exigeant $\Longrightarrow$ beaucoup de regroupements (on se rapproche des arbres binaires) ;

— grand : très exigeant $\Longrightarrow$ peu de regroupements.

Illustration α_4

```
> ctrl <- chaid_control(minsplit = 20,alpha4=0.0005)
> a1 <- chaid(vote3~.,data=USvoteS,control=ctrl)
> plot(a1)
```



```
> ctrl <- chaid_control(minsplit = 20,alpha4=0.25)
> a2 <- chaid(vote3~.,data=USvoteS,control=ctrl)
> plot(a2)
```

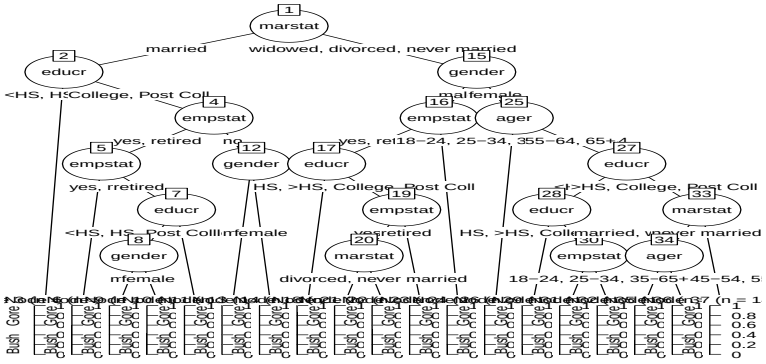
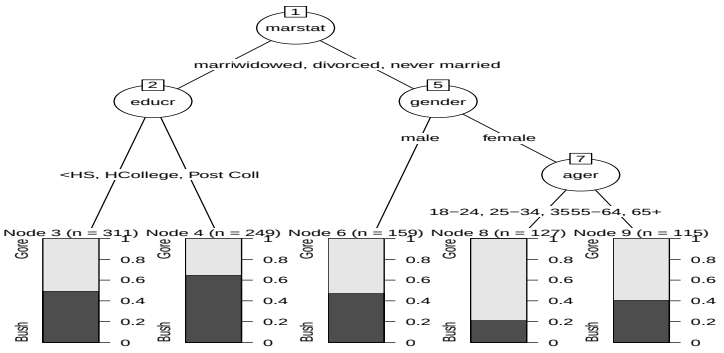
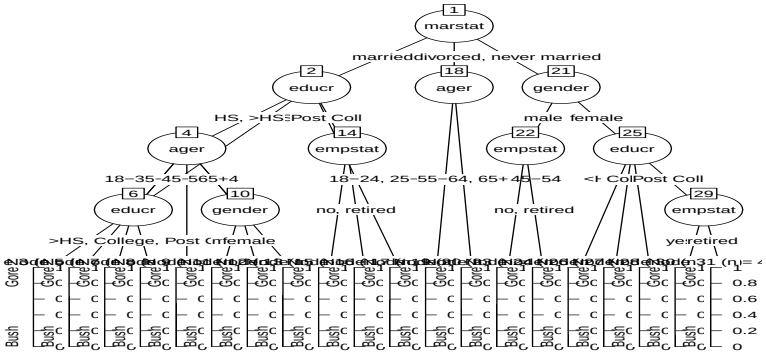


Illustration α_2

```
> ctrl <- chaid_control(minsplit = 20,alpha2=0.005)
> a3 <- chaid(vote3~.,data=USvoteS,control=ctrl)
> plot(a3)
```




```
> ctrl <- chaid_control(minsplit = 20,alpha2=0.5)
> a4 <- chaid(vote3~.,data=USvoteS,control=ctrl)
> plot(a4)
```



En pratique...

— L'*influence* de ces deux paramètres est bien entendu *conjointe*.

— Il n'est pas facile de les calibrer simultanément.

— *Approche classique* : évaluer les performances (erreur de classification AUC...) pour plusieurs valeurs de (α_2, α_4) sur un échantillon test ou par validation croisée.

