

# Apprentissage supervisé

Université Paris-Sud  
mohammed.sedki@u-psud.fr  
masedki.github.io

8 octobre 2019

## Introduction

### Validation croisée et bootstrap

## Sélection de variables en régression linéaire

Méthodes pas à pas

Critères d'information

Méthodes de régularisation, contraction de coefficients ou shrinkage

## Classification

Régression logistique

Sélection de variables en régression logistique

Analyse discriminante linéaire et quadratique

## Dendrologie

Arbres de décision uniques

Agrégation par bagging

Forêts aléatoires

Boosting

## Support Vector Machine (SVM)

# Introduction

# Références

- ▶ <http://wikistat.fr/>
- ▶ <https://github.com/wikistat/>

# Problèmes d'apprentissage statistique

1. Identifier les facteurs de risque du cancer de la prostate
2. Prédire si une personne est sujette au crise cardiaque, à partir de mesure clinique, son régime et des données démographiques
3. Classification d'échantillons de tissus dans différents types de cancer, en fonction de données d'expression de gènes
4. Classifier des images de tumeurs

# Apprentissage supervisé : point de départ

- ▶ Une variable de **sortie**  $Y$  (variable réponse, variable cible)
- ▶ Un vecteur de  $p$  variables  $X$  (appelé variables explicatives, covariables ou variables d'entrée *regressors, features, predictors*)
- ▶ En **régression**,  $Y$  est une variable quantitative (le dosage du PSA, tension artérielle)
- ▶ En **classification**,  $Y$  prend un nombre fini de valeurs (vivant/mort, les nombres 0-9, le type de cancer)
- ▶ Des données dites **d'apprentissage** *training data*  $(x_1, y_1), \dots, (x_N, y_N)$ . Ce sont les observations.

# Objectif

À partir des données d'apprentissage

- ▶ Prédire la réponse non observée d'un cas "test"
- ▶ Comprendre l'influence des variables explicatives sur la variable réponse (comment et de combien)
- ▶ Mesurer la qualité de notre "inférence statistique" notamment sa prédiction

# Pourquoi ce cours!!!

- ▶ Il est important de comprendre les différentes techniques pour savoir comment et quand les utiliser
- ▶ Il faut absolument comprendre les méthodes basiques en premier
- ▶ Il est indispensable d'évaluer la précision et les limites d'une méthode car souvent les méthodes simples fonctionnent mieux que les méthodes sophistiquées !!

# Un mot sur l'apprentissage non-supervisé

- ▶ Pas de réponse ( $Y$ )
- ▶ Buts plus flous : description de données, analyse de données
- ▶ Exemple : Regrouper des observations similaires (marketing,...)
- ▶ Difficulté de l'évaluation du résultat
- ▶ Différent de l'apprentissage supervisé, mais peut être une étape préliminaire à un apprentissage supervisé

# Apprentissage statistique versus Machine learning

Deux domaines qui nomment la même chose!!!

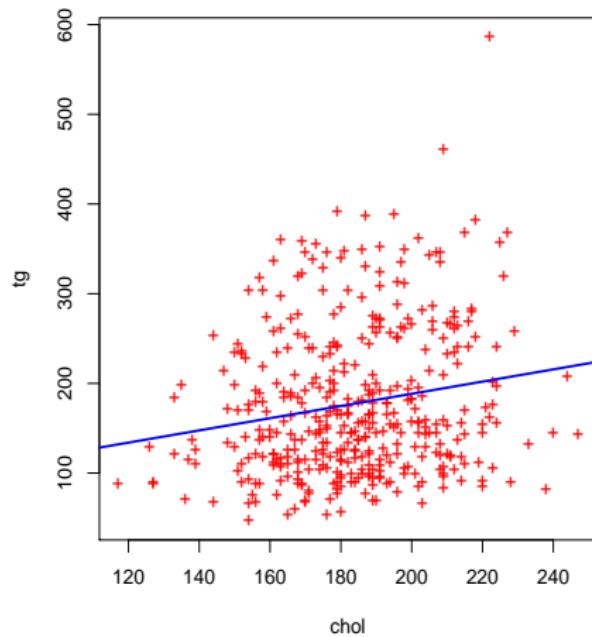
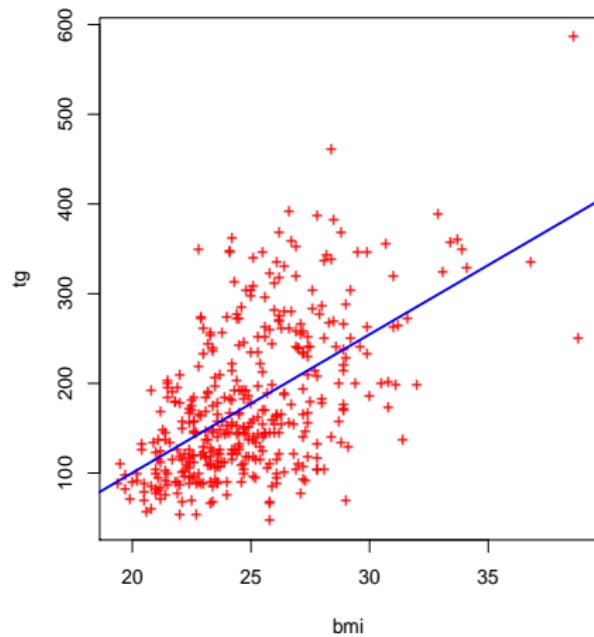
- ▶ Machine learning est une branche de l'intelligence artificielle
- ▶ Apprentissage statistique est une branche de la statistique

Différences entre les deux visions

- ▶ En Machine learning, on met l'accent sur l'application à grande échelle et la précision de la prédiction
- ▶ L'apprentissage statistique met l'accent sur les modèles et leur interprétabilité, précision et incertitude.

# Apprentissage statistique : exemple

triglycérides vs BMI et cholestérol avec la droite de régression



# Apprentissage : exemple + notations

- ▶ Peut-on prédire le taux de tri-glycérides en fonction du BMI et du taux de cholestérol, c'est à dire un modèle

$$\text{Triglycérides} \approx f(\text{BMI}, \text{cholestérol})$$

- ▶ Ici le taux de **Tri-glycérides** représente la variable réponse  $Y$  qu'on cherche à prédire ou à expliquer
- ▶ Le **BMI** est une variable explicative ou covariable qu'on note  $X_1$  et le taux **cholestérol** est aussi une variable explicative qu'on note  $X_2$ .

On peut maintenant écrire notre modèle

$$Y = f(X) + \varepsilon \quad \text{où} \quad X = \begin{pmatrix} X_1 \\ X_2 \end{pmatrix}$$

$\varepsilon$  capte l'erreur de mesure.

# Apprentissage : quelle est la meilleure fonction $f(X)$ ?

- ▶ Avec une "meilleure" fonction  $f$  on peut prédire  $Y$  pour une nouvelle observation des variables explicatives  $X = x$
- ▶ De manière générale, on peut identifier les composantes importantes de  $X = (X_1, X_2 \dots, X_p)$  pour expliquer  $Y$  and les composantes qui le sont moins. - Selon la complexité de  $f$ , on peut comprendre comment et de combien les composantes  $X_j$  de  $X$  influent sur la valeur de  $Y$ .

## La réponse est oui : la fonction de régression

- ▶ C'est  $f(0.5) = \mathbb{E}(Y|X = 0.5)$  la valeur moyenne de  $Y$  sachant que  $X = 0.5$ .
- ▶ C'est la fonction **optimale** pour l'erreur quadratique, c'est à dire, minimise  $\mathbb{E}((Y - g(X))^2 | X = x)$  pour toute fonction  $g$  et toute valeur de  $x$ .
- ▶  $\varepsilon = Y - f(X)$  est l'erreur incompressible, même si  $f$  est connue, pour chaque valeur de  $X$ , nous avons une réalisation d'une variable aléatoire  $Y$
- ▶ Pour toute estimateur  $\hat{f}(x)$  de  $f(x)$ , nous avons

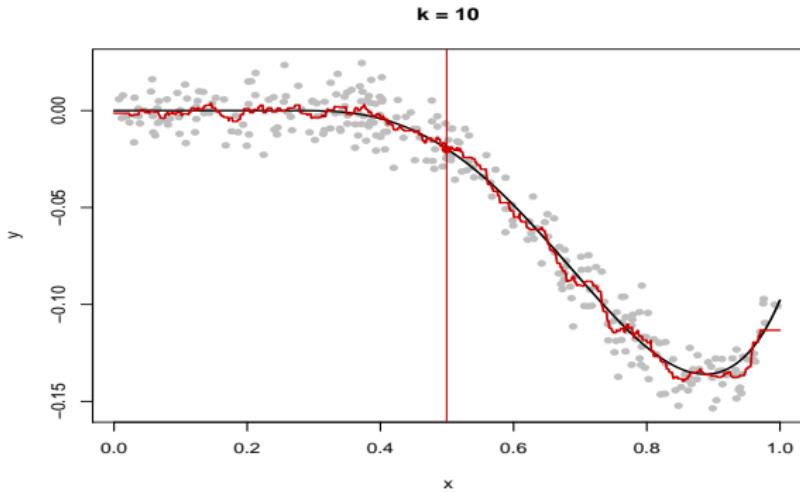
$$\mathbb{E}\left(\left(Y - \hat{f}(X)\right)^2 | X = x\right) = \underbrace{\left(f(x) - \hat{f}(x)\right)^2}_{\text{réductible}} + \underbrace{\text{Var}(\varepsilon | X = x)}_{\text{irréductible}}$$

## estimation de $f$ ?

- ▶ Nous n'avons aucune observation avec  $x = 0.5$
- ▶ On ne sait pas calculer  $\mathbb{E}(Y|X = x)$  !
- ▶ Approximation

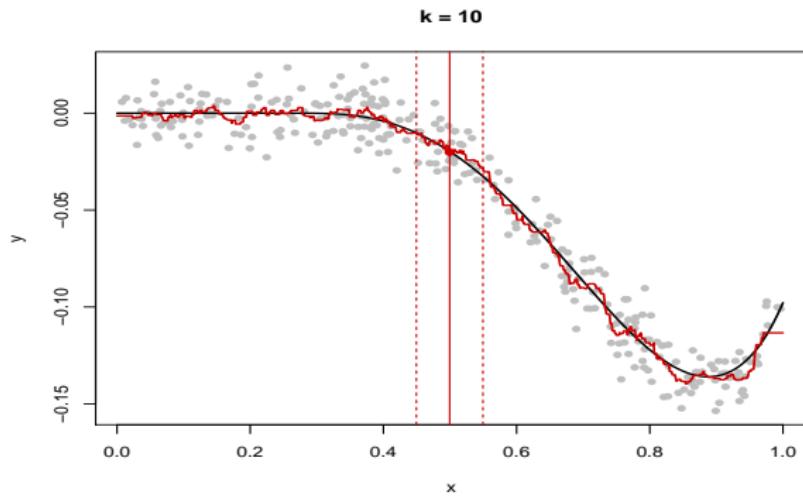
$$\hat{f}(x) = \text{Moyenne}\left(Y, X \in \mathcal{N}(x)\right)$$

où  $\mathcal{N}(x)$  représente un certain voisinage de  $x$ .

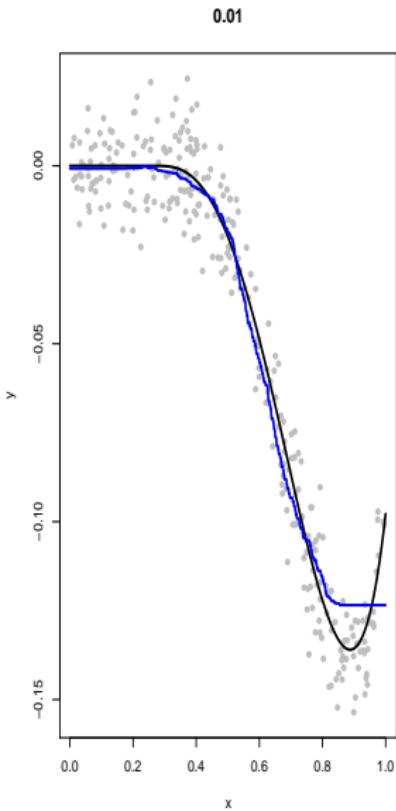
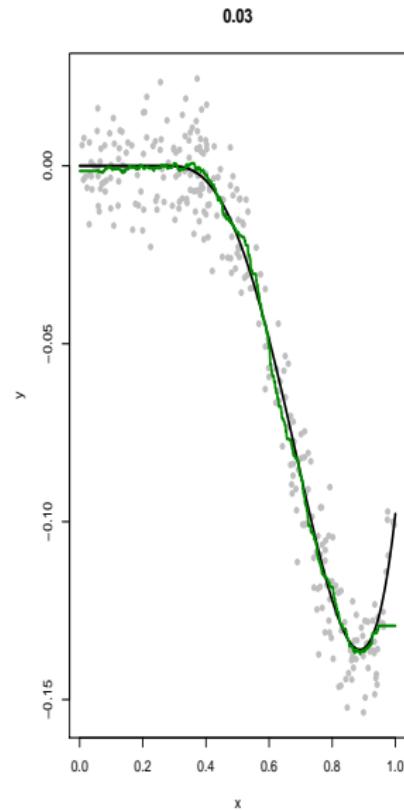
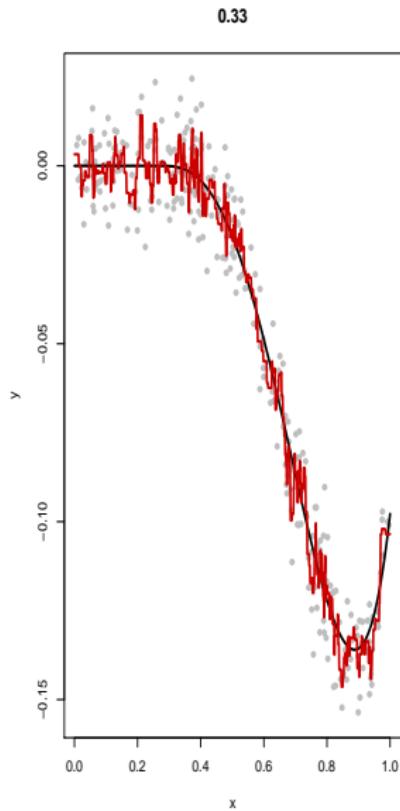


Une fonction idéale  $f(X)$  ? pour une certaine valeur de  $X$ , disons 0.5 ?

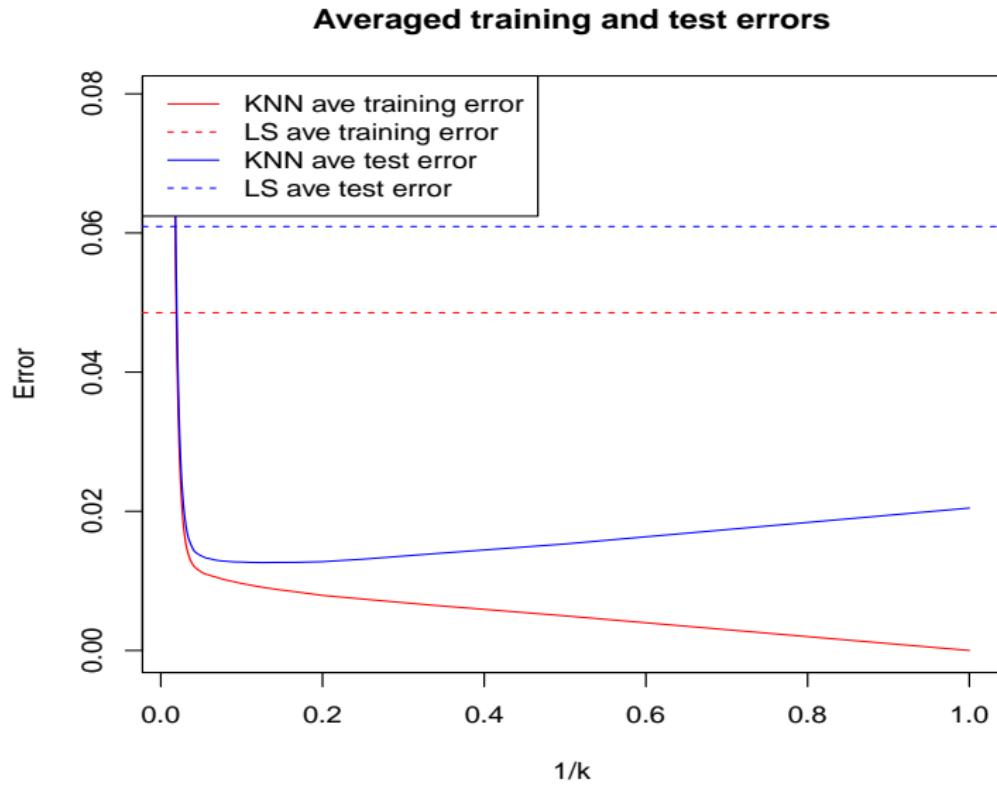
graphiquement



# Complexité d'un modèle (compromis biais variance)



# Évaluation de la précision : phénomène de sur-apprentissage



# Évaluer la précision : premier pas

Supposons que l'on ajuste un modèle  $\hat{f}(x)$  sur des données d'apprentissage  $\text{Tr} = \{(x_1, y_1), \dots, (x_N, y_N)\}$ .

