

Introduction to statistical learning

L. Rouvière

laurent.rouviere@univ-rennes2.fr

OCTOBER 2019

Outline

- *15 hours* for this introduction.
- *Materials*: slides + exercises on R.
- *4 parts*:
 1. *Setting for statistical learning*
 2. *Parametric vs non parametric approaches*
 3. *Penalized regressions*
 4. *Trees and random forests*
- *Prerequisites*: basics in probability, statistics (law of large numbers, estimation, bias, variance...) and data mining (linear model, logistic model, linear discriminant analysis...).

Contents

I	Mathematical setting for SL	3
1	Motivations	3
2	Mathematical framework for statistical learning	5
3	Some criterion for regression and supervised classification	7
3.1	Regression	7
3.2	Binary classification	8
3.3	Scoring	8
4	Estimating the risk	11
5	Overfitting	13
6	Bibliography	15
II	Parametric versus nonparametric approaches	16
1	Some parametric methods	17
1.1	Linear and logistic models	17
1.2	Linear discriminant analysis	18
1.2.1	Just one explanatory variable	19
1.2.2	LDA: general case	21
2	Some nonparametric methods	24
2.1	Kernel and nearest neighbors methods	25
2.2	The curse of dimensionality	32

3	Empirical risk minimization	34
3.1	Setting	34
3.2	Caret package	35
4	Bibliography	38
III	Linear model: variable selection and et regularization	40
1	Subset selection	41
2	Penalized regression	44
2.1	Ridge regression	44
2.2	Lasso regression	46
2.3	Supervised classification	48
3	Bibliography	50
IV	Trees	51
1	Binary trees	51
2	Choice of the split	54
2.1	Regression	54
2.2	Supervised classification	55
3	Pruning a tree	57
4	Bibliography	64
V	Bagging and random forests	65
1	Bagging	65
2	Random forests	67
2.1	The algorithm	67
2.2	OOB error	69
2.3	Variable importance	70
3	Bibliography	71

Part I

Mathematical setting for SL

1 Motivations

Statistical learning?

Many "definitions"

1. "... explores way of estimating functional dependency from a given collection of data" [Vapnik, 2000].
2. "...vast set of tools for modelling and understanding complex data" [James et al., 2015].
3. Learn a behavior from **examples**, let the *data* describes mechanisms of a problem.

Statement

- Due to *the digital revolution*, we are faced with more and more *complex* data.
- *Usual* algorithms are *not always efficient* for these kind of data.
- It is necessary to provide efficient algorithms which (**automatically**) **learn** from data.

History - see [Besse and Laurent,]

Period	Memory	Order of magnitude
1940-70	Byte	$n = 30, p \leq 10$
1970	MB	$n = 500, p \leq 10$
1980	MB	Machine Learning (computer science)
1990	GB	Data-Mining
2000	TB	$p > n$, statistical learning
2010	PB	n and p large, cloud, cluster...
2013	??	Big data
2017	??	Artificial Intelligence

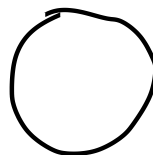
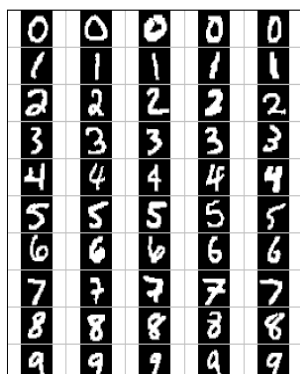
Computer resources \implies

- **Data Mining** (patterns in large datasets, outliers...).
- **Statistical learning** (algorithms that can *automatically* learn from the data) \implies data decides, not the user!

Handwritten recognition

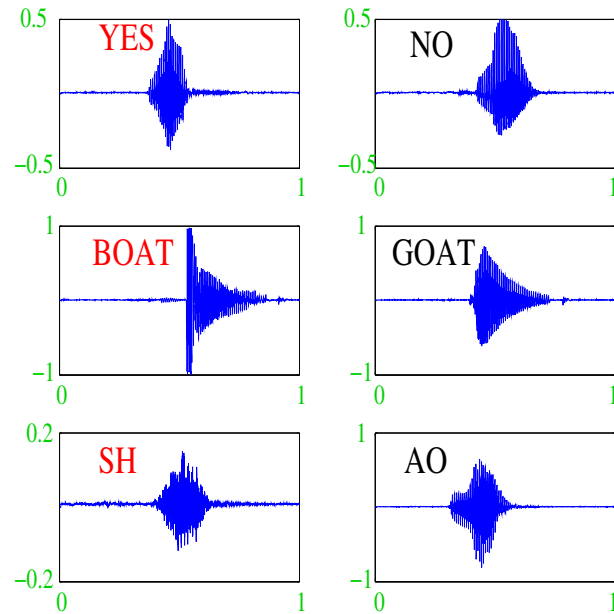
Statistical learning

Understand and learn a behavior from **examples**.