Exemple

— On veut expliquer avec un *arbre CHAID* la variable chd par les autres variables du jeu de données *SAheart*.

```
> donnees <- SAheart
> donnees$chd <- as.factor(donnees$chd)
> for (i in c(1:4,6:9)){donnees[,i] <- as.ordered(donnees[,i])}
```

— On va séparer l'échantillon en 2 et *estimer l'erreur de classification* sur une grille de valeur de α_2 et α_4 :

```
> alpha2 <- seq(0.01,0.35,by=0.05)
> alpha4 <- seq(0.01,0.35,by=0.05)
> gr.alpha <- expand.grid(alpha2,alpha4)
> names(gr.alpha) <- c("alpha2","alpha4")
> gr.alpha$perf <- 0
> set.seed(1234)
> perm <- sample(nrow(SAheart))
> dapp <- donnees[perm[1:300],]
> dtest <- donnees[-perm[1:300],]
```

— On estime *l'erreur de classification* sur les données test :

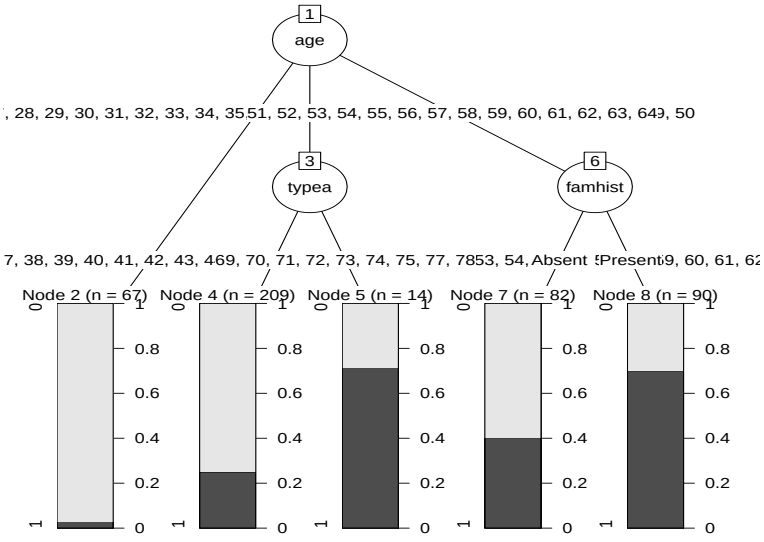
```
> for (i in 1:nrow(gr.alpha)){
>  ctrl <- chaid_control(alpha2=gr.alpha[i,1],alpha4=gr.alpha[i,2])
>  a <- chaid(chd~.,data=dapp,control=ctrl)
>  prev <- predict(a,newdata = dtest)
>  gr.alpha$perf[i] <- mean(prev!=dtest$chd)
}
```

— On récupère les valeurs de α_2 et α_4 qui minimisent l'erreur estimée :

```
> alpha_opt <- gr.alpha[which.min(gr.alpha$perf),]
> alpha_opt
  alpha2 alpha4      perf
1   0.01   0.01 0.2716049
```

— On peut *tracer l'arbre sélectionné* :

```
> ctrl <- chaid_control(alpha2=alpha_opt[1],alpha4=alpha_opt[2])
> arbre_final <- chaid(chd~.,data=donnees,control=ctrl)
> plot(arbre_final)
```