## Performance de $\hat{f}$ ?

Première idée : erreur moyenne de prédiction sur  $\text{Tr}$  :

$$\text{MSE}_{\text{Tr}} = \text{Moyenne}_{i \in \text{Tr}} \left( y_i - \hat{f}(x_i) \right)^2$$

## OPTIMISTE

(sur-apprentissage)

Meilleure idée : sur un jeu de données de *test*,  
 $\text{Te} = \{(x_{N+1}, y_{N+1}), \dots, \}$ , indépendant de  $\text{Tr}$  :

$$\text{MSE}_{\text{Te}} = \text{Moyenne}_{i \in \text{Te}} \left( y_i - \hat{f}(x_i) \right)^2$$

# Apprentissage : problème de classification

Ici la variable réponse  $Y$  est qualitative :

le type d'une tumeur  $\mathcal{C} = \{\text{maligne}, \text{ bénigne}\}$

Les objectifs sont

- ▶ Construire une règle de classification  $C(X)$  qui permet d'associer un "label" de  $\mathcal{C}$  pour une nouvelle observation  $X$  sans label connu a priori.
- ▶ Évaluer l'erreur de classification de la règle
- ▶ Comprendre le rôle des chaque variable explicative  $X = (X_1, X_2, \dots, X_p)$  dans la règle de classification

# Un peu de formalisme

Supposons qu'on a le cas général où  $\mathcal{C}$  possède  $K$  labels numérotés de 1 à  $K$ , on définit les probabilités conditionnelles

$$p_k(x) = \mathbb{P}(Y = k | X = x), \quad k = 1, 2, \dots, K.$$

La règle de classification dite de **Bayes** en un certain point  $x_0$  est donnée par

$$C(x_0) = j \quad \text{si} \quad j = \arg \max_{k \in \{1, \dots, K\}} p_k(x_0)$$

Une règle de classification basée sur les plus proches voisins

$$p_k(x_0) = \frac{1}{|\mathcal{N}_0|} \sum_{i \in \mathcal{N}_0} I(y_i = k)$$

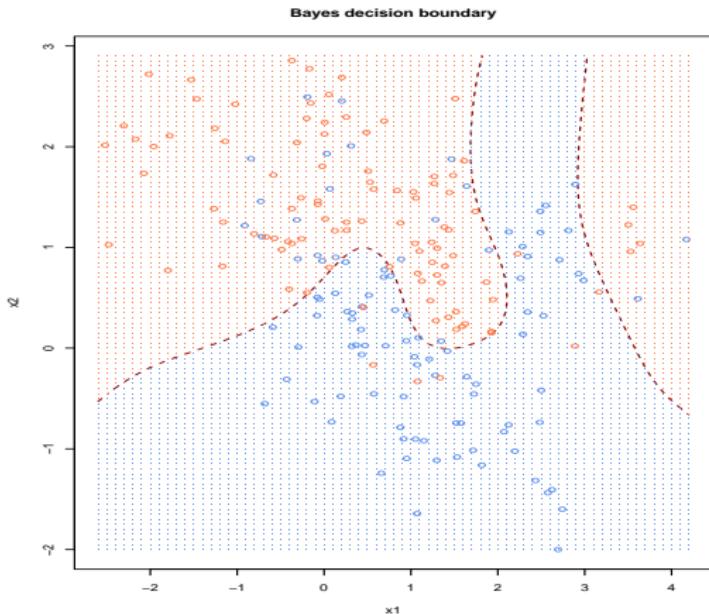
## Un peu de formalisme : évaluation

- ▶ La mesure de performance d'une règle de classification  $\widehat{C}(x)$  est donnée par l'erreur de classification

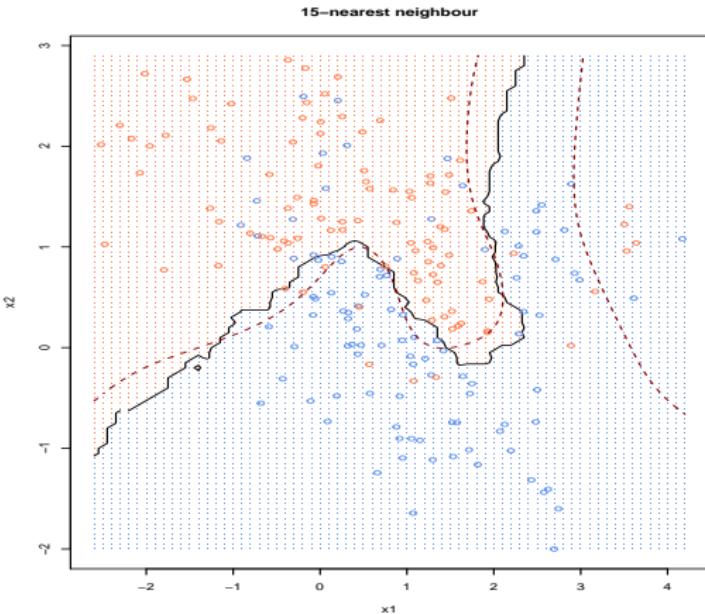
$$\text{Err}_{\text{Te}} = \text{Moyenne}_{i \in \text{Te}} I(y_i \neq \widehat{C}(x_i))$$

- ▶ La règle de classification de Bayes (avec les vraies  $p_k(x)$ ) réalise la plus petite erreur de classification dans la population.
- ▶ Les svm (*Support Vector Machines*) construisent des modèles structurés pour  $C(x)$
- ▶ La régression logistique, les modèles comme la LDA QDA proposent des constructions de modèles structurés pour  $p_k(x)$

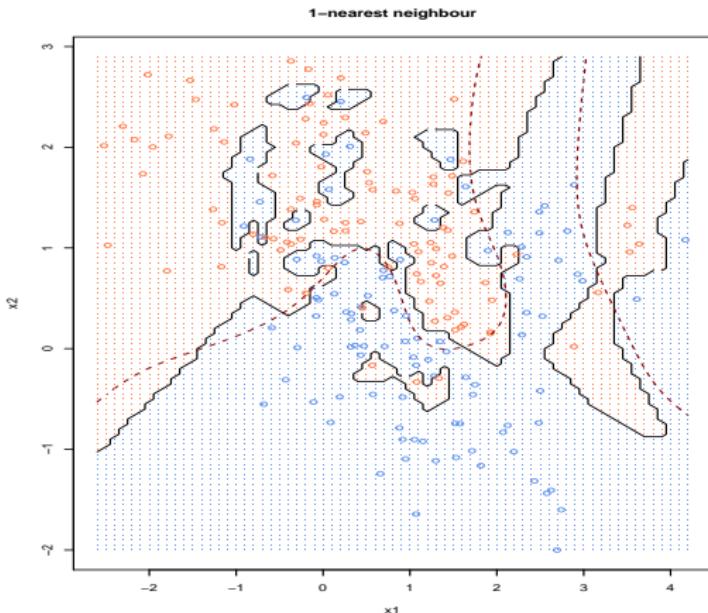
# Jeu de données avec la frontière issue de la règle de Bayes



# Règle de classification knn $k = 15$



# Règle de classification knn $k = 1$



# Les Knn sont victimes du fléau de la dimension

- ▶ Ces méthodes, basées sur des moyennes autour des voisins sont plutôt bonnes si
  - petite dimension  $p \leq 4$
  - grand échantillon  $n \gg p$
- ▶ des versions lissées, obtenues par
  - méthodes à noyaux
  - lissage par splines,
  - ...

**Raison.** le *fléau de la dimension*.

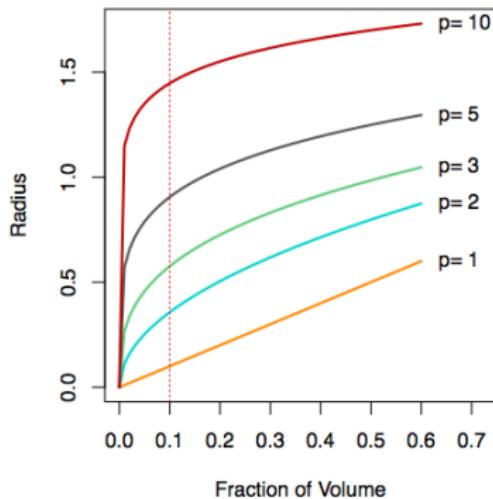
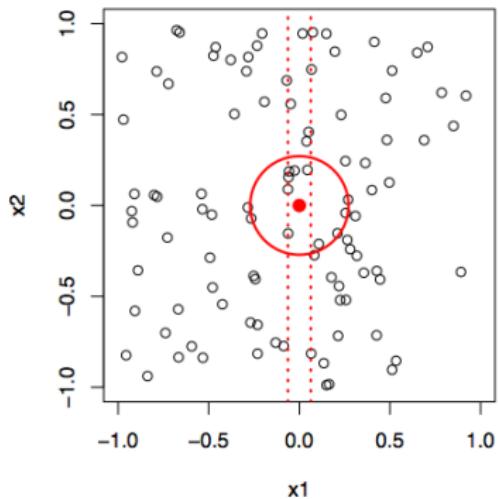
Les voisins les plus proches peuvent être éloignés en grande dimension

- ▶ Il faut une quantité raisonnable de valeurs de  $y_i$  à moyenner pour que  $\hat{f}(x)$  ait une faible variance
- ▶ En grande dimension, pour obtenir cette quantité d'observation, il faut s'éloigner beaucoup de  $x$ .

On perd l'idée de moyenne **locale** autre de  $X = x$ .

# Le fléau de la dimension

10% Neighborhood



# Validation croisée et bootstrap

## But

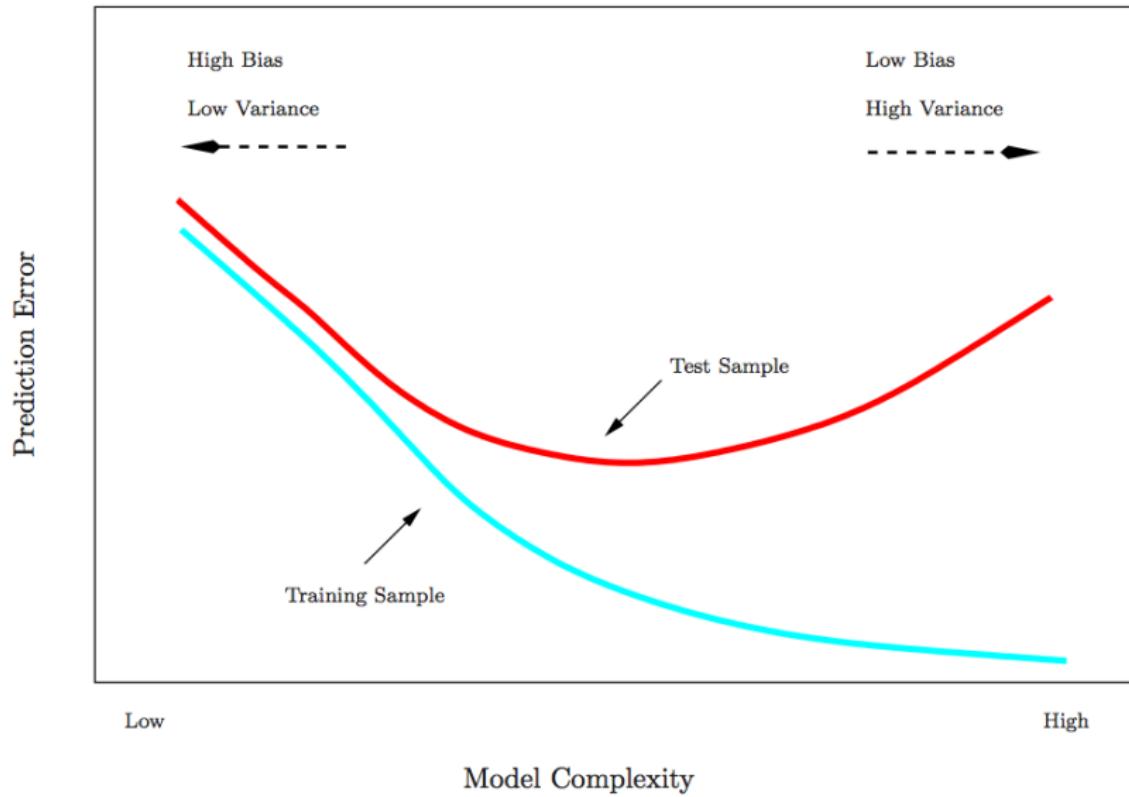
- ▶ Dans cette partie, nous allons discuter de deux méthodes de *ré-échantillonnage* : la validation croisée et le bootstrap
- ▶ Ces méthodes ré-ajustent le modèle que l'on souhaite sur des échantillons issus de l'échantillon d'apprentissage, dans le but d'obtenir des informations supplémentaires sur ce modèle
- ▶ Par exemples, ces méthodes fournissent des estimations de l'erreur sur des ensembles de test, le biais et la variance des estimations de paramètres... .

# Erreurs d'entraînement et erreur de test

On rappelle la différence entre *erreur de test* et *erreur d'entraînement* :

- ▶ L'*erreur de test* est l'erreur moyenne commise par une méthode d'apprentissage statistique pour prédire une réponse sur une nouvelle observation, qui n'a pas été utilisée pour ajuster le modèle.
- ▶ En revanche, l'*erreur d'entraînement* peut être facilement calculée en appliquant la méthode d'apprentissage sur les données d'entraînement.
- ▶ Mais l'erreur d'entraînement est souvent bien différente de l'erreur de test, et en particulier, l'erreur d'entraînement sous-estime parfois grandement l'erreur de test — on parle d'erreur trop *optimiste*.

# Erreurs d'entraînement et erreur de test



# Estimations de l'erreur de prédiction

- ▶ La meilleure solution : un grand ensemble de test clairement désigné. Bien souvent, ce n'est pas disponible.
- ▶ Certaines méthodes permettent de corriger l'erreur d'entraînement pour estimer l'erreur de test, avec des arguments fondés mathématiquement.  
Cela inclut les *C<sub>p</sub> de Mallows*, les critères AIC et BIC.  
Ils seront discutés plus tard.
- ▶ Ici, nous nous intéressons à une classe de méthodes qui estime l'erreur de test en mettant de côté un sous-ensemble des données d'entraînement disponibles pour ajuster les modèles, et en appliquant la méthode ajustée sur ces données mises de côté.

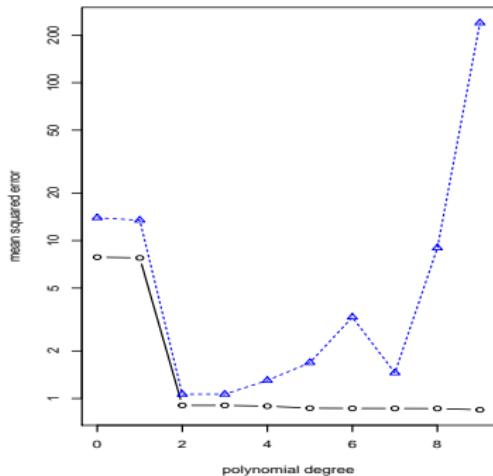
## Approche par ensemble de validation

- ▶ Cette méthode propose de diviser l'échantillon d'apprentissage en deux : un ensemble d'entraînement et un ensemble de validation
- ▶ Le modèle est ajusté sur l'ensemble d'entraînement, et on l'utilise ensuite pour prédire les réponses sur l'échantillon de validation.
- ▶ L'erreur obtenue en comparant prédition et observation sur cet échantillon de validation approche l'erreur de test. On utilise typiquement des moindres carrés (MSE) en régression et des taux de mauvaises classification si la réponse est qualitative (ou une fonction de coût d'erreur)

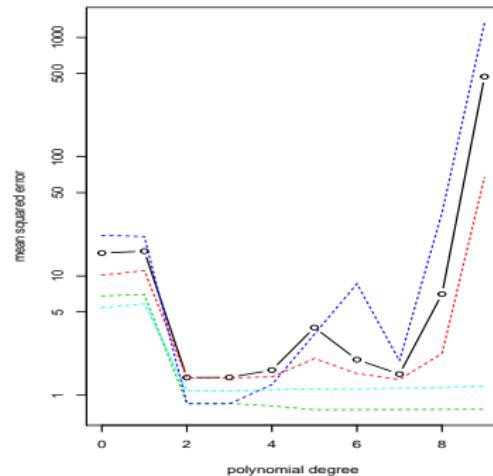


# Exemple sur les données simulées (degré 2)

- ▶ On veut comparer la régression linéaires à des régressions polynomiales de différents degrés
- ▶ On divise en deux les 200 observations : 100 pour l'entraînement, 100 pour le test.



Sur une partition aléatoire



Variabilité d'une partition à l'autre

## Inconvénients de l'approche par ensemble de validation

- ▶ L'estimation obtenue par cette méthode peut être très variable, et dépend de la chance ou malchance dans la construction du sous-échantillon de validation
- ▶ Dans cette approche, seule une moitié des observations est utilisée pour ajuster les modèles — celles qui sont dans l'ensemble d'entraînement.
- ▶ Cela suggère que l'erreur calculée peut surestimer l'erreur de test d'un modèle ajusté sur l'ensemble des données (moins de variabilité d'échantillonnage dans l'inférence des paramètres du modèle)

Déjà mieux : échanger les rôles entraînement-validation et faire la moyenne des deux erreurs obtenues. On *croise* les rôles.

# Validation croisée à $K$ groupes

- ▶ C'est la méthode la plus couramment utilisée pour estimer l'erreur de test
- ▶ L'estimation peut être utilisée pour choisir le meilleur modèle (la meilleure méthode d'apprentissage), ou approcher l'erreur de prédiction du modèle finalement choisi.
- ▶ L'idée est de diviser les données en  $K$  groupes de même taille. On laisse le  $k$ -ème bloc de côté, on ajuste le modèle, et on l'évalue sur le bloc laissé de côté.
- ▶ On répète l'opération en laissant de côté le bloc  $k = 1$ , puis  $k = 2, \dots$  jusqu'à  $k = K$ . Et on combine les résultats

1	2	3	4	5
Validation	Train	Train	Train	Train

# Détails

Pour chacune des observations, on obtient une prédition

$\hat{y}_i = \hat{f}(x_i)$  ou  $\hat{C}(x_i)$  au moment où  $i$  est dans le groupe mis de côté, et une seule prédition.

On compare alors ces prédictions aux observations comme pour l'erreur de test

$$MSE_{(K)} = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2$$

ou

$$\tau_{(K)} = \frac{1}{n} \sum_{i=1}^n \mathbf{1}\{y_i - \hat{C}(x_i)\}$$

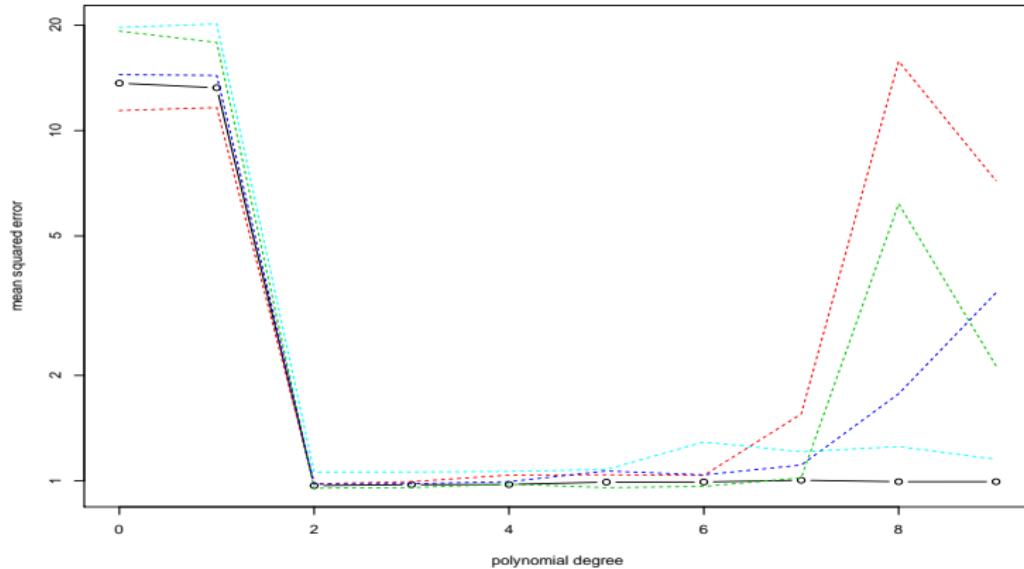
Lorsque  $K = n$ , on parle de  
« leave-one out  
cross-validation » (LOOCV)

## Danger avec le leave-one out !

On dit que LOOCV ne secoue pas assez les données. En effet, les classifiers  $\hat{C}$  ou les fonctions de régression inférées  $\hat{f}$  avec  $(n - 1)$  données sont très corrélés les uns aux autres.

On ne voit plus l'erreur d'échantillonnage, autrement dit la variabilité de l'estimation de la fonction. C'était pourtant tout l'intérêt de la validation croisée. On choisit généralement  $K = 5$  ou  $K = 10$  blocs.

# Retour au jeu de données simulé



En cas d'égalité, choisir le modèle le plus *parcimonieux* car il aura naturellement moins de variance d'estimation dans les coefficients du modèle.

## Validation croisée : les pièges

Considérons un classifieur simple pour prédire une réponse  $Y$  binaire, mais avec de nombreuses co-variables ( $p = 5000$ ). On procède comme suit

1. On démarre avec les 5000 prédicteurs et un échantillon de taille 50, et on cherche les 100 prédicteurs qui ont la plus grande corrélation par la réponse
2. On utilise une méthode d'apprentissage, par exemple la régression logistique, sur ces 100 meilleures co-variables.

Comment doit-on estimer l'erreur de test ?

Peut-on utiliser la validation croisée à l'étape 2 ???

# Réponse : NON, NON et NON !

1. On démarre avec les 5000 prédicteurs et un échantillon de taille 50, et on cherche les 100 prédicteurs qui ont la plus grande corrélation par la réponse
  2. On utilise une méthode d'apprentissage, par exemple la régression logistique, sur ces 100 meilleures co-variables.
- ▶ Cela ignore le fait que l'étape 1 a déjà utiliser les réponses observées. Cette étape est une forme d'entraînement du classifier final.
  - ▶ Il est facile de simuler des données réalistes, avec  $Y$  indépendant de  $X$  (dont l'erreur de test doit être de 50%) mais dont l'erreur calculée par CV dans l'étape 2 est proche de 0 !

*Essayer de le faire vous même !*
  - ▶ Cette erreur est pourtant comise dans de nombreux articles traitant de données génomiques !

# Bootstrap

- ▶ Le *bootstrap* est un outil statistique flexible et puissant qui peut être utilisé pour quantifier l'incertitude associée à la valeur d'un estimateur, ou d'une méthode statistique.
- ▶ Par exemple, elle fournit une approximation de la variance de l'estimateur d'un coefficient, ou un intervalle de confiance pour ce coefficient.

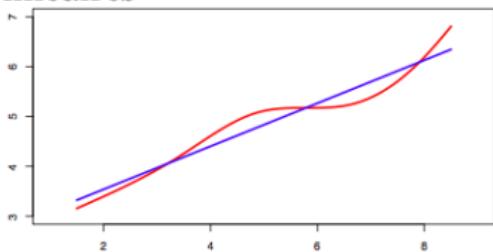
## D'où vient ce nom ?

- ▶ De l'expression « *to pull oneself up by one's bootstraps* », dont on pense que l'origine provient du livre *The Surprising Adventures of Baron Munchausen* de Rudolph Erich Raspe : « The Baron had fallen to the bottom of a deep lake. Just when it looked like all was lost, he thought to pick himself up by his own bootstraps. »
- ▶ Ne pas confondre avec la séquence d'amorçage des ordinateurs, pour laquelle on parle parfois de « *bootstrap* ».

# Sélection de variables en régression linéaire

# Régression linéaire : rappel

- ▶ Une approche simple pour faire de l'apprentissage supervisé. Elle suppose que  $Y$  dépend linéairement de  $X_1, \dots, X_p$
- ▶ Les vraies fonctions de régression ne sont jamais linéaires



- ▶ Même si cela semble trop simple, la régression linéaire est extrêmement utile à la fois conceptuellement et en pratique.

## Le cas de régression linéaire simple

- ▶ On pose un modèle de la forme

$$Y = \beta_0 + \beta_1 X + \varepsilon,$$

où  $\beta_0, \beta_1$  inconnus sont ordonnée à l'origine (intercept) et pente (slope). Ce sont les *coefficients* du modèle

- ▶ Étant estimés ces coefficients par  $\widehat{\beta}_0$  et  $\widehat{\beta}_1$ , on prédit  $y$  sachant  $x$  avec

$$\widehat{y} = \widehat{\beta}_0 + \widehat{\beta}_1 x,$$

# Régression linéaire multiple

- ▶ Notre modèle

$$Y = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p + \varepsilon$$

- ▶ On interprète  $\beta_j$  comme l'effet moyen sur  $Y$  d'un accroissement de  $X_j$  d'une unité *lorsque tous les autres prédicteurs sont fixés.*
- ▶ On ne peut faire aucune affirmation en terme de *causalité*.

Exemple.  $Y = \text{serum triglycerides mg/dl}$ ,  
 $A = \text{age in years}$  et  $B = \text{body-mass index, kg/m}^2$ .

$$\hat{Y} = -247.25 + 3.5A + 9.3B.$$

Comment s'interprète 9.3 ?

## Interpréter les coefficients de régression

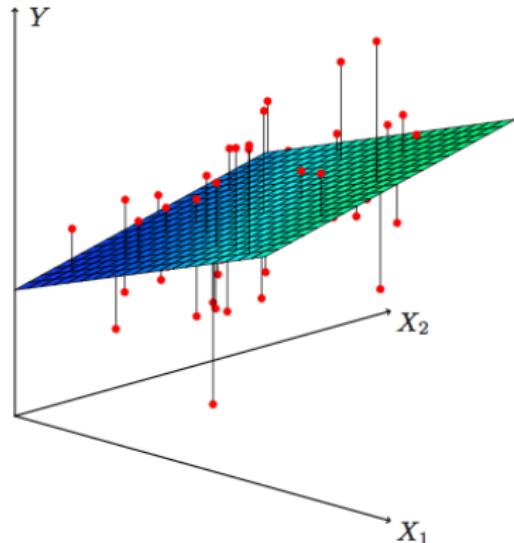
- ▶ Le scenario idéal lorsque les prédicteurs sont indépendants, et le design équilibré
  - ▶ chaque coeff peut être estimé et testé séparément
  - ▶ interprétation de gauche est OK
- ▶ La corrélation entre  $X_j$  pose des problèmes
  - ▶ la variance des estimateurs s'accroît
  - ▶ l'interprétation devient hasardeuse (lorsque  $X_j$  change, tout change !)

# Estimation et prédition pour la régression multiple

- ▶ À partir d'estimation des coef  $\widehat{\beta}_0, \widehat{\beta}_1, \dots, \widehat{\beta}_p$ , on peut prédire avec

$$\widehat{y} = \widehat{\beta}_0 + \widehat{\beta}_1 x_1 + \dots + \widehat{\beta}_p x_p$$

- ▶ Comme en dimension 1, on estime les  $\beta$  en minimisant la somme des carrés résiduelle. Formule théorique qui dépend d'une inversion de matrice produit. → utiliser un logiciel de statistique
- ▶ De même,  $SE(\beta_j)$  pour chaque coefficient,  $t$ -test de nullité, test de Fisher,...



# Résultats pour les données triglycérides

	Coeff	Std.Err	t-stat	p-value
Intercept	-247.25	21.24	-11.64	<2e-16
BMI	9.30	0.90	10.32	<2e-16
age	3.50	0.19	18.69	<2e-16

	Corrélations		
	TG	BMI	age
TG	1.00	0.56	0.73
BMI		1.00	0.34
age			1.00

# Quelques rappels

On note

- ▶ La somme des carrés totale  $SST = SYY = \sum_{i=1}^n (y_i - \bar{y})^2$
- ▶ La somme des carrés résiduelle  $RSS = \sum_{i=1}^n (y_i - \hat{y}_i)^2$ , où

$$\hat{\mathbf{Y}} = \mathbf{X}\hat{\boldsymbol{\beta}} \quad \hat{\boldsymbol{\beta}} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{Y}$$

- ▶ La somme des carrés expliquée par la régression

$$SSreg = \sum_{i=1}^n (\hat{y}_i - \bar{y})^2$$

# Des questions importantes

1. Y a t-il au moins un des  $X_j$  utile pour prédire  $Y$  ?
2. Sont-ils vraiment tous utiles ?
3. Comment le modèle s'ajuste aux données ?
4. Avec une nouvelle valeur de  $X$ , quelle réponse doit-on prédire ? Précision de la prédiction ?

# Des questions importantes

1. Y a t-il au moins un des  $X_j$  utile pour prédire  $Y$  ?
2. Sont-ils vraiment tous utiles ?
3. Comment le modèle s'ajuste aux données ?
4. Avec une nouvelle valeur de  $X$ , quelle réponse doit-on prédire ? Précision de la prédiction ?

Pour la première question, on utilise la  $F$ -statistique

$$F = \frac{(SST - RSS)/p}{RSS/(n - p - 1)} \sim F_{p, n-p-1}$$

Quantité	Valeur
Residual Std.Err	50.34
$R^2$	0.63
$F$ -stat	343.7

# Des questions importantes

1. Y a t-il au moins un des  $X_j$  utile pour prédire  $Y$  ?
  2. **Sont-ils vraiment tous utiles ?**
  3. Comment le modèle s'ajuste aux données ?
  4. Avec une nouvelle valeur de  $X$ , quelle réponse doit-on prédire ? Précision de la prédiction ?
- ▶ Choix de co-variables : approche complète Comparer les modèles linéaires avec tous les sous-ensembles possibles de co-variables
  - ▶ Souvent  $2^p$  trop grand ( $\log_{10}(2^{40}) \approx 12.0$ ) On utilise une méthode qui ne parcourt que certains sous-ensembles. Deux approches standard Sélections progressive, ou rétrograde
  - ▶ Nécessite de répondre à la question suivante pour effectuer la comparaison.

# Choix de co-variables

## Méthode progressive (forward)

1. Commencer par le modèle nul (à zéro co-variables)
2. Ajuster les  $p$  régressions linéaires simples et ajouter au modèle nul la co-variable qui à le plus petit RSS
3. Ajouter à ce modèle à une co-variable la co-variable qui fait baisser le plus le RSS
4. Continuer jusqu'à un critère d'arrêt (par exemple sur la  $p$ -value du  $t$ -test)

## Méthode rétrograde (backward)

1. Commencer par le modèle avec tous les co-variables
2. Supprimer la variable avec la plus grande  $p$ -value —i.e., la co-variable la moins significative pour le modèle
3. Ré-ajuster le modèle, et enlever de nouveau la co-variable de plus grande  $p$ -value
4. Continuer jusqu'à un critère d'arrêt (par exemple portant sur la valeur de la  $p$ -value de la co-variable que l'on enlèverait)

# Choix de co-variables

- ▶ Critère plus systématique pour choisir le modèle « optimal » dans ceux que l'on parcourt
- ▶ Avec  $C_p$  de Mallows, Akaike information criterion (AIC), Bayesian information criterion (BIC),  $R^2$  ajusté et validation croisée (CV)

## Évaluer un sous ensembles de covariables

On supposera dans la suite que nous avons  $m$  covariables au total et on entend par modèle un sous-ensemble de ces covariables de taille  $p$ .

# Évaluer un sous ensembles de covariables

On supposera dans la suite que nous avons  $m$  covariables au total et on entend par modèle un sous-ensemble de ces covariables de taille  $p$ .

Rappelons que

$$R^2 = \frac{\text{SSreg}}{\text{SST}} = 1 - \frac{\text{RSS}}{\text{SST}}$$

et

$$R_{\text{adj}}^2 = 1 - \frac{\text{RSS}/(n-p-1)}{\text{SST}/(n-1)}$$

où  $p$  est le nombre de variables du modèle.

- ▶ On sélectionne le modèle avec le  $R_{\text{adj}}^2$  le plus élevé, cela revient à choisir le sous-ensemble de variables qui minimise

$$S^2 = \frac{\text{RSS}}{n-p-1}$$

où  $p$  est le nombre de variables du modèle.

## Choix basé sur $R_{\text{adj}}^2$

- ▶ Souvent le choix basé sur  $R_{\text{adj}}^2$  montre un phénomène de *over-fitting*.
- ▶ Supposons que la valeur maximale de  $R_{\text{adj}}^2 = 0.692$  pour un sous-ensemble  $p = 10$  de covariables,  $R_{\text{adj}}^2 = 0.691$  pour  $p = 9$  et  $R_{\text{adj}}^2 = 0.541$  pour un sous-ensemble de  $p = 8$  covariables.
- ▶ Il est clairement préférable de choisir le modèle à  $p = 10$  covariables.

## Choix basé sur $R_{\text{adj}}^2$

- ▶ Souvent le choix basé sur  $R_{\text{adj}}^2$  montre un phénomène de *over-fitting*.
- ▶ Supposons que la valeur maximale de  $R_{\text{adj}}^2 = 0.692$  pour un sous-ensemble  $p = 10$  de covariables,  $R_{\text{adj}}^2 = 0.691$  pour  $p = 9$  et  $R_{\text{adj}}^2 = 0.541$  pour un sous-ensemble de  $p = 8$  covariables.
- ▶ Il est clairement préférable de choisir le modèle à  $p = 10$  covariables.

On va faire appel à un critère(s) basé(s) sur la vraisemblance.

# La vraisemblance

Rappelons que d'après les hypothèses du modèle linéaire

$$Y_i \mid x_{i1}, \dots, x_{ip} \sim \mathcal{N}(\beta_0 + \beta_1 x_{i1} + \dots + \beta_p x_{ip}, \sigma^2).$$

Et

$$f(y_i \mid x_{i1}, \dots, x_{ip}) = \frac{1}{\sigma\sqrt{2\pi}} \exp \left[ -\frac{(y_i - (\beta_0 + \beta_1 x_{i1} + \dots + \beta_p x_{ip}))^2}{2\sigma^2} \right]$$

# La vraisemblance

Rappelons que d'après les hypothèses du modèle linéaire

$$Y_i \mid x_{i1}, \dots, x_{ip} \sim \mathcal{N}(\beta_0 + \beta_1 x_{i1} + \dots + \beta_p x_{ip}, \sigma^2).$$

Et

$$f(y_i \mid x_{i1}, \dots, x_{ip}) = \frac{1}{\sigma\sqrt{2\pi}} \exp \left[ -\frac{(y_i - (\beta_0 + \beta_1 x_{i1} + \dots + \beta_p x_{ip}))^2}{2\sigma^2} \right]$$

- ▶ Écrire la vraisemblance

$$L(\beta_0, \beta_1, \dots, \beta_p, \sigma^2; y_1, \dots, y_n)$$

- ▶ Écrire la log-vraisemblance

$$\ell(\beta_0, \beta_1, \dots, \beta_p, \sigma^2; y_1, \dots, y_n)$$

## Mesurer l'ajustement ou l'adéquation du modèle

L'estimateur par maximum de vraisemblance de  $\sigma^2$  est donné par

$$\hat{\sigma}_{\text{MLE}}^2 = \frac{\text{RSS}}{n}.$$

$$\ell(\hat{\beta}_0, \hat{\beta}_1, \dots, \hat{\beta}_p, \hat{\sigma}_{\text{MLE}}^2; y_1, \dots, y_n) = -\frac{n}{2} \ln(2\pi) - \frac{n}{2} \ln\left(\frac{\text{RSS}}{n}\right) - \frac{n}{2}$$

# Critère d'information

Un critère d'information est une quantité qui réalise un compromis entre l'ajustement aux données (*la vraisemblance par exemple*) et la complexité du modèle.

Donc

- ▶ On peut chercher le modèle qui **maximise**  
*Ajustement - Pénalité*
- ▶ On peut chercher le modèle qui **minimise**  
*- Ajustement + Pénalité*

## Critère d'information $C_p$ de Mallows

Le critère d'information noté  $C_p$  de Mallows associé au modèle à  $p$  covariables est donné par

$$C_p = \frac{\text{RSS}_p}{S^2} + 2p - n$$

où  $\text{RSS}_p$  est la somme des carrés des résidus du modèle en question et  $S^2$  est l'estimateur de  $\sigma^2$  dans le modèle complet.

# Critère d'information AIC

Le critère d'information noté AIC (*Akaike's Information Criterion*) associé à un modèle à  $p$  covariables est donné par

$$\text{AIC} = -2\ell(\hat{\beta}_0, \hat{\beta}_1, \dots, \hat{\beta}_p, \hat{\sigma}_{\text{MLE}}^2; y_1, \dots, y_n) + 2K$$

où  $K = p + 2$  est nombre de paramètre du modèle.

On peut montrer que

$$\text{AIC} = n \log \hat{\sigma}_{\text{MLE}}^2 + 2p + \text{const.}$$

Ce critère est préférable pour la *prédition*.

# Critère d'information BIC

Le critère d'information noté BIC (*Bayesian Information Criterion*) associé à un modèle à  $p$  covariables est donné par

$$\text{BIC} = -2\ell(\hat{\beta}_0, \hat{\beta}_1, \dots, \hat{\beta}_p, \hat{\sigma}_{\text{MLE}}^2; y_1, \dots, y_n) + K \ln(n)$$

où  $K = p + 2$  est nombre de paramètre du modèle.

- ▶ Ce critère possède de *bonnes propriétés théoriques*.
- ▶ Ce critère est préférable pour *l'explication*.

## Sélection de modèle

L'idée de chercher le modèle (un sous-ensemble de covariables) qui minimise un des trois critères précédents.

Comment procéder ?

# Sélection de modèle

L'idée de chercher le modèle (un sous-ensemble de covariables) qui minimise un des trois critères précédents.

Comment procéder ?

Recherche exhaustive : calculer la valeur du critère pour chaque modèle.

# Sélection de modèle

L'idée de chercher le modèle (un sous-ensemble de covariables) qui minimise un des trois critères précédents.

Comment procéder ?

Recherche exhaustive : calculer la valeur du critère pour chaque modèle.

Problème combinatoire :  $2^{\text{le nombre de covariables}}$  modèles en compétition !!

## Sélection de modèle en forward

Le modèle de départ de la procédure de sélection *forward* est le modèle avec la constante seulement. La procédure consiste à

1. Ajouter séparément chaque variable au modèle actuel et calculer le critère d'intérêt (BIC, AIC, ou  $C_p$ ).
2. Si aucun des nouveaux modèles n'améliore le critère, alors : **stop**.
3. Mettre à jour le modèle en incluant la covariable qui apporte la meilleure amélioration au sens du critère. Aller à 1.

## Sélection de modèle backward

Le point de départ de la procédure d'élimination *backward* est le modèle complet incluant toutes les covariables. La procédure consiste à

- 1.** Si aucune élimination d'une covariable n'améliore le critère alors : **stop**.
- 2.** Mettre à jour le modèle en éliminant la covariable qui réalise la meilleure amélioration du critère. Aller à **1**.

# Données cancer de la prostate

<b>lcavol</b>	log(cancer volume)
<b>lweight</b>	log(prostate weight)
<b>age</b>	age
<b>lbph</b>	log(benign prostatic hyperplasia amount)
<b>svi</b>	seminal vesicle invasion
<b>lcp</b>	log(capsular penetration)
<b>gleason</b>	Gleason score
<b>pgg45</b>	percentage Gleason scores 4 or 5
<b>lpsa</b>	log(prostate specific antigen)

Stamey, T.A., Kabalin, J.N., McNeal, J.E., Johnstone, I.M., Freiha, F., Redwine, E.A. and Yang, N. (1989). Prostate specific antigen in the diagnosis and treatment of adenocarcinoma of the prostate : II. radical prostatectomy treated patients, Journal of Urology 141(5), 1076–1083.

## Comparaison de ces critères

- ▶ Je déconseille le  $R^2$  ajusté.
- ▶  $C_p$  et AIC sont des critères qui réalisent un compromis biais-variance. Ils sont donc indiqués pour choisir un modèle que l'on souhaite utiliser pour prédire.
- ▶ BIC pénalise plus les modèles de grandes dimensions. C'est le seul critère à être consistant (i.e., à fournir un estimateur qui converge lorsque  $n \rightarrow \infty$ )
- ▶ BIC étant plus sélectif, on doit le préférer si l'on souhaite un modèle explicatif.
- ▶ Lorsque la taille de la base d'apprentissage est grande, préférer BIC (AIC fournit des modèles de trop grandes dimensions)

# Sélection de variables : quelques remarques

## Interprétabilité

- ▶ Si le vrai modèle ne contient que **quelques variables liées à la response**  $\rightsquigarrow$  les algorithmes de sélection peuvent retrouver les prédicteurs pertinents.
- ▶ Si le vrai modèle contient **beaucoup de variables très corrélées**  $\rightsquigarrow$  les variables sélectionnées seront difficiles à interpréter.

## Limites liées à la stabilité

En présence de prédicteurs très corrélés ou lorsque  $n < p$ , **de petites perturbations** des données peuvent provoquer **de grandes différences** entre les ensembles de variables sélectionnées.

# Méthode de shrinkage

## *Régression ridge et Lasso*

- ▶ Les méthodes précédentes de choix de sous-ensembles utilisent les moindres carrés pour ajuster chacun des modèles en compétition.
- ▶ Alternativement, on peut ajuster un modèle contenant toutes les  $p$  covariables en utilisant une technique que *constraint* ou *régularise* les estimations des coefficients, ou de façon équivalente, pousse les coefficients vers 0.
- ▶ Il n'est pas évidant de comprendre pourquoi de telles contraintes vont améliorer l'ajustement, mais il se trouve qu'elles réduisent la variance de l'estimation des coefficients.