Avec Caret

— On peut faire la même chose avec *caret* (en plus efficace) :

```
> grille <- gr.alpha[,1:2]
> grille$alpha3 <- -1
> library(doMC)
> registerDoMC(cores = 3)
> bb <- train(donnees[,-10],donnees$chd,method="chaid",
     tuneGrid=grille,trControl=ctrl1,metric="Accuracy")
> bb
CHi-squared Automated Interaction Detection

462 samples
  9 predictor
  2 classes: '0', '1'
```

```
No pre-processing
Resampling: Repeated Train/Test Splits Estimated (1 reps, 75%)
Summary of sample sizes: 300
Resampling results across tuning parameters:

  alpha2  alpha4  Accuracy   Kappa
  0.01    0.01    0.7283951  0.3847747
  0.01    0.06    0.7283951  0.3847747
  0.01    0.11    0.7283951  0.3847747
  0.01    0.16    0.7283951  0.3847747
  0.01    0.21    0.7283951  0.3847747
  0.01    0.26    0.6851852  0.2528486
  0.01    0.31    0.6851852  0.2528486
  0.06    0.01    0.6851852  0.2528486
  0.06    0.06    0.6851852  0.2528486
  0.06    0.11    0.6851852  0.2528486
  0.06    0.16    0.6851852  0.2528486
  0.06    0.21    0.6851852  0.2528486
  0.06    0.26    0.6728395  0.3284843
  0.06    0.31    0.6728395  0.2302313
  0.11    0.01    0.6419753  0.2394366
  0.11    0.06    0.6419753  0.2839506
  0.11    0.11    0.6419753  0.2839506
  0.11    0.16    0.6419753  0.2839506
  0.11    0.21    0.6419753  0.2839506
  0.11    0.26    0.6296296  0.2646391
  0.11    0.31    0.6419753  0.2839506
  0.16    0.01    0.6419753  0.2394366
  0.16    0.06    0.6419753  0.2839506
  0.16    0.11    0.6419753  0.2839506
  0.16    0.16    0.6419753  0.2839506
  0.16    0.21    0.6419753  0.2839506
  0.16    0.26    0.6296296  0.2646391
  0.16    0.31    0.6419753  0.2839506
  0.21    0.01    0.6419753  0.2394366
  0.21    0.06    0.6419753  0.2394366
  0.21    0.11    0.6419753  0.2394366
  0.21    0.16    0.6419753  0.2394366
  0.21    0.21    0.6419753  0.2394366
  0.21    0.26    0.6419753  0.2394366
  0.21    0.31    0.6419753  0.2394366
  0.26    0.01    0.6419753  0.2394366
  0.26    0.06    0.6419753  0.2394366
  0.26    0.11    0.6419753  0.2394366
  0.26    0.16    0.6419753  0.2394366
  0.26    0.21    0.6419753  0.2394366
  0.26    0.26    0.6419753  0.2394366
  0.26    0.31    0.6419753  0.2394366
  0.31    0.01    0.6419753  0.2394366
  0.31    0.06    0.6419753  0.2394366
  0.31    0.11    0.6419753  0.2394366
  0.31    0.16    0.6419753  0.2394366
  0.31    0.21    0.6419753  0.2394366
  0.31    0.26    0.6419753  0.2394366
  0.31    0.31    0.6419753  0.2394366

Tuning parameter 'alpha3' was held constant at a value of -1
Accuracy was used to select the optimal model using  the largest value.
The final values used for the model were alpha2 = 0.01, alpha3 = -1
 and alpha4 = 0.21.
```

Cinquième partie

Bagging et forêts aléatoires

Cadre

— Idem que précédemment, on cherche à *expliquer* une variable Y par d variables explicatives $X_1, \dots, X_d$.

— Pour simplifier on se place en *régression* : Y est à valeurs dans $\mathbb{R}$ mais tout ce qui va être fait s'étant directement à la *classification binaire ou multiclasses*.

— *Notations* :

— (X, Y) un couple aléatoire à valeurs dans $\mathbb{R}^d \times \mathbb{R}$.

— $\mathcal{D}_n = (X_1, Y_1), \dots, (X_n, Y_n)$ un n-échantillon i.i.d. de même loi que (X, Y).

1 Bagging

— Le *bagging* désigne un ensemble de méthodes introduit par Léo Breiman [Breiman, 1996].

— *Bagging* : vient de la contraction de **B**ootstrap **Agg**regat**ing**.

— *Idée* : plutôt que de constuire un seul estimateur, en construire un grand nombre (sur des échantillons bootstrap) et les agréger.

Pourquoi agréger ?

— On se place dans le modèle de régression

$$Y = m(X) + \varepsilon.$$

— On note

$$\hat{m}_B(x) = \frac{1}{B} \sum_{k=1}^{B} m_k(x)$$

un estimateur de m obtenu en agrégeant B estimateurs $m_1, \dots, m_B$.

— *Rappels* : $\hat{m}_B(x) = \hat{m}_B(x; (X_1, Y_1), \dots, (X_n, Y_n))$ et $m_k(x) = m_k(x; (X_1, Y_1), \dots, (X_n, Y_n))$ sont des variables aléatoires.

— On peut *mesurer l'intérêt d'agréger* en comparant les performances de $\hat{m}_B(x)$ à celles des $m_k(x), k = 1, \dots, B$ (en comparant, par exemple, le *biais* et la *variance* de ces estimateurs).