# Régression ridge

- ▶ Rappelons que la procédure d'ajustement par moindres carrés estime les coefficients  $\beta_0, \beta_1, \dots, \beta_p$  en minimisant

$$\text{RSS}(\boldsymbol{\beta}) = \sum_{i=1}^n \left( y_i - \beta_0 - \sum_{j=1}^p \beta_j x_{ij} \right)^2.$$

- ▶ En revanche, la régression ridge estime les coefficients en minimisant

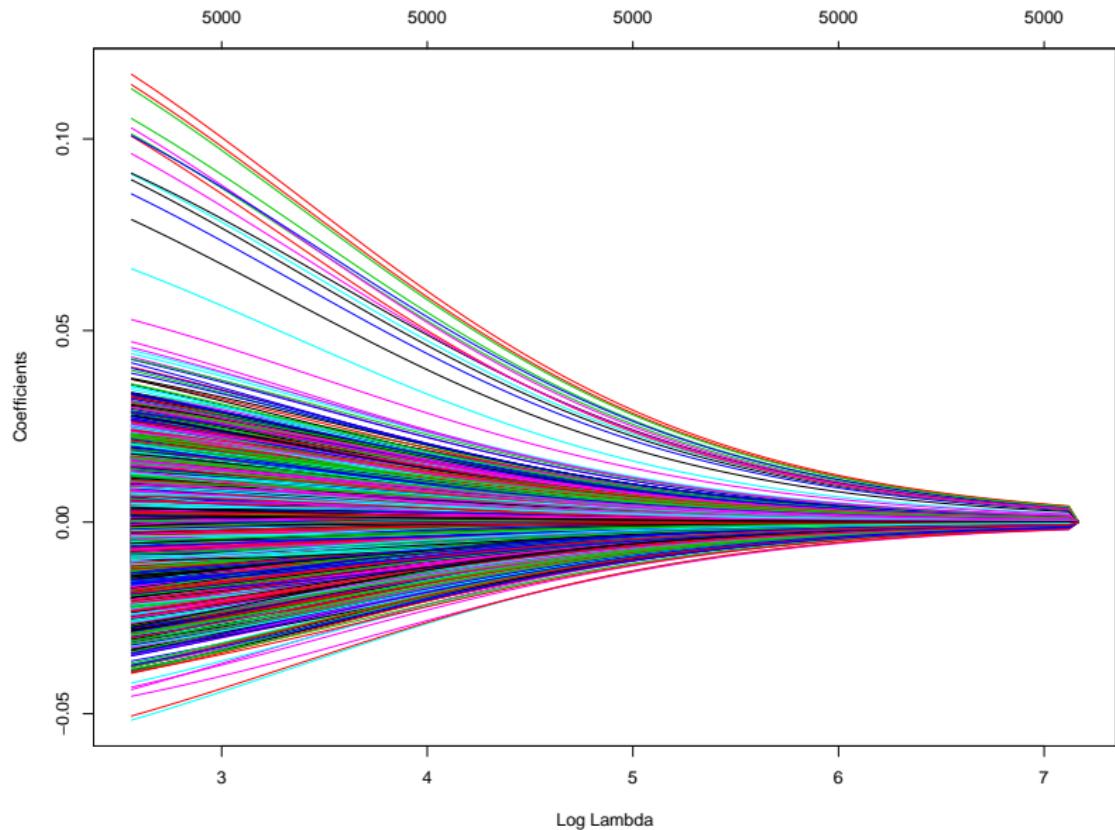
$$\sum_{i=1}^n \left( y_i - \beta_0 - \sum_{j=1}^p \beta_j x_{ij} \right)^2 + \lambda \sum_{j=1}^p \beta_j^2 = \text{RSS}(\boldsymbol{\beta}) + \lambda \sum_{j=1}^p \beta_j^2,$$

où  $\lambda$  est un *paramètre de réglage*, à déterminer par ailleurs.

## Régression ridge (suite)

- ▶ Comme les moindres carrés, la régression ridge cherche des estimations des coefficients qui s'ajustent sur les données, donc à rendre  $\text{RSS}(\boldsymbol{\beta})$  petit.
- ▶ Cependant, le second terme  $\lambda \sum_{j=1}^p \beta_j^2$ , appelé *pénalité ridge* est petit lorsque les  $\beta_j$  sont proches de 0, et tire donc les estimations vers ce point.
- ▶ Le paramètre de réglage  $\lambda$  sert à contrôler l'impact de cette pénalité sur l'estimation.
- ▶ Choisir une bonne valeur de  $\lambda$  est critique pour construire un modèle acceptable. On utilise la validation croisée.

# Exemple jeu de données $n = 1000$ et $p = 5000$

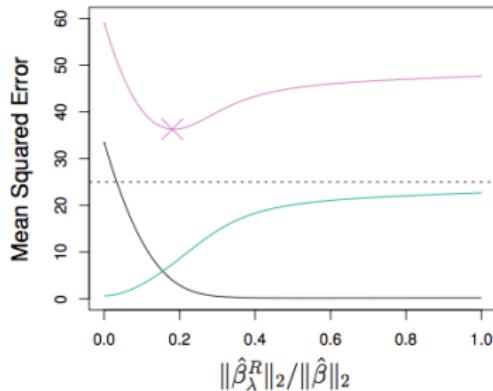
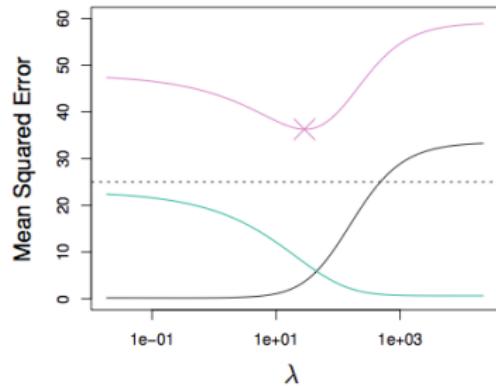


## Régression ridge : normaliser les prédicteurs

- ▶ La méthode des moindres carrés standard est insensible à la normalisation des prédicteurs : si l'on multiplie  $X_j$  par  $c$ , le coefficient sera remplacé par  $\hat{\beta}_j/c$ .
- ▶ En revanche, la régression ridge peut changer *substantiellement* lorsque l'on multiplie un prédicteur par une constante, à cause de la norme quadratique dans le terme de pénalité.
- ▶ C'est pourquoi il est vivement recommandé de toujours standardiser les prédicteurs (marginalement) avant d'utiliser la régression ridge.

# Pour la régression ridge ?

## Compromis biais-variance



Données simulées :  $n = 50$ ,  $p = 45$ , tous de coefficients non nuls. Biais au carré (en noir), variance (en vert) et erreur de test quadratique (en violet) pour la régression ridge.

Droite horizontale : erreur minimale.

# Le Lasso : *Least Absolute Shrinkage and Selection Operator*

- ▶ La régression ridge a un inconvénient évident : contrairement à la sélection de variable, la régression ridge inclut tous les prédicteurs dans le modèle final.
- ▶ Le Lasso est une alternative relativement récente qui répond à cette critique. On minimise en fait

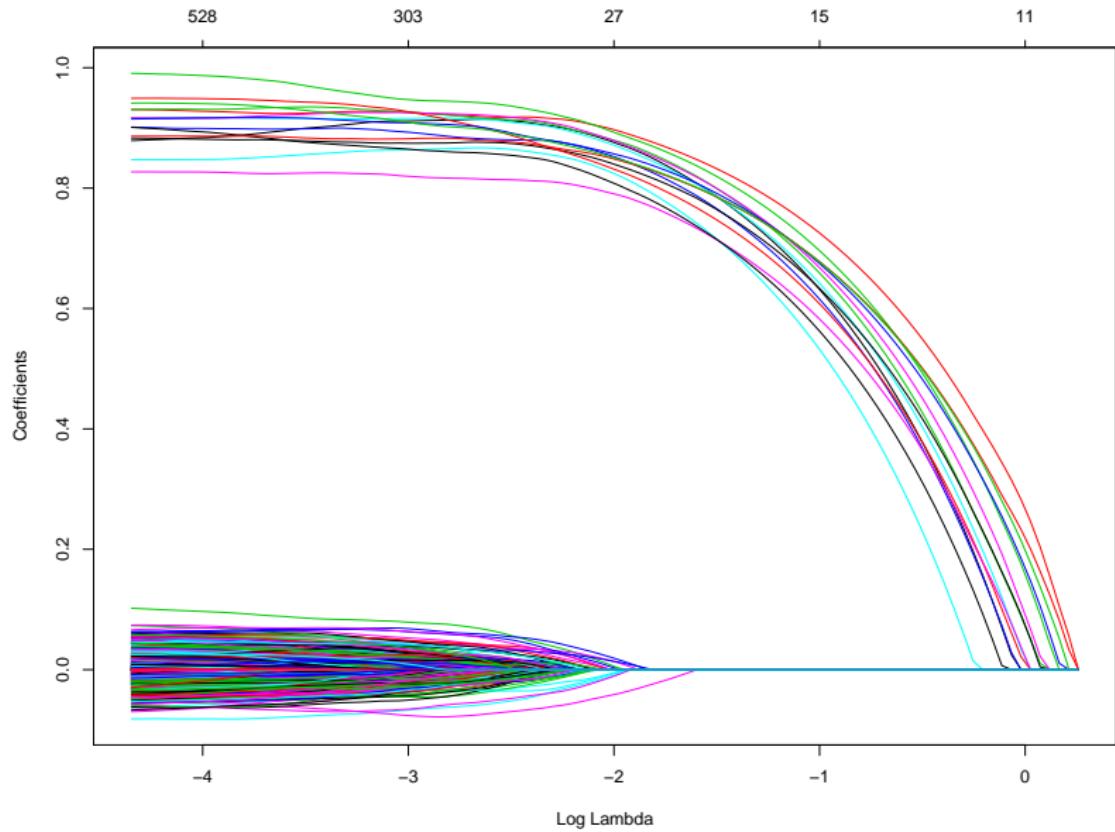
$$\sum_{i=1}^n \left( y_i - \beta_0 - \sum_{j=1}^p \beta_j x_{ij} \right)^2 + \lambda \sum_{j=1}^p |\beta_j| = \text{RSS}(\boldsymbol{\beta}) + \lambda \sum_{j=1}^p |\beta_j|,$$

- ▶ On parle de pénalité  $\ell^1$  au lieu de pénalité  $\ell^2$  (ou quadratique)

## Le Lasso (suite)

- ▶ Comme pour la régression ridge, le Lasso tire les estimations des coefficients vers 0.
- ▶ Cependant, dans le cas du Lasso, la pénalité  $\ell^1$  a pour effet de forcer certains coefficients à s'annuler lorsque  $\lambda$  est suffisamment grand.
- ▶ Donc, le Lasso permet de faire de la *sélection de variable*.
- ▶ On parle de modèle creux (sparse), c'est-à-dire de modèles qui n'impliquent qu'un sous ensemble des variables.
- ▶ Comme pour la régression ridge, choisir une bonne valeur de  $\lambda$  est critique. Procéder par validation ou validation croisée.

Exemple :  $n = 1000$  et  $p = 5000$



# Qu'est qui fait marcher le Lasso ?

Avec les multiplicateurs de Lagrange, on peut voir

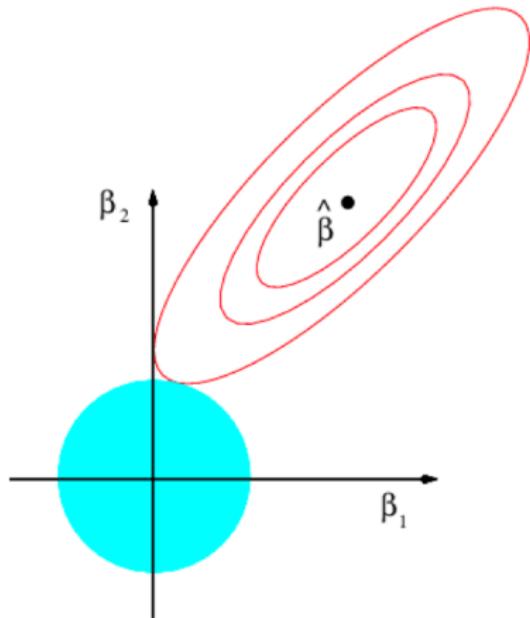
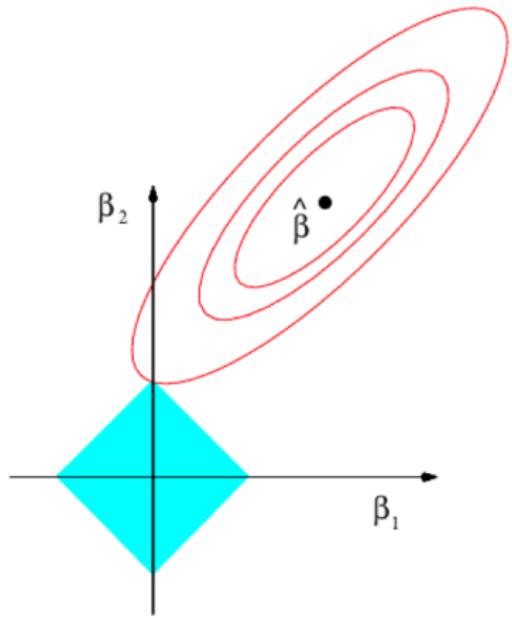
- ▶ La régression ridge comme

$$\text{minimise } \sum_{i=1}^n \left( y_i - \beta_0 - \sum_{j=1}^p \beta_j x_{ij} \right)^2 \text{ sous la contrainte } \sum_{j=1}^p \beta_j^2 \leq s$$

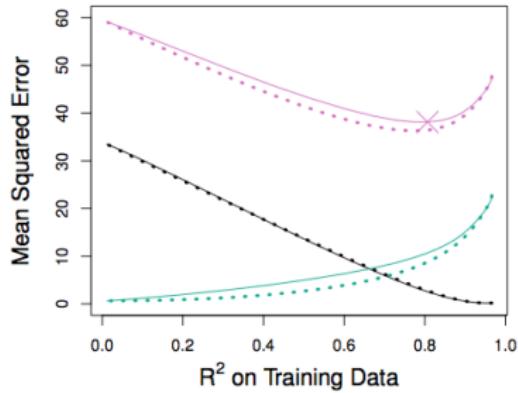
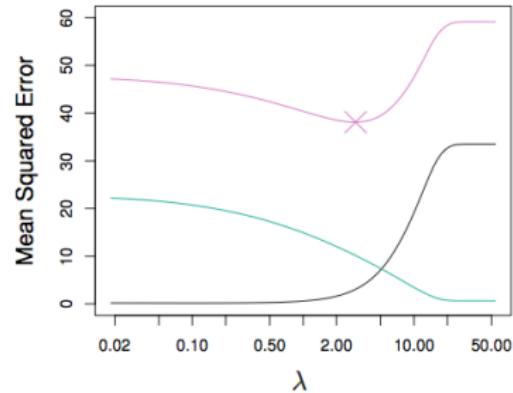
- ▶ Le Lasso comme

$$\text{minimise } \sum_{i=1}^n \left( y_i - \beta_0 - \sum_{j=1}^p \beta_j x_{ij} \right)^2 \text{ sous la contrainte } \sum_{j=1}^p |\beta_j| \leq s$$

# Le Lasso en image



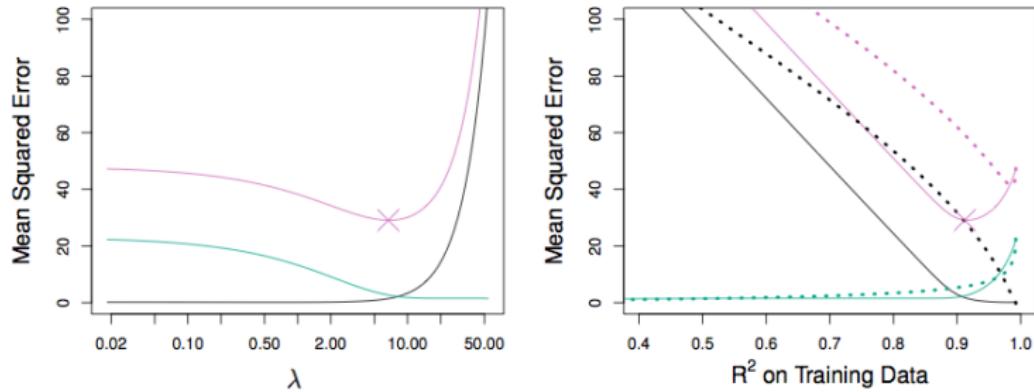
# Comparaison du Lasso et de la régression ridge



À gauche, biais au carré (noir), variance (en vert) et erreur quadratique de test (violet) pour le Lasso sur données simulées.

À droite, comparaison du biais au carré, de la variance et de l'erreur de test quadratique pour le Lasso (traits plats) et la régression ridge (pointillés)

# Comparaison du Lasso et de la régression ridge (suite)



À gauche, biais au carré (noir), variance (en vert) et erreur quadratique de test (violet) pour le Lasso sur données simulées (où seulement deux prédicteurs sont influents).

À droite, comparaison du biais au carré, de la variance et de l'erreur de test quadratique pour le Lasso (traits plats) et la régression ridge (pointillés)

# Conclusions

- ▶ Ces deux exemples montrent qu'il n'y a pas de meilleur choix universel entre la régression ridge et le Lasso.
- ▶ En général, on s'attend à ce que le Lasso se comporte mieux lorsque la réponse est une fonction d'un nombre relativement faible de prédicteurs.
- ▶ Cependant, le nombre de prédicteurs reliés à la réponse n'est jamais connu *a priori* dans des cas concrets.
- ▶ Une technique comme la validation croisée permet de déterminer quelle est la meilleure approche.

# Choisir le paramètre de réglage $\lambda$

- ▶ Comme pour les méthodes du début, la régression ridge et le Lasso doivent être calibré pour déterminer le meilleur modèle.
- ▶ C'est-à-dire qu'il faut une méthode qui choisisse une valeur du paramètre de réglage  $\lambda$ , ou de la contrainte  $s$ .
- ▶ La *validation croisée* fournit une façon simple d'attaquer ce problème. On fixe une grille de valeurs de  $\lambda$  possible et sur cette grille, on estime l'erreur de test par validation croisée.
- ▶ On choisit alors la valeurs de  $\lambda$  pour laquelle cette estimation de l'erreur de test est la plus faible.
- ▶ Enfin, le modèle est ré-ajusté pour utiliser toutes les observations de la base d'entraînement avec la valeur de  $\lambda$  précédemment obtenue.

# Le zoo des méthodes lasso I

- The elasticnet *Zou et Hastie 2005* vise à activer les variables corrélées simultanément

$$\hat{\boldsymbol{\beta}}^{\text{e-net}} = \arg \min_{\boldsymbol{\beta} \in \mathbb{R}^p} \left\{ \frac{1}{2} \text{RSS}(\boldsymbol{\beta}) + \lambda (\alpha \|\boldsymbol{\beta}\|_1 + (1 - \alpha) \|\boldsymbol{\beta}\|_2) \right\}$$

- Adaptive/Weighted-Lasso pondère chaque composante du vecteur de coefficients

$$\hat{\boldsymbol{\beta}}^{\text{lasso}} = \arg \min_{\boldsymbol{\beta} \in \mathbb{R}^p} \left\{ \frac{1}{2} \text{RSS}(\boldsymbol{\beta}) + \lambda \|\mathbf{w} \circ \boldsymbol{\beta}\|_1 \right\}.$$

# Le zoo des méthodes lasso II

- ▶ **Group-Lasso** *Yuan and Lin 2006* vise à activer les variables par groupes

$$\hat{\boldsymbol{\beta}}^{\text{group}} = \arg \min_{\boldsymbol{\beta} \in \mathbb{R}^p} \left\{ \frac{1}{2} \text{RSS}(\boldsymbol{\beta}) + \lambda \sum_{k=1}^K w_k \|\boldsymbol{\beta}_{\mathcal{G}_k}\|_1 \right\}$$

- ▶ **Cooperative-Lasso** *Chiquet et al. 2010* vise à activer les variables par groupes de même signe

$$\hat{\boldsymbol{\beta}}^{\text{coop}} = \arg \min_{\boldsymbol{\beta} \in \mathbb{R}^p} \left\{ \frac{1}{2} \text{RSS}(\boldsymbol{\beta}) + \lambda \sum_{k=1}^K w_k \left( \|\boldsymbol{\beta}_{\mathcal{G}_k}^+\|_1 + \|\boldsymbol{\beta}_{\mathcal{G}_k}^-\|_1 \right) \right\}$$

# Bilan

- ▶ Les méthodes de sélection de modèles sont essentielles pour l'analyse de données, et l'apprentissage statistique, en particulier avec de gros jeu de données contenant de nombreux prédicteurs.
- ▶ Les questions de recherches qui donnent des solutions creuses (parcimonieuses, ou sparses), comme le Lasso, sont d'actualité.

# Classification

# Régression logistique

Notons  $p(X) = \mathbb{P}(Y = 1|X)$  et considérons un seul prédicteur  $X$ . La régression logistique pose

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

qui est toujours entre 0 et 1! On a alors

$$\log\left(\frac{p(X)}{1 - p(X)}\right) = \beta_0 + \beta_1 X$$

(transformation logit)

**Estimation par maximum de vraisemblance** La vraisemblance

$$L(\beta_0, \beta_1) = \prod_{i:y_i=1} p(x_i) \prod_{i:y_i=0} [1-p(x_i)]$$

que l'on maximise pour obtenir  $\beta_0$  et  $\beta_1$  (Ordinateur)

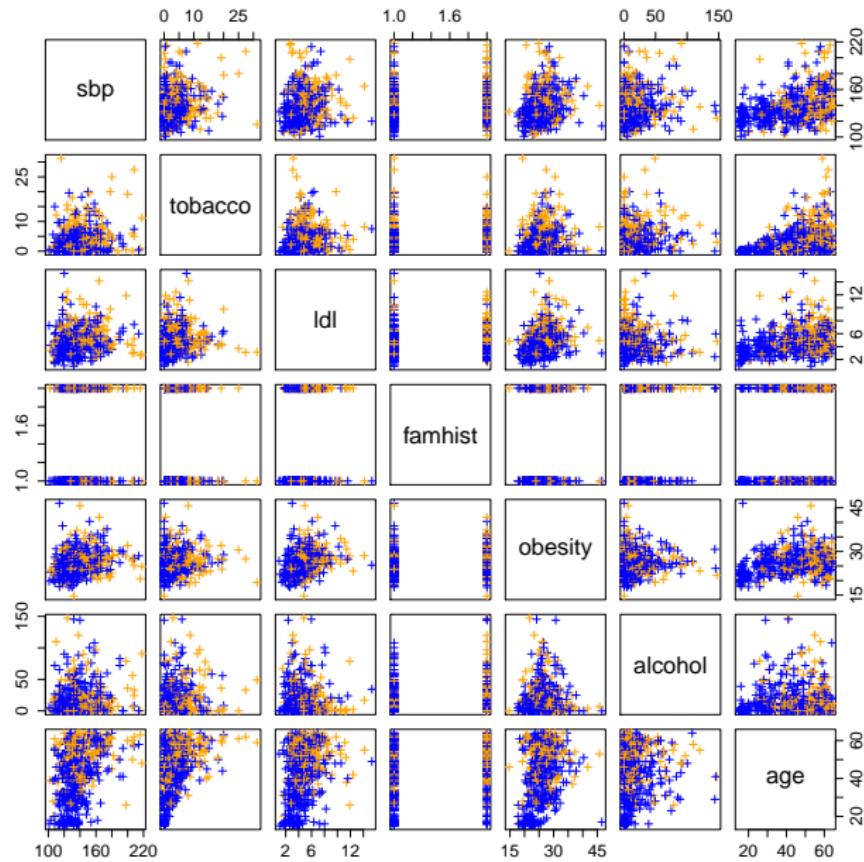
La plupart des logiciels de statistique le font (`glm` de R par exemple)

# Régression logistique à plusieurs co-variables

$$\log \left( \frac{p(X)}{1-p(X)} \right) = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p$$

## Exemple : maladie cardiaque en Afrique du Sud

- ▶ 160 cas d'infarctus du myocarde (MI) et 302 cas de contrôle (homme entre 15-64 ans), de la province de Cap-Occidental en Afrique du Sud, au début des années 80
- ▶ Prévalence très élevée dans cette région : 5.1 %
- ▶ Mesure de 7 prédicteurs (facteurs de risque), montrés dans la page suivante
- ▶ Le but est d'identifier l'influence et la force relative des facteurs de risque
- ▶ Cette étude fait partie d'un programme de santé publique dont le but était de sensibiliser la population sur une régime plus équilibré



orange : MI  
bleu :  
contrôle  
**famhist** : 1 si  
antécédents  
familiaux

## Exemple (suite)

```
> heartfit <- glm(chd ~ ., data=heart, family=binomial)
> summary(heartfit)
```

Call:

```
glm(formula = chd ~ ., family = binomial, data = heart)
```

Coefficients:

	Estimate	Std. Error	z value	Pr(> z )	
(Intercept)	-4.1295997	0.9641558	-4.283	1.84e-05	***
sbp	0.0057607	0.0056326	1.023	0.30643	
tobacco	0.0795256	0.0262150	3.034	0.00242	**
ldl	0.1847793	0.0574115	3.219	0.00129	**
famhistPresent	0.9391855	0.2248691	4.177	2.96e-05	***
obesity	-0.0345434	0.0291053	-1.187	0.23529	
alcohol	0.0006065	0.0044550	0.136	0.89171	
age	0.0425412	0.0101749	4.181	2.90e-05	***
---					

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 596.11 on 461 degrees of freedom

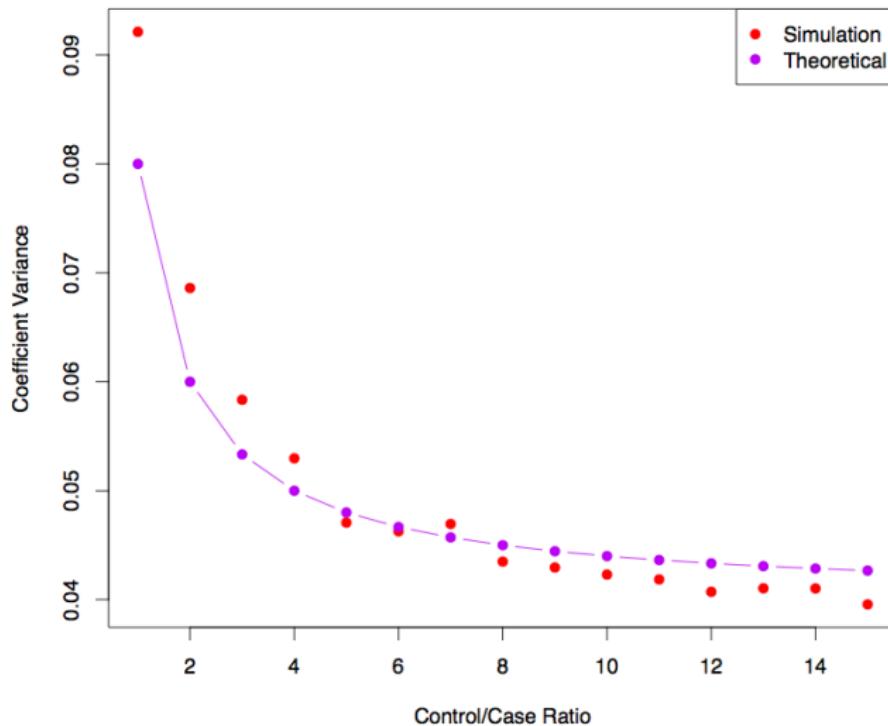
## Échantillonnage du contrôle et régression logistique

- ▶ Dans les données d'Afrique du Sud, il y a 160 MI et 302 contrôle —  $\tilde{\pi} = 0.35$  des cas. Cependant, la prévalence des MI dans la région est de  $\pi = 0.05$ .
- ▶ Ce biais d'échantillonnage permet d'estimer les  $\beta_j$ ,  $j \neq 0$ , avec plus de précision (si modèle correct). Mais l'estimation de  $\beta_0$  doit être corrigée.
- ▶ Une simple transformation permet de le faire :

$$\hat{\beta}_0^* = \hat{\beta}_0 + \log\left(\frac{\pi}{1-\pi}\right) - \log\left(\frac{\tilde{\pi}}{1-\tilde{\pi}}\right)$$

- ▶ Souvent, les cas pathologiques sont rares et on les prend tous. On peut sur-échantillonner jusqu'à 5 fois plus que les cas témoins. Au delà, peu de gain dans la variance d'erreur d'échantillonnage.

# Gain de variance par biais d'échantillonnage de données binaires



Au delà d'un facteur 5 de sur-représentation des cas pathologiques, le gain n'est plus intéressant.

## Régression logistique à plus de deux modalités

Jusqu'à maintenant, nous avons discuté de régression logistique pour expliquer un  $Y$  à deux modalités. Il est facile de généraliser à plus de deux classes. Une possibilité (utilisée dans la bibliothèque `glmnet` de R) est la forme symétrique

$$\mathbb{P}(Y = k|X) = \frac{\exp(\beta_{0k} + \beta_{1k}X_1 + \cdots + \beta_{pk}X_p)}{\sum_{\ell=1}^K \exp(\beta_{0\ell} + \beta_{1\ell}X_1 + \cdots + \beta_{p\ell}X_p)}$$

Il y a donc une fonction linéaire par classe ou modalité. En fait, ce modèle est sur-paramétrisé, et comme dans le cas de 2 classes, on peut supprimer l'une des fonctions linéaires et seules ( $K - 1$ ) sont utiles. *Le vérifier!*

La régression logistique multi-classe porte plusieurs noms. On parle parfois de régression multinomiale.

# Sélection de modèles

Revenons à l'expression

$$\underset{(\beta_0, \boldsymbol{\beta}) \in \mathbb{R}^{p+1}}{\text{minimiser}} \left\{ -\frac{1}{N} \ell(\beta_0, \boldsymbol{\beta}) + \lambda P_\alpha(\boldsymbol{\beta}) \right\}$$

- ▶  $\ell(\beta_0, \boldsymbol{\beta}) = \frac{1}{2\sigma^2} \|y - \beta_0 \mathbf{1} - X\boldsymbol{\beta}\|_2^2 + c$  pour une régression linéaire multiple.
- ▶ Dans le cas d'une régression logistique

$$\ell(\beta_0, \boldsymbol{\beta}) = \sum_{i=1}^N \left\{ y_i (\beta_0 + \boldsymbol{\beta}' x_i) - \log(1 + e^{\beta_0 + \boldsymbol{\beta}' x_i}) \right\}$$

- ▶ La terme de régularisation (de pénalité)

$$P_\alpha(\boldsymbol{\beta}) = \alpha \|\boldsymbol{\beta}\|_1 + (1 - \alpha) \|\boldsymbol{\beta}\|_2$$

Lasso si ( $\alpha = 1$ ) ridge si  $\alpha = 1$ .

# Deux exemples en grande dimension

Allons faire des testes sous R

- ▶ Jeu de données leukemia du package **spikeslab** : les gênes exprimés différemmentiellement exprimés pour les deux types de leucémie *Acute Myeloblastic Leukemia* et *Acute Lymphoblastic Leukemia*.  $n = 72$  et  $p = 3571$ .
- ▶ Jeu de données nki du package **BreastCancerNKI** de bioconductor : déterminer les gênes exprimés sous la condition *estrogen receptor status* actif.  $n = 337$  et  $p = 24481$

# Analyse discriminante

L'idée est modéliser la loi de  $X$  dans chaque classe séparément, et d'utiliser le *théorème de Bayes* pour obtenir  $\mathbb{P}(Y = k|X)$ .

Lorsque l'on utilise des lois gaussiennes pour chaque classes, cela donne l'analyse discriminante linéaire ou quadratique.

Cette approche est plus générale, et l'on pourrait utiliser d'autres distributions. Concentrons nous d'abord sur le cas gaussien.

Notons

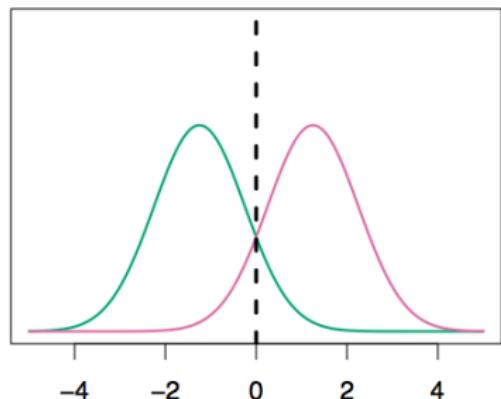
- ▶  $f_k(x)$  la *densité* de  $X$  sachant que l'on est dans la classe  $k$  ;
- ▶  $\pi_k$  la probabilité (marginale) de la classe  $k$

On a alors

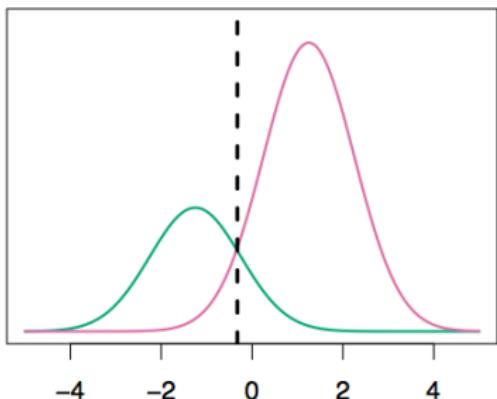
$$\mathbb{P}(Y = k|X = x) = \frac{\pi_k f_k(x)}{\sum_{\ell=1}^K \pi_\ell f_\ell(x)}.$$

# Classifier dans la classe de densité la plus élevée

$$\pi_1=.5, \quad \pi_2=.5$$



$$\pi_1=.3, \quad \pi_2=.7$$



On classifie une nouvelle observation dans la classe de densité la plus élevée.

Lorsque les fréquences (marginales) des classes sont différentes, il faut prendre en compte cette différence, et l'on compare les  $\pi_k f_k(x)$ .

Sur l'exemple ci-dessus, la classe des roses a une probabilité marginale plus élevée. La frontière entre les deux décisions s'est déplacée sur la gauche.

## Pourquoi utiliser l'analyse discriminante linéaire ?

- ▶ Lorsque les classes sont bien séparées, l'estimation des paramètres de la régression logistique devient instable. L'analyse discriminante linéaire (LDA) ne souffre pas de ce problème.
- ▶ Lorsque  $n$  est petit et la distribution de  $X$  est à peu près gaussienne dans chaque classe, LDA est plus stable que la régression logistique.
- ▶ LDA est aussi populaire lorsqu'il y a plus de deux modalités pour  $Y$  car elle permet de projeter les données dans des plans séparent les groupes.

## En dimension 1

On cherche

$$p_k(x) = \mathbb{P}(Y = k | X = x) \text{ avec}$$

$$f_k(x) = \frac{1}{\sqrt{2\pi}\sigma_k} \exp\left(-\frac{1}{2}\left(\frac{x - \mu_k}{\sigma_k}\right)^2\right)$$

LDA suppose que

$$\sigma_1 = \dots = \sigma_K = \sigma.$$

Ce qui donne

$$p_k(x) = \frac{\pi_k e^{-\frac{1}{2}\left(\frac{x - \mu_k}{\sigma}\right)^2}}{\sum_{\ell=1}^K \pi_\ell e^{-\frac{1}{2}\left(\frac{x - \mu_\ell}{\sigma}\right)^2}}$$

après avoir simplifier par  $1/\sqrt{2\pi}\sigma$  en facteur partout.

## En dimension 1

On cherche

$$p_k(x) = \mathbb{P}(Y = k | X = x) \text{ avec}$$

$$f_k(x) = \frac{1}{\sqrt{2\pi}\sigma_k} \exp\left(-\frac{1}{2}\left(\frac{x - \mu_k}{\sigma_k}\right)^2\right)$$

LDA suppose que

$$\sigma_1 = \dots = \sigma_K = \sigma.$$

Ce qui donne

$$p_k(x) = \frac{\pi_k e^{-\frac{1}{2}\left(\frac{x - \mu_k}{\sigma}\right)^2}}{\sum_{\ell=1}^K \pi_\ell e^{-\frac{1}{2}\left(\frac{x - \mu_\ell}{\sigma}\right)^2}}$$

après avoir simplifier par  
 $1/\sqrt{2\pi}\sigma$  en facteur partout.

Pour construire un classificateur en  $X = x$ , il faut maintenant chercher quel  $p_k(x)$  est le plus grand.

Noter que les dénominateurs de dépendent pas de  $k$ .

En passant au log, il suffit de comparer les

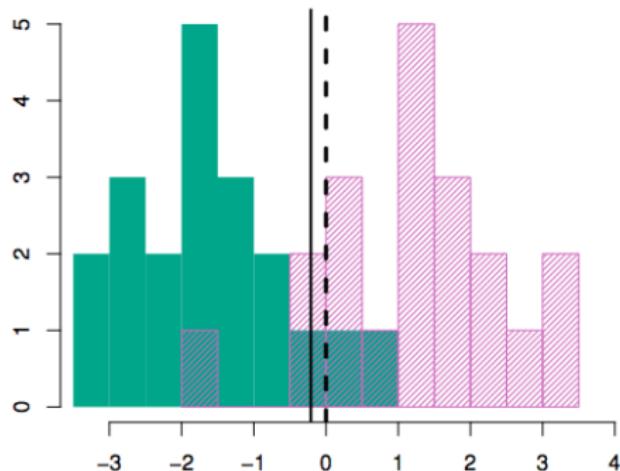
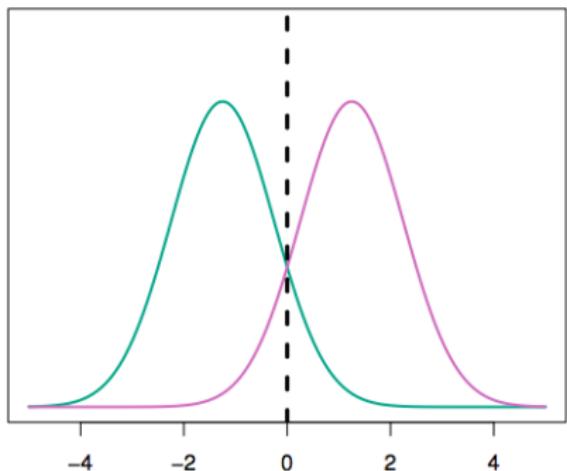
$$\delta_k(x) = x \cdot \frac{\mu_k}{\sigma^2} - \frac{\mu_k^2}{2\sigma^2} + \log(\pi_k).$$

Ces fonctions **linéaires** de  $x$  s'appellent les scores de Fisher.