Biais et variance

— *Hypothèse :* les variables aléatoires $m_1, \dots, m_B$ sont i.i.d.

— Biais :

$$\mathbf{E}[\hat{m}_B(x)] = \mathbf{E}[m_k(x)].$$

Conclusion
Agréger ne modifie pas le biais.

— Variance :

$$\mathbf{V}[\hat{m}_B(x)] = \frac{1}{B} \mathbf{V}[m_k(x)].$$

Conclusion
Agréger tue la variance.

1	2	3	4	5	6	7	8	9	10

3	4	6	10	3	9	10	7	7	1	m_1
2	8	6	2	10	10	2	9	5	6	m_2
2	9	4	4	7	7	2	3	6	7	m_3
6	1	3	3	9	3	8	10	10	1	m_4
3	7	10	3	2	8	6	9	10	2	m_5
	⋮								⋮	
7	10	3	4	9	10	10	8	6	1	m_B

— Les conclusions précédentes sont vraies sous *l'hypothèse* que les variables aléatoires $m_1, \ldots, m_B$ sont i.i.d.

— Les estimateurs $m_1, \ldots, m_B$ étant construits sur le même échantillon, *l'hypothèse d'indépendance n'est clairement pas raisonnable !*

Idée

Atténuer la dépendance entre les estimateurs $m_k, k = 1, \ldots, B$ en introduisant de nouvelles sources d'aléa.

Idée : échantillons bootstrap

— Echantillon *initial* :

— Echantillons *bootstrap* :

— A la fin, on *agrège* :

$$\hat{m}_B(x) = \frac{1}{B} \sum_{k=1}^{B} m_k(x).$$

Bagging

— Les estimateurs m_k ne vont pas être construits sur l'échantillon $\mathcal{D}_n = (X_1, Y_1), \ldots, (X_n, Y_n)$, mais sur des *échantillons bootstrap* de $\mathcal{D}_n$.

Bagging

Entrées :

— $x \in \mathbb{R}^d$ l'observation à prévoir ; $\mathcal{D}_n$ l'échantillon

— un régresseur (arbre CART, 1 plus proche voisin...)

— B le nombre d'estimateurs que l'on agrège.

Pour $k = 1, \ldots, B$:

1. Tirer un échantillon *bootstrap* dans $\mathcal{D}_n$
2. *Ajuster le régresseur* sur cet échantillon bootstrap : $m_k(x)$

Sortie : L'estimateur $\widehat{m}_B(x) = \frac{1}{B} \sum_{k=1}^{B} m_k(x)$.

Tirage de l'échantillon boostrap

— Les tirages bootstrap sont représentés par B *variables aléatoires* $\theta_k, k = 1, \ldots, B$.

— Les tirages bootstrap sont généralement effectués selon *la même loi et de façon indépendante* : $\theta_1, \ldots, \theta_B$ sont *i.i.d.* de même loi que θ.

— 2 techniques sont généralement utilisées :

1. tirage de n *observations avec remise* ;

2. tirage de $\ell < n$ *observation sans remise.*

Conséquence

Les estimateurs agrégés contiennent 2 sources d'aléa (échantillon et tirage bootstrap) :

$$m_k(x) = m(x, \theta_k, \mathcal{D}_n).$$

Choix du nombre d'itérations

— Deux paramètres sont à choisir : *le nombre d'itérations* B *et le régresseur.*

— On a d'après la *loi des grands nombres*

$$\begin{aligned}\lim_{B\to\infty} \hat{m}_B(x) = \lim_{B\to\infty} \frac{1}{B}\sum_{k=1}^{B} m_k(x) = \lim_{B\to\infty} \frac{1}{B}\sum_{k=1}^{B} m(x, \theta_k, \mathcal{D}_n)\\ = \mathbf{E}_\theta[m(x, \theta, \mathcal{D}_n)] = \bar{m}(x, \mathcal{D}_n) \quad p.s|\mathcal{D}_n.\end{aligned}$$

— Lorsque B est grand, $\hat{m}_B$ se *"stabilise"* vers l'estimateur *bagging* $\bar{m}(x, \mathcal{D}_n)$.

Conséquence importante

Le nombre d'itérations B n'est pas un paramètre à calibrer, il est conseillé de le prendre le plus grand possible en fonction du temps de calcul.

Choix du régresseur

Propriété : biais et variance

On a

$$\mathbf{E}[\hat{m}_B(x)] = \mathbf{E}[m_k(x, \theta_k, \mathcal{D}_n)]$$

et

$$\mathbf{V}[\hat{m}_B(x)] = \rho(x)\mathbf{V}[m(x, \theta_k, \mathcal{D}_n)] + \frac{1-\rho(x)}{B}\mathbf{V}[m(x, \theta_k, \mathcal{D}_n)]$$

où $\rho(x) = corr(m(x, \theta_k, \mathcal{D}_n), m(x, \theta_{k'}, \mathcal{D}_n)))$ pour $k \neq k'$.

Conséquence

— Bagger ne modifie pas le biais.

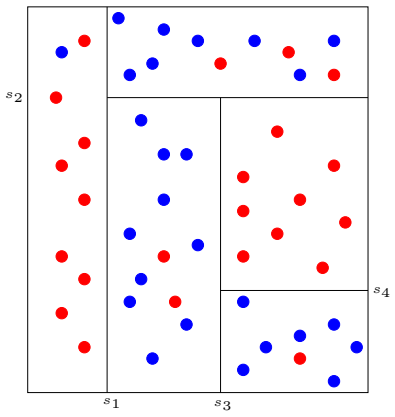
— Pour$B$ grand, $\mathbf{V}[\hat{m}_B(x)] \approx \rho(x)\mathbf{V}[\hat{m}_k(x, \theta_k(\mathcal{D}_n)] \Longrightarrow$ la *variance diminue d'autant plus que la corrélation entre les prédicteurs diminue.*

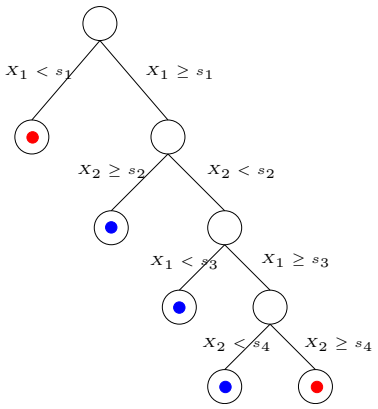
— Il est donc nécessaire d'agréger des estimateurs *sensibles à de légères perturbations de l'échantillon.*

— Les arbres sont connus pour posséder de telles propriétés.

2 Forêts aléatoires

Rappels sur les arbres





Paramètre à calibrer

Profondeur de l'arbre :

— petite : biais ↗, variance ↘

— grande : biais ↘, variance ↗

Définition

— Comme son nom l'indique, une *forêt aléatoire* est définie à partir d'un ensemble d'arbres.

Définition

Soit $T_k(x), k = 1, \ldots, B$ des prédicteurs par arbre ($T_k : \mathbb{R}^d \to \mathbb{R}$). Le prédicteur des *forêts aléatoires* est obtenu par agrégation de cette collection d'arbres :

$$\hat{T}_B(x) = \frac{1}{B} \sum_{k=1}^{B} T_k(x).$$

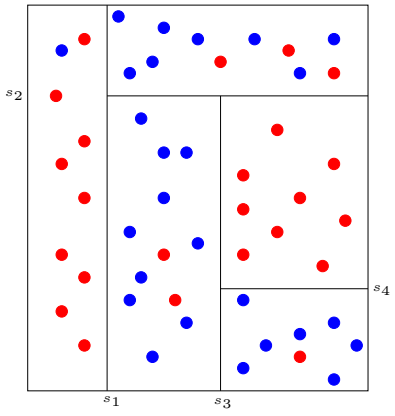
Forêts aléatoires

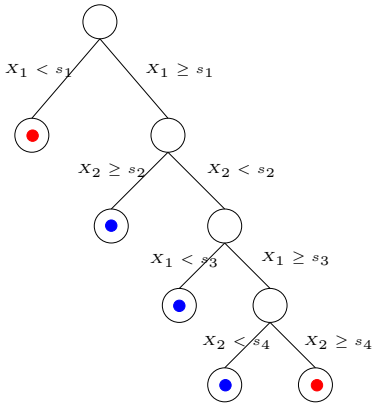
— Forêts aléatoires = *collection d'abres.*

— Les forêts aléatoires les plus utilisées sont (de loin) celles proposées par *Léo Breiman* (au début des années 2000).