Si  $K = 2$ , et  $\pi_1 = \pi_2 = 0.5$ , vérifier que la frontière de décision est en

$$x = \frac{\mu_1 + \mu_2}{2}.$$

# Analyse discriminante : estimation



Exemple simulé avec  $\mu_1 = -1.5$ ,  $\mu_2 = 1.5$ ,  $\pi_1 = \pi_2 = 0.5$ , et  $\sigma^2 = 1$ .

Typiquement, on ne connaît pas ces paramètres et on doit les apprendre sur des données. Dans ce cas, on estime simplement les paramètres et on << *plug* >> les estimations dans les formules théoriques.

# Analyse discriminante linéaire (suite)

$$\widehat{\pi_k} = n_k/n$$

$$\widehat{\mu_k} = \sum_{i:y_i=k} x_i / n_k$$

$$\widehat{\sigma_k^2} = \sum_{i:y_i=k} \left( x_i - \widehat{\mu_k} \right)^2 / (n_k - 1)$$

$$\widehat{\sigma^2} = \sum_{k=1}^K \frac{n_k - 1}{n - K} \cdot \widehat{\sigma_k^2}$$

En dimension  $p > 1$ , les formules se généralisent.

Rappelons

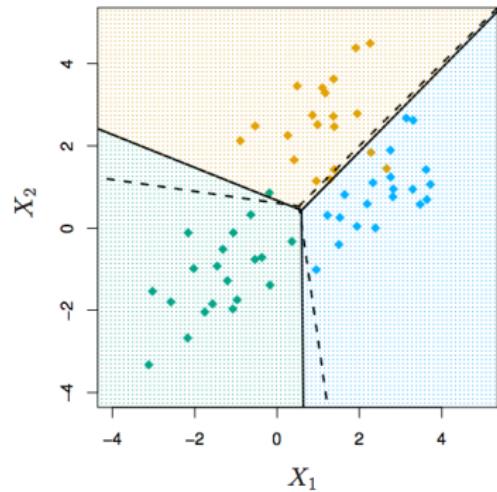
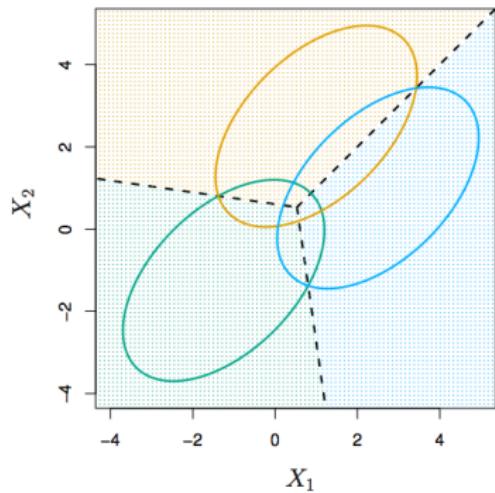
$$f(x) = \frac{1}{(2\pi)^{p/2} |\Sigma|^{1/2}} e^{\frac{1}{2}(x-\mu)^T \Sigma^{-1} (x-\mu)}$$

Les scores deviennent

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

qui est toujours une fonction linéaire en  $x$ .

## Exemple simulé en dimension 2

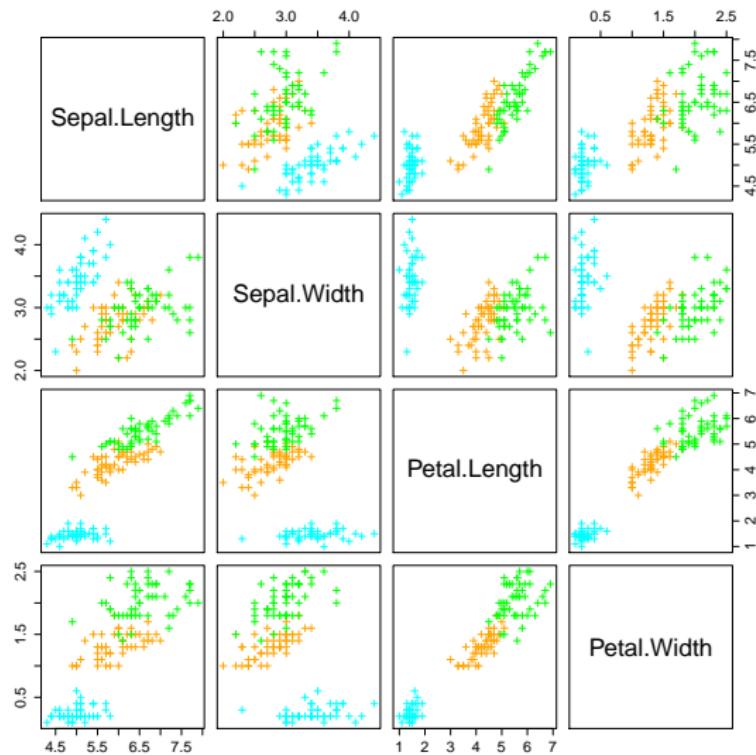


Exemple simulé  $\pi_1 = \pi_2 = \pi_3 = 1/3$ .

Les lignes pointillées : frontières théoriques

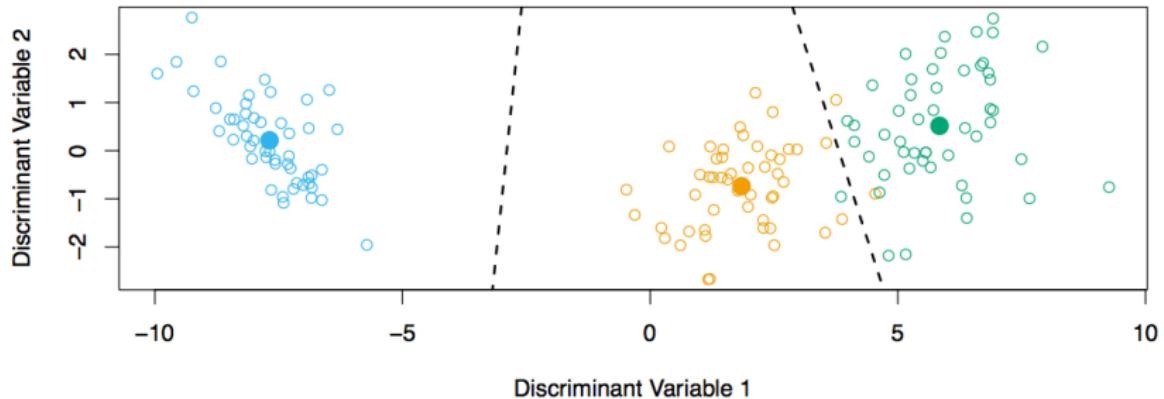
Les lignes pleines : frontières estimées par LDA

# Autre exemple : les Iris de Fisher



4 variables explicatives  
3 espèces  
50 observations/classes  
bleu : « setosa »  
orange : « versicolor »  
vert : « virginica »

# Plan discriminant des Iris



Lorsqu'il y a  $K$  groupes, LDA fournit une projection de dimension  $(K - 1)$  qui sépare au mieux les nuages de points.

$(K - 1)$ ? Essentiellement parce les centres des classes  $\mu_1$ ,  $\mu_2$  et  $\mu_3$  (ou plutôt leurs estimations) définissent un plan dans l'espace.

## Autres formes d'analyse discriminante

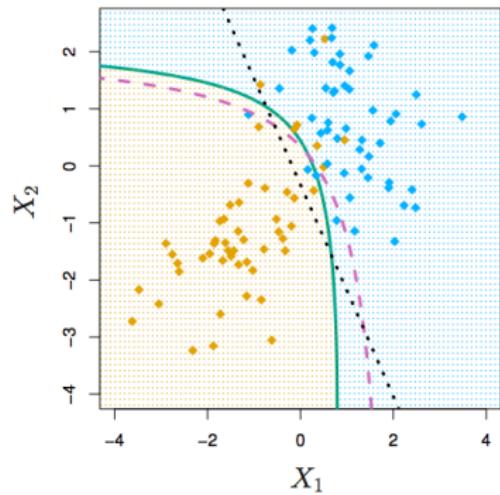
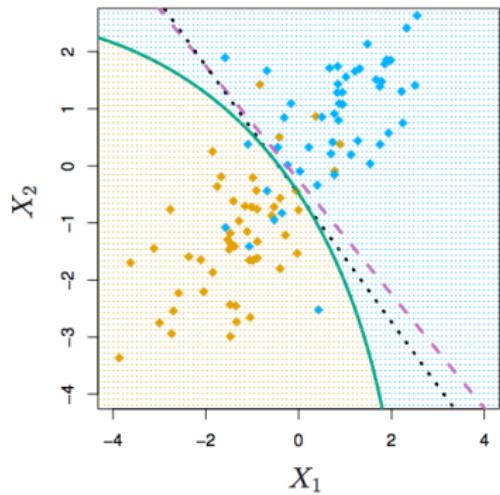
$$\mathbb{P}(Y = k | X = x) = \frac{\pi_k f_k(x)}{\sum_{\ell=1}^K \pi_\ell f_\ell(x)}$$

Lorsque les  $f_k(x)$  sont des gaussiennes de même variance, LDA.

En changeant la forme des  $f_k(x)$ , on obtient d'autres classificateurs.

- ▶ Avec des gaussiennes, mais de variances distinctes, on obtient l'*analyse discriminante quadratique*
- ▶ Avec des  $f_k(x) = \prod_{j=1}^p f_{jk}(x_j)$  qui se factorisent par co-variables, on obtient *naive Bayes*
- ▶ Beaucoup d'autres cas, y compris non-paramétriques.

# Analyse discriminante quadratique



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

Comme les  $\Sigma_k$  sont différents, des termes quadratiques apparaissent.

## Naive Bayes

Suppose que les co-variables sont indépendantes conditionnellement à chaque classe.

Utile quand  $p$  grand et que des méthodes comme QDA ou LDA tombent.

- ▶ Le cas gaussien revient à supposer que les  $\Sigma_k$  sont diagonales

$$\delta_k(x) \propto \log \left( \pi_k \prod_{j=1}^p f_{kj}(x_j) \right) = -\frac{1}{2} \sum_{j=1}^p \frac{(x_j - \mu_{jk})^2}{\sigma_{kj}^2} + \log(\pi_k)$$

- ▶ peut servir dans le cas mixte où certaines co-variables sont qualitatives. Dans ce cas, on replace les densités intra-classes correspondantes par des fréquences intra-classes.

Même si l'hypothèse de départ semble très forte, mérite d'être testé car donne souvent de bons résultats.

# Régression logistique ou LDA ?

Pour un problème à deux classes, LDA vérifie

$$\log \left( \frac{p_1(x)}{1 - p_1(x)} \right) = \log \left( \frac{p_1(x)}{p_2(x)} \right) = c_0 + c_1 x_1 + \cdots + c_P x_p$$

Même forme que pour la régression logistique.

La différence est dans la façon dont sont estimés les paramètres.

- ▶ La régression logistique utilise des vraisemblances conditionnelles et ne modélise que  $\mathbb{P}(Y = k|X)$  — on parle d'*apprentissage discriminant*
- ▶ LDA utilise une vraisemblance complète basée sur la loi jointe de  $(X, Y)$  — on parle d'*apprentissage génératif*
- ▶ Malgré ces différences, en pratique, les résultats sont souvent similaires.

## Conclusion et... .

- ▶ La régression logistique est très populaire pour la classification, surtout lorsqu'il n'y a que  $K = 2$  classes
- ▶ LDA est utile, même lorsque  $n$  est petit, ou lorsque les classes sont bien séparées et que l'hypothèse gaussienne est raisonnable. Et  $K > 2$
- ▶ *Naive Bayes* est utile lorsque  $p$  est grand
- ▶ En petite dimension ( $p < 4$ ), et avec beaucoup d'observations, on peut faire de l'analyse discriminante non-paramétrique, en remplaçant les gaussiennes par des densités intra-classes estimées (par une méthode à noyau)
- ▶ Autres méthodes (modèles additifs généralisés, SVM, random forest, etc.) dans les cours suivants.

# Dendrologie

# Méthodes basées sur des arbres

- ▶ Nous décrivons ici des méthodes *basées sur des arbres* pour la classification et la régression.
- ▶ Cela implique de *stratifier* ou *segmenter* l'espace des prédicteurs en un certain nombre de régions simples
- ▶ Comme l'ensemble des règles des partitionnement peuvent être résumées par un arbre, ce type d'approches sont connues comme des méthodes à *arbres de décision*

# Pours et contres

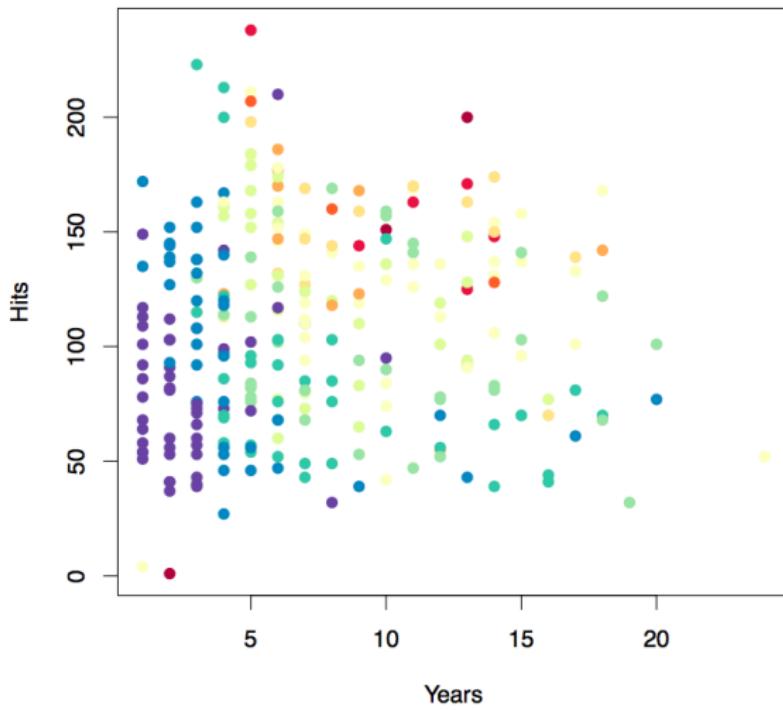
- ▶ Les méthodes basées sur des arbres sont simples et utiles pour l'interprétation.
- ▶ Cependant, elles ne sont pas capables de rivaliser avec les meilleures approches d'apprentissage supervisé en terme de qualité de prédiction
- ▶ Nous discuterons donc aussi de *bagging*, *forêts aléatoires (random forests)*, et *boosting*. Ces méthodes développent de nombreux arbres de décision qui sont ensuite *combinés* pour produire une réponse consensus.

# Les bases des arbres de décision

- ▶ Les arbres de décision sont utiles aussi bien pour des problèmes de régression que de classification.
- ▶ Nous commençons par présenter des problèmes de régression et nous viendrons ensuite à la classification.

# Données de salaire au baseball : comment les stratifier ?

Le salaire est codé par des couleurs : les faibles valeurs sont en bleu, puis vert, les plus fortes valeurs en orange puis rouge.



## L'arbre de décision sur ces données



## Détails sur la précédente figure

- ▶ C'est un arbre de régression pour prédire le **log** des salaires des joueurs, basé sur
  - ▶ l'expérience (**Years**)
  - ▶ le nombre de succès (**Hits**)
- ▶ Pour chaque nœud interne, l'étiquette (de la forme  $X_j < t_k$ ) indique la branche de gauche émanant du nœud, et la branche droite correspond à  $X_j \geq t_k$ .
- ▶ Cet arbre a deux nœuds internes et trois nœuds terminaux ou feuilles. Le nœud le plus haut dans la hiérarchie est la racine.
- ▶ L'étiquette des feuilles est la réponse moyenne des observations qui satisfont aux critères pour la rejoindre.

## Résultats

- ▶ En tout, l'arbre distingue trois classes de joueurs en partitionnant l'espace des prédicteurs en trois régions :  
 $R_1 = \{X : \text{Years} < 4.5\}$ ,  $R_2 = \{X : \text{Years} \geq 4.5, \text{Hits} < 117.5\}$   
et  $R_3 = \{X : \text{Years} \geq 4.5, \text{Hits} \geq 117.5\}$ .

# Interprétation des résultats

- ▶ **Years** est le facteur le plus important pour expliquer **Salary** : les joueurs de moindre expérience gagnent moins que les joueurs expérimentés
- ▶ Sachant qu'un joueur a peu d'expérience, le nombre de **Hits** l'année passée n'influence pas son salaire
- ▶ Mais, parmi les joueurs expérimentés, le nombre de **Hits** de l'année passée affecte son salaire (positivement)
- ▶ C'est sûrement une simplification de la réalité, mais comparé à un modèle de régression (linéaire par exemple), la fonction de régression est simple à décrire, interpréter et expliquer.

# Détails sur la construction de l'arbre

## Algorithme CART (Classification and Regression Trees)

1. Division de l'espace des prédicteurs en  $J$  régions distinctes, non recouvrantes :  $R_1, R_2, \dots, R_J$ .
2. Pour toute nouvelle observation des prédicteurs  $X = x_0$ , on regarde dans quelle région on tombe, disons  $R_\ell$ . La prédiction est la moyenne des valeurs observées dans la partie de l'ensemble d'entraînement qui tombent dans  $R_\ell$ .

## Détails sur la construction de l'arbre (suite)

- ▶ Pour limiter l'espace des partitions possibles, les arbres de décision divisent l'espace en rectangles ou boîtes parallèles aux axes.
- ▶ Le but est de trouver les boîtes  $R_1, \dots, R_J$  qui minimisent un critère des moindres carrés, ici

$$SSE = \sum_{j=1}^J \sum_{i:x_i \in R_j} \left( y_j - \hat{y}_{R_j} \right)^2,$$

où  $\hat{y}_{R_j}$  est la réponse moyenne sur les observations d'entraînement qui tombent dans  $R_j$ .

## Détails sur la construction de l'arbre (suite)

- ▶ Malheureusement, il est impossible de traverser l'ensemble des partitionnements de l'espace des prédicteurs en  $J$  boîtes.
- ▶ Pour cette raison, on met en place un algorithme *glouton*, *top-down* qui construit l'arbre binaire de façon récursive.
- ▶ L'algorithme démarre à la racine de l'arbre et sépare ensuite l'espace des prédicteurs en ajoutant progressivement des nœuds.
- ▶ On parle d'algorithme *glouton* car à chaque étape de la construction de l'arbre, on construit la meilleure division possible du nœud en deux sous-nœuds.

# L'algorithme de construction de l'arbre $\mathcal{T}_0$ (phase 1)

## Initialisation

Nœud racine : on place l'ensemble de l'échantillon d'estimation à la racine de l'arbre

## Récurrence sur chaque nœud

On partitionne chaque nœud en deux classes :

$$\mathcal{R}_1(j, s) = \{X : X_j \leq s\}, \quad \mathcal{R}_2(j, s) = \{X : X_j > s\}$$

en cherchant  $j$  et  $s$  qui minimisent

$$\sum_{x_i \in \mathcal{R}_1(j, s)} \left( y_i - \hat{c}_1 \right)^2 + \sum_{x_i \in \mathcal{R}_2(j, s)} \left( y_i - \hat{c}_2 \right)^2$$

où  $\hat{c}_m = \text{ave}\left(y_i \mid x_i \in \mathcal{R}_m(j, s)\right)$ .

# Algorithme (suite...)

## Phase 1 : Construction de $\mathcal{T}_0$

Initialisation

[...]

Récurrence sur chaque noeud

[...]

Terminaison

On arrête de diviser un noeud de  $\mathcal{T}_0$  lorsqu'il y a peu d'observations (disons 5).

# Algorithme (suite...)

## Phase 1 : Construction de $\mathcal{T}_0$

Initialisation

[...]