— Elles consistent à *agréger* des arbres construits sur des *échantillons bootstrap.*

— On pourra trouver de la doc à l'url
`http://www.stat.berkeley.edu/~breiman/RandomForests/`
et consulter la thèse de Robin Genuer [Genuer, 2010].





Arbres pour forêt

— Breiman propose de sélectionner la "meilleure" variable dans un ensemble composé uniquement de m variables choisies aléatoirement parmi les d variables initiales.

— Objectif : diminuer la corrélation entre les arbres que l'on agrège.

Algorithme : randomforest

Entrées :

— $x \in \mathbb{R}^d$ l'observation à prévoir ;

— $\mathcal{D}_n$ l'échantillon ;

— B nombre d'arbres ; n_{max} nombre max d'observations par nœud

— $m \in \{1, \dots, d\}$ le nombre de variables candidates pour découper un nœud.

Pour $k = 1, \dots, B$:

1. Tirer un échantillon *bootstrap* dans $\mathcal{D}_n$
2. Construire un *arbre CART sur cet échantillon bootstrap*, chaque coupure est sélectionnée en minimisant la fonction de coût de CART sur un ensemble de m *variables choisies au hasard* parmi les d. On note $T(., \theta_k, \mathcal{D}_n)$ l'arbre construit.

Sortie : L'estimateur $T_B(x) = \frac{1}{B} \sum_{k=1}^{B} T(x, \theta_k, \mathcal{D}_n)$.

Commentaires

— Si on est en discrimination (Y qualitative), l'étape d'agrégation consiste à faire *voter les arbres à la majorité*.

— Il y a deux sources d'aléa présentes dans θ_k : le tirage bootstrap et les m variables sélectionnées à chaque étape de la construction de l'arbre.

— Méthode *simple à mettre en oeuvre* et déjà *implémentée* sur la plupart des logiciels statistiques (sur R, il suffit de lancer la fonction **randomForest** du package randomForest).

— Estimateur connu pour fournir des *estimations précises* sur des données complexes (beaucoup de variables, données manquantes...).

— Estimateur *peu sensible* au choix de ses paramètres (B, n_{max}, m...)

Choix des paramètres

— B : réglé... *le plus grand possible.*

Intérêt du bagging (rappel)

Diminuer la variance des estimateurs qu'on agrège :

$$\mathbf{V}[\hat{T}_B(x)] = \rho(x)\mathbf{V}[T(x, \theta_k, \mathcal{D}_n)] + \frac{1 - \rho(x)}{B}\mathbf{V}[T(x, \theta_k, \mathcal{D}_n)]$$

Conséquence

— Le biais n'étant pas amélioré par "l'agrégation bagging", il est recommandé d'agréger des estimateurs qui possèdent un biais faible (*contrairement au boosting*).

— On choisira donc des arbres *"profonds"*, c'est-à-dire avec *peu d'observation dans les nœuds terminaux.*

— Par défaut dans randomForest, $n_{max} = 5$ en régression et 1 en classification.

Choix de m

— Il est en *relation avec la corrélation* entre les arbres $\rho(x)$.

— Ce paramètre a une (légère) influence sur le compromis biais/variance de la forêt.

— $m \searrow$

1. tendance à se rapprocher d'un *choix "aléatoire"* des variables de découpe des arbres $\Longrightarrow$ les arbres sont de plus en plus différents $\Longrightarrow \rho(x) \searrow \Longrightarrow$ la variance de la forêt diminue.
2. mais... le biais des arbres $\nearrow \Longrightarrow$ les biais de la forêt $\nearrow$.

— Inversement lorsque $m \nearrow$.

Conclusion

— Il est recommandé de comparer les performances de la forêt pour plusieurs valeurs de m.

— Par défaut $m = d/3$ en régression et $\sqrt{d}$ en classification.