Récurrence sur chaque noeud

[...]

Terminaison

On arrête de diviser un noeud de  $\mathcal{T}_0$  lorsqu'il y a peu d'observations (disons 5).

Problème de sur-apprentissage

- un arbre trop profond sur-apprend
- un arbre trop peu profond n'est pas assez précis

D'où « Élagage de  $\mathcal{T}_0$  »

# Sur-apprentissage

L'arbre  $\mathcal{T}_0$  obtenu est trop profond. Faire un compromis entre

- ▶ sur-apprentissage (trop profond)
- ▶ erreur de prédiction trop grande (trop court)

Introduire un paramètre  $\alpha$  qui règle le compromis, et minimiser le critère pénalisé défini pour  $T \subset \mathcal{T}_0$  par

$$\mathcal{C}_\alpha(T) := \sum_{m=1}^{|T|} N_m(T) Q_m(T) + \alpha |T|,$$

où  $|T|$  = nombre de feuilles de  $T$ ;  $N_m(T) = \text{Card } \{x_i \in \mathcal{R}_m(T)\}$ ;

$$Q_m(T) = \frac{1}{N_m(T)} \sum_{x_i \in \mathcal{R}_m(T)} (y_i - \hat{c}_m(T))^2;$$

$$\hat{c}_m(T) = \text{ave}\left(y_i \middle| x_i \in \mathcal{R}_m(T)\right)$$

# Algorithme (suite...)

## Phase 2 : Élagage

### Calcul des minima $\mathcal{T}_\alpha$ du critère pénalisé

On part de  $\mathcal{T}_0$  et on construit une suite d'arbres itérativement. À chaque étape, on supprime le nœud interne de tel sorte à produire la plus petite augmentation de

$$\sum_m N_m(T) Q_m(T)$$

et l'on s'arrête lorsque l'arbre est réduit à un seul nœud (racine). Tous les minima  $T = \mathcal{T}_\alpha$  des fonctions  $T \mapsto \mathcal{C}_\alpha(T)$  sont dans cette suite

# Algorithme (suite...)

## Phase 2 : Élagage

### Calcul des minima $\mathcal{T}_\alpha$ du critère pénalisé

On part de  $\mathcal{T}_0$  et on construit une suite d'arbres itérativement. À chaque étape, on supprime le nœud interne de tel sorte à produire la plus petite augmentation de

$$\sum_m N_m(T) Q_m(T)$$

et l'on s'arrête lorsque l'arbre est réduit à un seul nœud (racine). Tous les minima  $T = \mathcal{T}_\alpha$  des fonctions  $T \mapsto \mathcal{C}_\alpha(T)$  sont dans cette suite

## Calibration de $\alpha$

Par validation croisée à 5 ou 10 blocs

# Algorithme (suite... et fin !)

## Phase 2 : Élagage

### Calcul des minima $\mathcal{T}_\alpha$ du critère pénalisé

On part de  $\mathcal{T}_0$  et on construit une suite d'arbres itérativement. À chaque étape, on supprime le nœud interne de tel sorte à produire la plus petite augmentation de

$$\sum_m N_m(T) Q_m(T)$$

et l'on s'arrête lorsque l'arbre est réduit à un seul nœud (racine). Tous les minima  $T = \mathcal{T}_\alpha$  des fonctions  $T \mapsto \mathcal{C}_\alpha(T)$  sont dans cette suite

## Calibration de $\alpha$

Par validation croisée à 5 ou 10 blocs

**FIN**

# Arbres de classification

- ▶ Similaires aux arbres de régression, sauf qu'ils sont utilisés pour prédire une réponse discrètes
- ▶ Pour un arbre de classification, on prédit à l'aide la classe la plus fréquente dans cette feuille parmi les données d'entraînement.

## Différence avec la régression

$Y \in \{1, 2, \dots, K\}$ , donc  $\hat{f}(x) \in \{1, 2, \dots, K\}$

Il faut changer le critère des moindres carrés en un critère plus adapté, lié à la mesure des erreurs de classification.

Si la feuille  $m$  représente la région  $\mathcal{R}_m$  avec  $N_m$  observations, on définit

$$\hat{p}_{mk} = \frac{1}{N_m} \sum_{x_i \in \mathcal{R}_m} \mathbf{1}\{y_i = k\},$$

la proportion d'observation du noeud  $m$  appartenant à la  $k^{\text{e}}$  classe.

Alors la valeurs de  $\hat{f}$  sur  $\mathcal{R}_m$  est  $\hat{c}_m = \operatorname{argmax}_k \hat{p}_{mk}$  (vote à la majorité simple).

# Mesure d'erreur

Les différentes mesures d'impureté  $Q_m(T)$  d'une feuille  $m$  sont

- ▶ **erreur de classification** :  $\frac{1}{N_m} \sum_{x_i \in \mathcal{R}_m} \mathbf{1}\{y_i \neq \hat{c}_m\} = 1 - \hat{p}_{m\hat{c}_m}$

Le critère que l'on minimise pour construire  $\mathcal{T}_0$  est

$$\sum_{m=1}^{|T|} N_m(T) Q_m(T)$$

On pénalise ensuite comme pour la régression.

# Mesure d'erreur

Les différentes mesures d'impureté  $Q_m(T)$  d'une feuille  $m$  sont

- ▶ **erreur de classification** :  $\frac{1}{N_m} \sum_{x_i \in \mathcal{R}_m} \mathbf{1}\{y_i \neq \hat{c}_m\} = 1 - \hat{p}_{m\hat{c}_m}$
- ▶ **Index de Gini** :  $\sum_{k \neq k'} \hat{p}_{mk} \hat{p}_{mk'} = \sum_k \hat{p}_{mk} (1 - \hat{p}_{mk})$

Le critère que l'on minimise pour construire  $\mathcal{T}_0$  est

$$\sum_{m=1}^{|T|} N_m(T) Q_m(T)$$

On pénalise ensuite comme pour la régression.

# Mesure d'erreur

Les différentes mesures d'impureté  $Q_m(T)$  d'une feuille  $m$  sont

- ▶ **erreur de classification** :  $\frac{1}{N_m} \sum_{x_i \in \mathcal{R}_m} \mathbf{1}\{y_i \neq \hat{c}_m\} = 1 - \hat{p}_{m\hat{c}_m}$
- ▶ **Index de Gini** :  $\sum_{k \neq k'} \hat{p}_{mk} \hat{p}_{mk'} = \sum_k \hat{p}_{mk} (1 - \hat{p}_{mk})$
- ▶ **Entropie** :  $-\sum_k \hat{p}_{mk} \log \hat{p}_{mk}.$

Le critère que l'on minimise pour construire  $\mathcal{T}_0$  est

$$\sum_{m=1}^{|T|} N_m(T) Q_m(T)$$

On pénalise ensuite comme pour la régression.

# Bibliothèque rpart de R

Lire la « vignette » de la bibliothèque

```
vignette("longintro", package="rpart")
```

La commande **rpart** fait presque tout.

# Avantages et inconvénients des arbres

- ▲ Les arbres sont faciles à expliquer à n'importe qui. Ils sont plus faciles à expliquer que les modèles linéaires
- ▲ Les arbres peuvent être représentés graphiquement, et sont interprétables même pour des non-experts
- ▲ Ils peuvent gérer des prédicteurs discrets sans introduire des variables binaires
- ▼ Malheureusement, ils n'ont pas la même qualité prédictive que les autres approches de ce cours.

Cependant, en agrégeant plusieurs arbres de décision, les performances prédictives s'améliorent substantiellement.

# Bagging

- ▶ L'agrégation bootstrap ou bagging est méthode de réduction de la variance en apprentissage statistique. Elle est particulièrement utile sur les arbres de décision.
- ▶ Rappelons que, sur un ensemble de  $n$  observations indépendantes  $Z_1, \dots, Z_n$ , chacune de variance  $\sigma^2$ , la variance de la moyenne  $\bar{Z}$  est  $\sigma^2/n$ .
- ▶ En pratique, il n'est pas possible de moyenner des arbres de décision construits sur de multiples ensembles d'entraînement (pas assez de données observées)

## Bagging (suite)

- ▶ Au lieu de cela, on peut bootstrapper en ré-échantillonnant plusieurs fois les données d'entraînement.
- ▶ Alors, à partir de  $B$  échantillons bootstrap, on entraîne une méthode d'apprentissage pour ajuster  $B$  fonctions de régressions, notées  $\hat{f}^{*b}(x)$ ,  $b = 1, \dots, B$
- ▶ La fonction de régression « bagguée » est alors

$$\hat{f}_{\text{bag}}(x) = \frac{1}{B} \sum_{b=1}^B \hat{f}^{*b}(x)$$

## Bagging (suite)

- ▶ Sur un problème de classification,  $\hat{f}^{*b}(x)$  renvoie une classe possible pour chaque échantillon bootstrap  $b$ .
- ▶ La décision finale  $\hat{f}_{\text{bag}}(x)$  se prend par un vote à la majorité simple parmi les  $B$  prédictions des classifieurs bootstrap.
  
- ▶ Sur des gros jeux de données d'entraînement, faire parfois du sous-échantillonnage bootstrap.

## Erreur *out-of-bag*

- ▶ Il y a une façon simple d'estimer l'erreur de test quand on fait du bagging.
  - ▶ La clé du bagging est l'entraînement de nombreux  $\hat{f}(x)$  sur des échantillons bootstraps. On peut donc utiliser les observations hors du  $b^e$  bootstrap pour évaluer chaque  $\hat{f}^{*b}(x)$
  - ▶ Ce qui donne l'algorithme ci-dessous.
1. Pour chaque observation  $(x_i, y_i)$ , calculer  $\hat{y}_i^{\text{oop}}$  la prédiction en n'utilisant que les estimateurs  $\hat{f}^{*b}(x)$  qui n'ont pas vu cette observation dans leur entraînement
  2. Évaluer l'erreur entre  $\hat{y}_i^{\text{oop}}$  et les  $y_i$  (erreur quadratique moyenne ou taux de mauvaise classification)

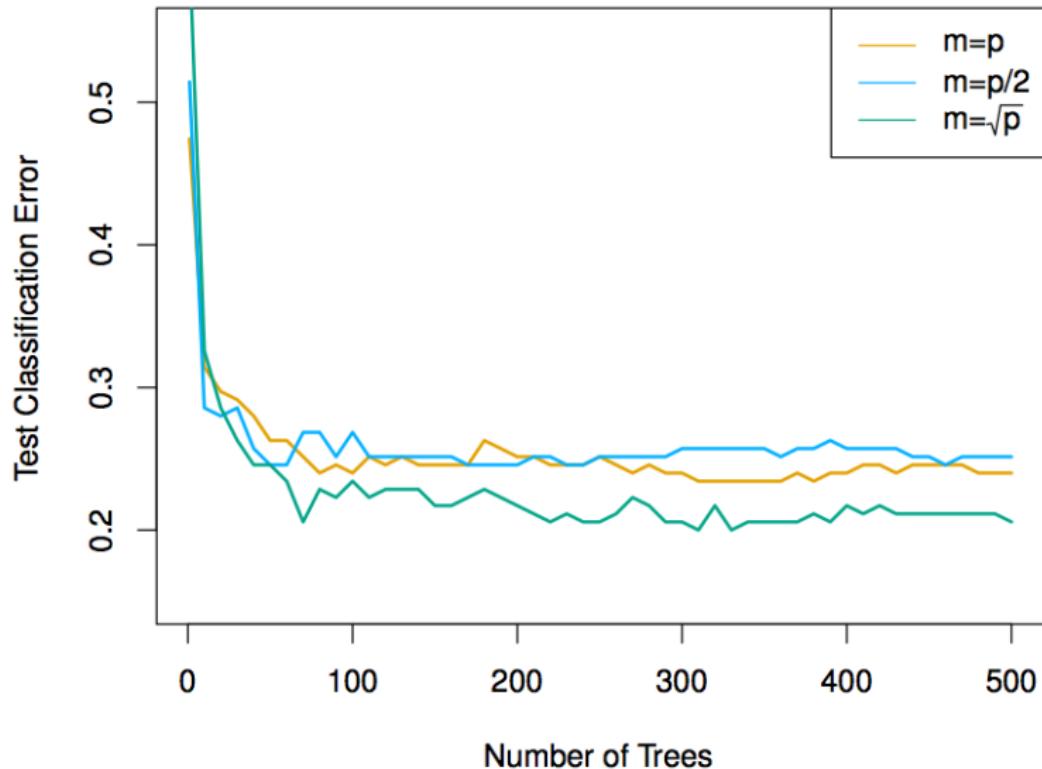
# Forêts aléatoires (random forests)

- ▶ Les *forêts aléatoires* améliorent le bagging des arbres CART en dé-corrélatant ces arbres. Ce qui permet de mieux réduire la variance
- ▶ Comme précédemment, on construit une suite d'arbres de décisions sur des échantillons bootstraps
- ▶ Au lieu d'utiliser l'algorithme CART, on utilise un algorithme randomisé.
- ▶ Au moment de diviser un nœud, cet algorithme ne regarde que  $m$  prédicteurs parmi les  $p > m$  prédicteurs, tirés au hasard.
- ▶ Une nouvelle sélection de prédicteurs est faite à chaque division de nœud.
- ▶ Il est inutile d'élaguer les arbres de décision, le bagging se charge d'éviter les erreurs de sur-apprentissage.

# Paramètres de réglage

- ▶ Il faut choisir
  - ▶ le nombre de prédicteurs  $m$  (par défaut  $\sqrt{p}$  en régression) tirés à chaque division de noeud
  - ▶ le nombre total d'arbres
  - ▶ la taille des sous-échantillons bootstraps si gros jeu de données
- ▶ On peut s'appuyer sur l'erreur *out-of-bag*

## Exemple sur données d'expression de 500 gènes



- ▶ Résultats de forêts aléatoires pour prédire les 15 classes à partir du niveau d'expression de 500 gènes
- ▶ L'erreur de test (évaluée par OOB) dépend du nombre d'arbres. Les différentes couleurs correspondent à différentes valeurs de  $m$ .
- ▶ Les forêts aléatoires améliorent significativement le taux d'erreur de CART (environ 45.7%)

# Boosting

- ▶ Comme le bagging, boosting est une approche générale qui peut s'appliquer à de nombreuses méthodes de régression et classification. Nous présenterons ici uniquement le boosting sur des arbres de décision.
- ▶ Rappelons que le bagging crée de multiples copies du jeu de données par bootstrap, ajuste des arbres de décision sur chacune des copies, et les combinent pour créer un seul modèle de prédiction.
- ▶ Ainsi, chaque arbre est construit sur un échantillon bootstrap indépendamment des autres arbres
- ▶ Le boosting fonctionne un peu de la même façon, sauf que les arbres sont construits *itérativement* : chaque nouvel arbre utilise l'information des arbres précédemment entraînés.

# Algorithme de boosting pour les arbres de régression

1. Fixer  $\hat{f}(x) = 0$  et  $r_i = y_i$  pour tout  $i$  de l'ensemble d'entraînement.
2. Pour  $b = 1, \dots, B$  faire
  - ▶ Ajuster un arbre  $\hat{f}^b$  à  $d$  nœuds internes ( $d + 1$  feuilles) pour prédire les  $r_i$  avec  $x_i$
  - ▶ Mettre à jour  $\hat{f}$  en ajoutant ce nouvel arbre (à un coefficient  $\lambda$  de réduction près)

$$\hat{f}(x) \leftarrow \hat{f}(x) + \lambda \hat{f}^b(x)$$

- ▶ Mettre à jour les résidus

$$r_i \leftarrow r_i - \lambda \hat{f}^b(x_i)$$

3. Retourner le modèle

$$\hat{f}(x) = \sum_{b=1}^B \lambda \hat{f}^b(x).$$

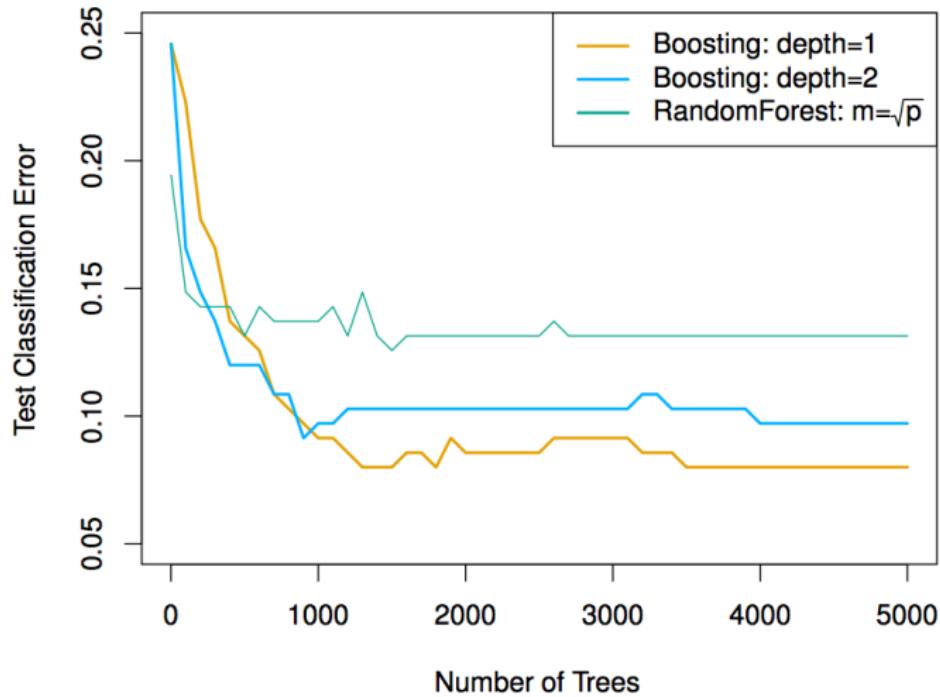
## Quelle est l'idée sous-jacente ?

- ▶ Contrairement à l'ajustement d'un grand arbre de décision aux données (d'entraînement), ce qui revient à s'ajuster *fortement* aux données (avec potentiellement du sur-apprentissage), le boosting *apprend lentement*.
- ▶ Étant donné le modèle courant, on ajuste un arbre de décision aux résidus de ce modèle. Nous ajoutons ce nouvel arbre de décision à la fonction de régression estimée et corrigéons les résidus en conséquence.
- ▶ Chacun de ces arbres est relativement petit, avec peu de noeuds terminaux, déterminés par un paramètre de réglage  $d$  de l'algorithme
- ▶ En ajoutant progressivement ces arbres, on améliore  $\hat{f}$  progressivement, dans les zones où les performances de la version courante ne sont pas bonnes. Le paramètres  $\lambda$  est une sorte de paramètres de « shrinkage » qui règle la vitesse à laquelle on apprend.

# Boosting pour la classification

- ▶ Le boosting pour la classification fonctionne dans le même esprit, mais est un peu plus compliqué. Nous ne donnerons pas de détails ici.
- ▶ Voir, par exemple, le chapitre de 10 de ESL
- ▶ Le package R `gbm` (gradient boosted models) permet d'utiliser le boosting dans de nombreux problèmes de classification et régression.