Application sur les données spam

```
> library(randomForest)
> foret1 <- randomForest(type~.,data=spam)
> foret1

Call:
 randomForest(formula = type ~ ., data = spam)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 7

        OOB estimate of  error rate: 5.26%
Confusion matrix:
     0   1 class.error
0 1352  42  0.03012912
1   79 827  0.08719647
```

Mesure de performance

— Comme pour les autres classifieurs et régresseurs il convient de définir des critères qui permettent de *mesurer la performance des forêts aléatoires.*

— **Exemples :**

— Erreur de prédiction : $\mathbf{E}[(Y - \hat{T}_B(X))^2]$ en régression ;

— Probabilité d'erreur : $\mathbf{P}(Y \neq \hat{T}_B(X))$ en classification.

— Comme pour les autres méthodes, ces critères peuvent être évalués par *apprentissage/validation* ou *validation croisée.*

— La phase *bootstrap* des algorithme bagging permet de définir une nouvelle méthode d'estimation de ces critères : méthode OOB (Out Of Bag).

Erreur Ouf Of Bag

— Pour chaque observation (X_i, Y_i) de $\mathcal{D}_n$, on désigne par $\mathcal{I}_B$ l'ensemble des arbres de la forêt qui *ne contiennent pas cette observation* dans leur échantillon bootstrap.

— La prévision de Y au point X_i se fait selon

$$\hat{Y}_i = \frac{1}{|\mathcal{I}_B|} \sum_{k \in \mathcal{I}_B} T(X_i, \theta_k, \mathcal{D}_n).$$

Estimateurs Our Of Bag

— L'erreur de prédiction est estimée par $\frac{1}{n}\sum_{i=1}^n (\hat{Y}_i - Y_i)^2$.

— La probabilité d'erreur est estimée par $\frac{1}{n}\sum_{i=1}^n \mathbf{1}_{\hat{Y}_i \neq Y_i}$.

3	4	6	10	3	9	10	7	7	*1*	m_1
2	8	6	2	10	10	2	9	5	6	m_2
2	9	4	4	7	7	2	3	6	7	m_3
6	*1*	3	3	9	3	8	10	10	*1*	m_4
3	7	10	3	2	8	6	9	10	2	m_5
7	10	3	4	9	10	10	8	6	*1*	m_6

Exemple

— Les échantillons 2, 3 et 5 *ne contiennent pas* la première observation, donc

$$\hat{Y}_1 = \frac{1}{3}(m_2(X_1) + m_3(X_1) + m_5(X_1)).$$

— On fait de même pour *toutes les observations* $\Longrightarrow \hat{Y}_2, \ldots, \hat{Y}_n$.

— On estime l'erreur selon

$$\frac{1}{n}\sum_{i=1}^{n}(\hat{Y}_i - Y_i)^2.$$

Exemple

— On construit la forêt avec $m = 1$:

```
> foret2 <- randomForest(type~.,data=spam,mtry=1)
> foret2

Call:
 randomForest(formula = type ~ ., data = spam, mtry = 1)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 1

        OOB estimate of  error rate: 8.04%
Confusion matrix:
     0   1 class.error
0 1367  27  0.01936872
1  158 748  0.17439294
```

Conclusion

L'erreur OOB est de 8.04%, elle est de 5.26% lorsque $m = 7$.

Importance des variables

— Un des reproches souvent fait aux forêts est l'aspect *boîte noire* et *manque d'interprétabilité* par rapport aux modèles paramétriques tels que le modèle logistique.

— Il existe un *indicateur* qui permet de mesurer l'*importance des variables* présentes dans le modèle.

— Comme l'erreur OOB, ce critère est basé sur le fait que toutes les observations *ne sont pas utilisées* pour construire les arbres de la forêt.

— Soit OOB_k l'*échantillon Out Of Bag* associé au k^{eme} arbre : il contient les observations qui *ne sont pas* dans le k^{eme} échantillon bootstrap.

— Soit E_{OOB_k} l'erreur de prédiction de l'arbre k mesurée sur cet échantillon :

$$E_{OOB_k} = \frac{1}{|OOB_k|}\sum_{i \in OOB_k}(T(X_i, \theta_k, \mathcal{D}_n) - Y_i)^2.$$

— Soit OOB_k^j l'échantillon OOB_k dans lequel on a *perturbé aléatoirement* les valeurs de la variable j et $E_{OOB_k^j}$ *l'erreur de prédiction* de l'arbre k mesurée sur cet échantillon :

$$E^j_{OOB_k} = \frac{1}{|OOB_k^j|} \sum_{i \in OOB_k^j} (T(X_i^j, \theta_k, \mathcal{D}_n) - Y_i)^2,$$

Définition

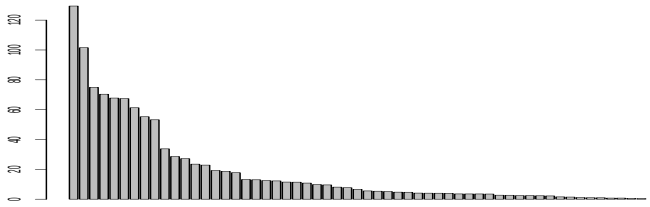
L'importance de la j^{eme} variable est définie par

$$Imp(X_j) = \frac{1}{B} \sum_{k=1}^{B} (E^j_{OOB_k} - E_{OOB_k}).$$

Exemple

— L'importance s'obtient facilement avec le package randomForest

```
> imp <- importance(foret1)
> imp1 <- sort(imp,decreasing=TRUE)
> ord <- order(imp,decreasing=TRUE)
> ord
 [1] 52 53 55  7 56 16 21 25 57  5 24 19 26 23 46 27 11  8 50 12 37  3 18  6 45
[26] 17 10  2 28 42 49 35  1 36 39 13 54  9 30 33 22 51 29 14 43 44 31 20 48 15
[51] 40  4 41 34 32 38 47
> barplot(imp1,beside=TRUE)
```

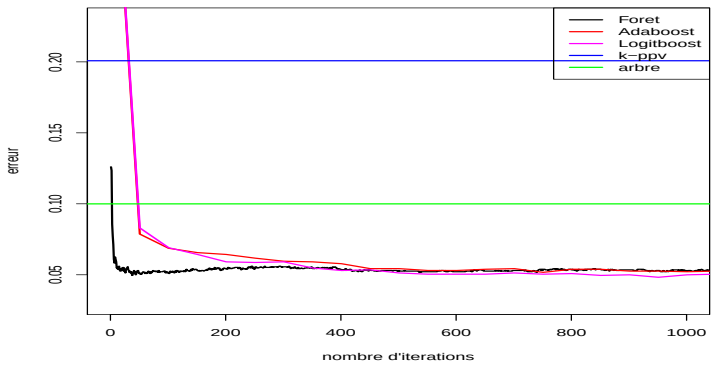


Comparaison de méthodes

— On *compare les performances du boosting* (Adaboost et Logitboost), des forêts aléatoires, d'un arbre de classification ainsi que la méthode des k-ppv sur les données *spam*.

— Pour ce faire, on ajuste les différents modèles sur un *échantillon d'apprentissage de taille* 2300 et on compare les performances de chaque méthode en estimant la probabilité d'erreur par l'*erreur empirique calculée sur l'échantillon test* de taille 2301 :

$$L_n(\hat{g}) = \frac{1}{n_{test}} \sum_{i \in \mathcal{D}_{test}} \mathbf{1}_{\hat{g}(X_i) \neq Y_i}.$$



Méthode	Erreur
Forêt	0.050
Ada	0.052
Logit	0.048
k-ppv	0.200
arbre	0.100

Références

[Breiman, 1996] Breiman, L. (1996). Bagging predictors. *Machine Learning*, 26(2) :123–140.

[Breiman et al., 1984] Breiman, L., Friedman, J., Olshen, R., and Stone, C. (1984). *Classification and regression trees*. Wadsworth & Brooks.

[Friedman, 1989] Friedman, J. (1989). Regularized discriminant analysis. *Journal of the American Statistical Association*, 84 :165–175.

[Genuer, 2010] Genuer, R. (2010). *Forêts aléatoires : aspects théoriques, sélection de variables et applications*. PhD thesis, Université Paris XI.

[Hastie et al., 2009] Hastie, T., Tibshirani, R., and Friedman, J. (2009). *The Elements of Statistical Learning : Data Mining, Inference, and Prediction*. Springer, second edition.

[Kass, 1980] Kass, G. (1980). An exploratory technique for investigating large quantities of categorical data. *Applied Statistics*, 29(2) :119–127.

[Saporta, 2011] Saporta, G. (2011). *Probabilités, analyse des données et statistique*. Tecnip, 3ème edition.