# Données d'expression de gènes



- ▶ La figure précédente présente les résultats du boosting et de random forest sur le jeu de données d'expression de gènes
- ▶ L'erreur de test est représentée en fonction du nombres d'arbres. Pour les deux modèles ajustés par boosting,  $\lambda = 0.01$ . Les arbres de profondeur 1 battent légèrement ceux de profondeur 2, mais tous deux sont meilleurs que les forêts aléatoires. (L'erreur standard sur ce taux de mauvaise classification estimé est de 0.02, de telle sorte que la différence n'est pas très significative).
- ▶ La meilleure erreur de test avec un seul arbre est de 24%

# Réglage des paramètres du boosting

- 1. Le nombre d'arbres  $B$ .** Contrairement au bagging et aux forêts aléatoires, le boosting peut conduire à un sur-apprentissage si  $B$  est trop grand, même si il faut des grandes valeurs de  $B$  pour le constater.
- 2. Le paramètre de shrinkage  $\lambda$ .** Il contrôle la vitesse à laquelle le boosting apprend. Les valeurs typiques sont 0.01 ou 0.001, mais le meilleur choix dépend du problème. De très petites valeurs de  $\lambda$  peuvent nécessiter l'ajustement de nombreux arbres.
- 3. Le nombre de nœuds internes  $d$ .** Il contrôle la complexité des éléments ajoutés. Souvent,  $d = 1$  est une bonne valeur, auquel cas l'arbre est en forme de queue de cerise.

# Mesure de l'importance des variables

- ▶ Pour les arbres de régression baggués ou pour random forest, on collecte le nombre moyen (moyenne sur les arbres agrégés) de fois qu'un prédicteur intervient dans l'arbre. Une grande valeur
- ▶ Pour les arbres de classification, on ajoute la quantité complète de l'indice de Gini qui a baissé par séparation suivant un certain prédicteur, et on moyenne sur la forêt.

# Résumé

- ▶ Les arbres de décision sont des modèles simples et interprétables.
- ▶ Cependant, ils fournissent souvent de mauvais résultats comparés à d'autres méthodes.
- ▶ Bagging, random forest ou boosting sont de bonne méthodes pour améliorer la qualité de la prédiction des arbres de décision. Ces méthodes agrègent de nombreux arbres entraînés sur les données et ensuite combinent ces arbres pour construire la décision finale.
- ▶ Ces deux dernières méthodes (les forêts aléatoires et le boosting) font parmi de l'état de l'art actuel des méthodes d'apprentissage supervisé. Cependant, le classifieur ou la fonction de régression produite peut être difficile à interpréter.

# Support Vector Machines (SVM)

SVM attaque le problème de classification entre deux classes de façon directe

On essaie de trouver un (hyper)plan qui sépare les classes dans l'espace des prédicteurs

Faute d'y arriver directement,

- ▶ on adoucit la définition de « séparer » et
- ▶ on enrichit l'espace des covariables de sorte à rendre la séparation possible

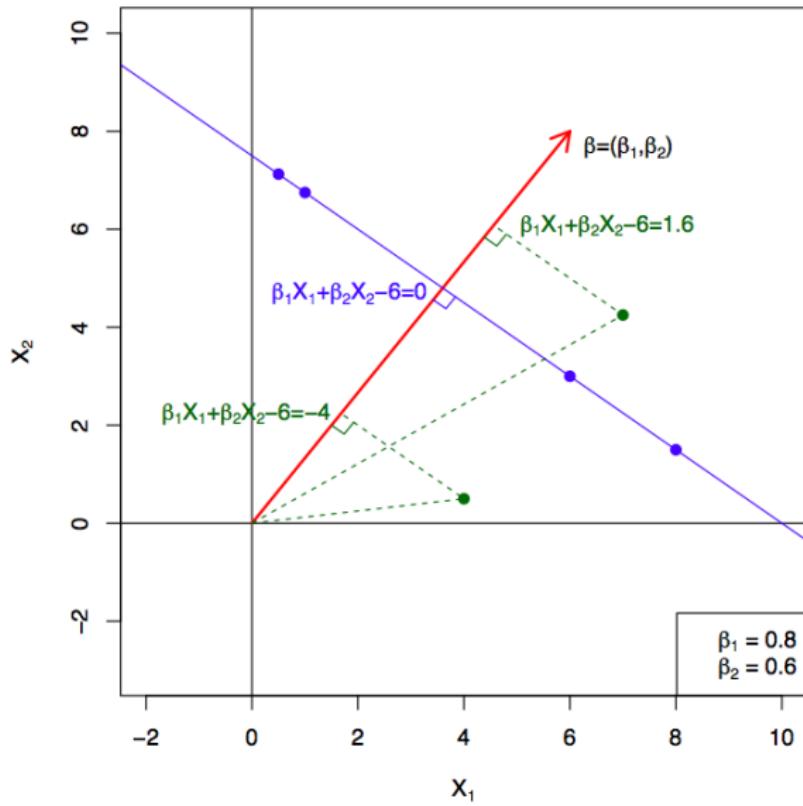
# Qu'est-ce qu'un hyperplan ?

- ▶ Un hyperplan d'un espace de dimension  $p$  est un sous-espace affine de dimension  $p - 1$ .
- ▶ En général, l'équation d'un hyperplan est de la forme

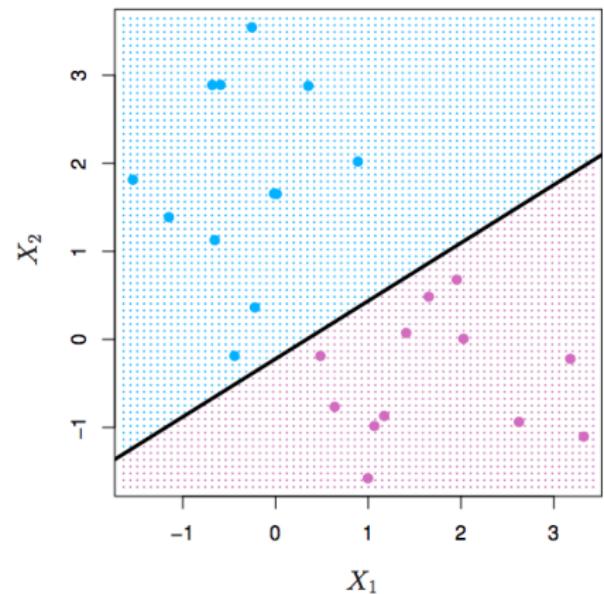
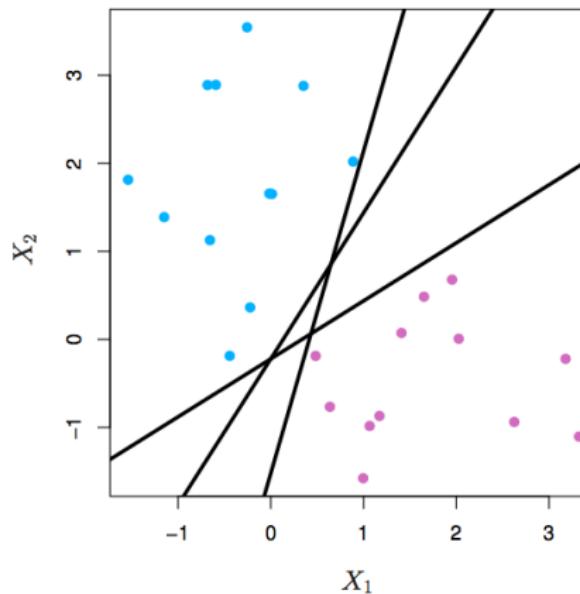
$$\beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p = 0$$

- ▶ En dimension  $p = 2$ , un hyperplan est une droite.
- ▶ Si  $\beta_0$ , il passe par l'origine.
- ▶ Le vecteur  $(\beta_1, \beta_2, \dots, \beta_p)$  est un vecteur normal. Il pointe dans une direction orthogonale à l'hyperplan

## Hyperplan en dimension 2



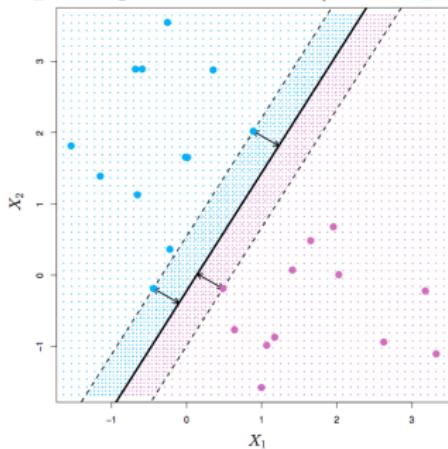
# Hyperplan de séparation



- ▶ Si  $f(X) = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p$ , alors  $f(X) > 0$  pour les points d'un côté de l'hyperplan et  $f(X) < 0$  de l'autre côté.
- ▶ Si l'on suppose que les points  $Y_i = +1$  sont en bleu et  $Y_i = -1$  en mauve, alors  $Y_i \cdot f(X_i) > 0$  pour tout  $i$ . De plus  $f(X) = 0$  définit un *hyperplan séparateur*.

# Classifier de marge maximale

Parmi tout les hyperplans séparateurs, on cherche celui qui crée le plus grand écart (ou la plus grande marge) entre les deux classes



Problème d'optimisation sous contraintes qui maximise  $M$ , fonction de  $\beta_0, \dots, \beta_p$ , sous les contraintes  $\sum_j \beta_j^2 = 1$ , et  $y_i(\beta_0 + \beta_1 x_{i1} + \dots + \beta_p x_{ip}) \geq M$  pour tout  $i$

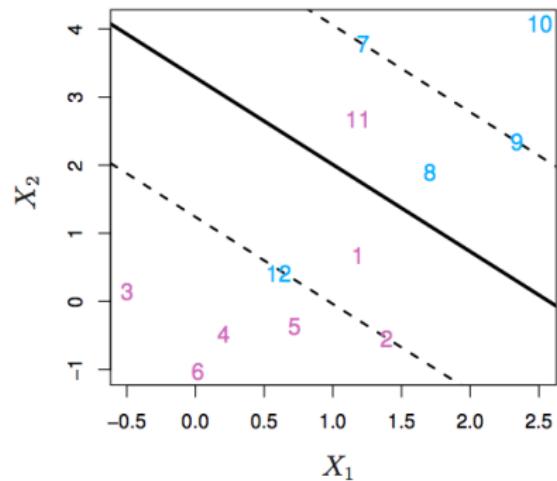
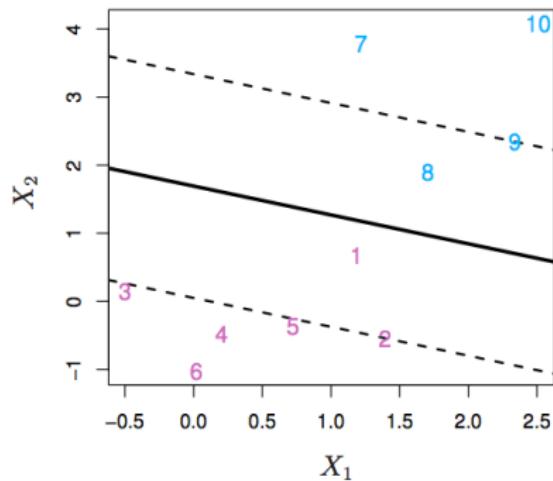
Ce problème peut se voir comme une question de programmation convexe quadratique et résolue efficacement. La fonction `svm()` du package `e1071` résout ce problème efficacement.

# Données non séparables et données bruitées

- ▶ Parfois, les données ne sont pas séparables par un hyperplan.  
C'est souvent le cas, sauf si  $n < p$ .
- ▶ Souvent, les données sont séparables, mais bruitées. Un classifieur construit en maximisant la marge de l'hyperplan séparateur peut avoir une mauvaise erreur de test.

Pour toutes ces raisons, *le classifieur SVM* maximise une marge « douce ».

# SVM classifieur



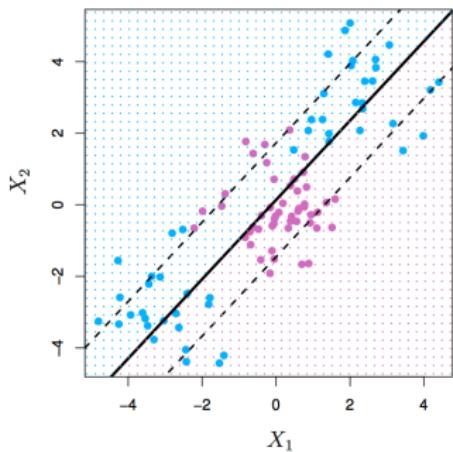
maximiser <sub>$\beta_0, \dots, \beta_p$</sub>   $M$  sous les contraintes  $\sum_j \beta_j^2 = 1,$

$$y_i(\beta_0 + \beta_1 x_{i1} + \dots + \beta_p x_{ip}) \geq M(1 - \varepsilon_i)$$

$$\varepsilon_i \geq 0, \quad \sum_{i=1}^n \varepsilon_i \leq C.$$

Le paramètre  $C$  est un paramètre de régularisation.

# Une frontière linéaire n'est pas toujours une bonne idée



Parfois, il est simplement abscons d'utiliser une frontière linéaire, quelle que soit la valeur de  $C$  que l'on utilise.

# Extension de l'espace des covariables

- ▶ Agrandir les espaces des covariables en ajoutant des transformations non linéaires des  $X_j$  (polynomiales, etc.) On passe alors d'un espace de dimension  $p$  à un espace de dimension  $M$ .
- ▶ Ajuster un classifieur SVM dans l'espace agrandi
- ▶ Cela fournit une frontière de décision non-linéaire dans l'espace original.

**Exemple.** Si on utilise  $(X_1, X_2, X_1^2, X_2^2, X_1X_2)$  au lieu de  $(X_1, X_2)$ , alors la frontière entre les deux décisions est de la forme

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_1^2 + \beta_4 X_2^2 + \beta_5 X_1 X_2 = 0$$

c'est-à-dire une conique.

# Non-linéarités et noyaux

- ▶ Les polynômes (en particulier en grande dimension) devient rapidement incontrôlables.
- ▶ Il y a une façon plus élégante et mesurée d'introduire des non-linéarités dans les classifiers SVM, via l'utilisation de *noyaux*
- ▶ Avant de présenter ce point, nous devons comprendre le rôle du *produit scalaire* dans les SVM.

# Produit scalaire et SVM

$\langle x_i, x_{i'} \rangle = \sum_{j=1}^p x_{ij}x_{i'j}$  est le produit scalaire entre les vecteurs  $x_i$  et  $x_{i'}$

- ▶ Le classifieur construit par SVM peut s'écrire

$$f(x) = \beta_0 + \sum_{i=1}^n \alpha_i \langle x, x_i \rangle$$

avec  $n$  paramètres

- ▶ Pour estimer ces paramètres  $\alpha_i$  et  $\beta_0$ , tout ce dont on a besoin est l'ensemble des  $n(n - 1)/2$  produits scalaires  $\langle x_i, x_{i'} \rangle$  entre paires d'observations de l'ensemble d'entraînement

Il s'avère que la plupart des  $\hat{\alpha}_i$  peuvent être nuls :

$$\hat{f}(x) = \beta_0 + \sum_{i \in \mathcal{S}} \hat{\alpha}_i \langle x, x_i \rangle$$

$\mathcal{S}$  est l'ensemble support des indices  $i$  tels que  $\hat{\alpha}_i \neq 0$ .

# Noyaux et SVM

- ▶ Si l'on peut calculer les produits scalaires entre observations, on peut ajuster un classifier SVM. Celui ci peut être très abstrait.
- ▶ Quelques fonctions noyaux très particulières le font pour nous.  
Par exemple

$$K(x_i, x_{i'}) = \left(1 + \sum_{j=1}^p x_{ij}x_{i'j}\right)^d$$

calcule les produits scalaires nécessaires pour des polynômes de degré  $d$ .

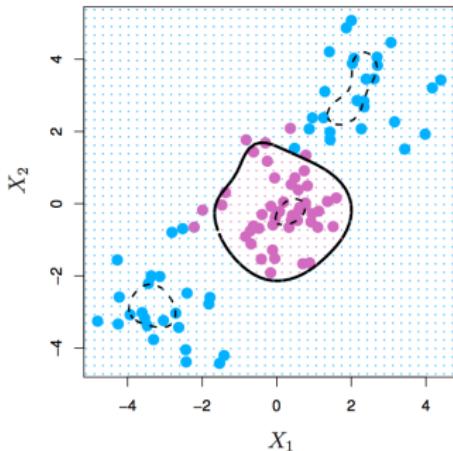
(Essayez avec  $p = 2$  et  $d = 2\dots$

- ▶ La solution est de la forme

$$\hat{f}(x) = \beta_0 + \sum_{i \in \mathcal{S}} \hat{\alpha}_i K(x, x_i)$$

# Noyaux radiaux

$$K(x_i, x_{i'}) = \exp \left( -\gamma \sum_{j=1}^p (x_{ij} - x_{i'j})^2 \right)$$

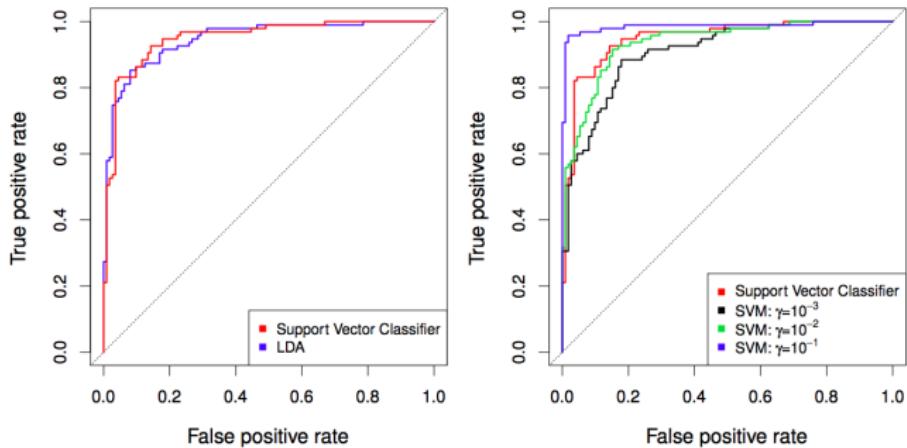


$$\hat{f}(x) = \beta_0 + \sum_{i \in \mathcal{S}} \hat{\alpha}_i K(x, x_i)$$

L'espace des covariables complétées est implicite, mais de très grande dimension.

Contrôle la variance en annulant la plupart des dimensions sévèrement.

## Exemple : données cardiaques



Courbes ROC sur l'ensemble d'entraînement

# SVM sur plus de deux classes ?

Tel que défini précédemment, SVM fonctionne sur des problèmes à deux classes. Que fait-on si le nombre de classes  $K > 2$  ?

- OVA** One Versus All. On ajuste  $K$  classifiers SVM différents à 2 classes  $\hat{f}_k(x)$ ,  $k = 1, \dots, K$  chaque classe contre le reste des autres classes. On prend la décision finale avec la valeurs de  $\hat{f}_k(x)$  la plus grande.
- OVO** One Versus One. On ajuste  $K(K - 1)/2$  classifiers SVM par paires de réponses  $\hat{f}_{k\ell}(x)$ . Décision finale avec la classe qui gagne le plus souvent.

Lequel choisir ? En pratique, si  $K$  est trop grand, utiliser OVO.

# SVM et régression logistique

Avec  $f(X) = \beta_0 + \sum_j \beta_j X_j$ , on peut ré-écrire l'optimisation au cœur de SVM comme

$$\text{minimize}_{(\beta_j, j=1 \dots p)} \left\{ \sum_{i:1}^n \max \left[ 0, 1 - y_i f(x_i) \right] + \lambda \sum_j \beta_j^2. \right\}$$

- ▶ Classes bien séparées : SVM souvent meilleur que la régression logistique.
- ▶ Lorsque ce n'est pas le cas, la régression logistique (avec pénalité ridge) et SVM sont similaires.
- ▶ Pour des frontières de décision non linéaires, les SVM à noyaux sont populaires. On peut utiliser des noyaux dans le calcul de régression logistique et la LDA aussi. Mais les calculs sont beaucoup plus long.

# Interpretability vs Flexibility