What is the number? 0, 1, 2...?

Speech recognition



Ozone prediction

- During one year, we have measured *ozone concentration* in a city (V4) ;
- Other meteorological variables are available (temperature, nebulosity, wind...).

```
> head(Ozone)
  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

Can we **explain and predict** ozone concentration for tomorrow given meteorological predictions?

Spam detection

- For 4601 emails, we have identified 1813 *spams*.
- In addition to this class label there are *57 variables* indicating the frequency of some words and characters in the e-mail.

```
> 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

From these informations, can we **automatically detect** if a new e-mail is (or not) a spam?

Supervised vs unsupervised learning

- *Supervised learning*: explain/predict an output $y \in \mathcal{Y}$ from inputs $x \in \mathcal{X}$:
 - Linear and logistic models;

- Linear discriminant analysis;
- Tree and random forests...
- *Unsupervised learning*: describe hidden structure from "unlabeled" data (make groups):
 - Hierarchical classifications;
 - k -means algorithms;
 - Mixture models...

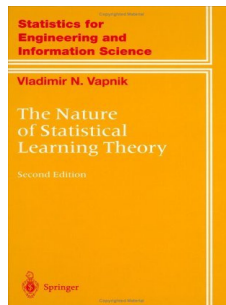
Wide range of applications

finance, economy, marketing, biology, medicine...

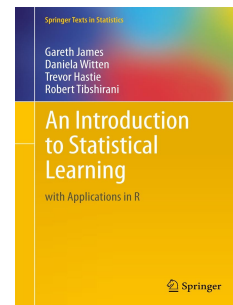
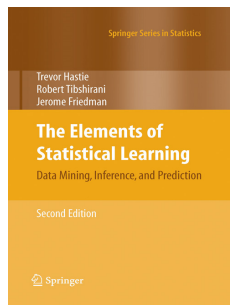
Theory for statistical learning

References

- *Reference book*: [Vapnik, 2000]
- see also [Bousquet et al., 2003].



The Elements of Statistical Learning [Hastie et al., 2009, James et al., 2015]



- Available (with datasets, R commands...) at:

<https://web.stanford.edu/~hastie/ElemStatLearn/> <http://www-bcf.usc.edu/~gareth/ISL/>

2 Mathematical framework for statistical learning

Regression vs supervised classification

- *Input/output data*: $d_n = (x_1, y_1), \dots, (x_n, y_n)$ where $x_i \in \mathcal{X}$ are the inputs $y_i \in \mathcal{Y}$ the outputs.

Goal

1. **Explain** connections between inputs x_i and outputs y_i ;

2. **Predict** the output y for a new input $x \in \mathcal{X}$.

Vocabulary

- When the output \mathcal{Y} is continuous, we are faced with a *regression* problem.
- When the output is categorical ($\text{Card}(\mathcal{Y})$ finite), it is a *supervised classification* problem.

Examples

- Most of the presented problems are *supervised learning* problems: we have to predict an output y by inputs x :

y_i	x_i	
Number	picture	Super. Class.
Word	curve	Super. Class.
Spam	word frequencies	Super. Class.
O_3 concentration	meteo. variables.	Regression

Remark

- Wide range of *input objects* x_i (continuous, categorical, curves, pictures...).

Mathematical framework (begin)

- Given observations $d_n = \{(x_1, y_1), \dots, (x_n, y_n)\}$ we want to *explain/predict* outputs $y_i \in \mathcal{Y}$ from inputs $x_i \in \mathcal{X}$.
- We have to find a *machine (function)* $f : \mathcal{X} \rightarrow \mathcal{Y}$ such that

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

- *Requirement*: a **criterion** to measure **performances** of machines f .
- We use a *cost function* $\ell : \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R}^+$ such that

$$\begin{cases} \ell(y, y') = 0 & \text{if } y = y' \\ \ell(y, y') > 0 & \text{if } y \neq y'. \end{cases}$$

Interpretation

$\ell(y, y')$ measure the *cost (error)* between **one** prediction y' and **one** observation y .

Statistical framework

- We assume that data $d_n = \{(x_1, y_1), \dots, (x_n, y_n)\}$ are *realizations of a n -sample* $\mathcal{D}_n = \{(X_1, Y_1), \dots, (X_n, Y_n)\}$.
- X_i 's are random variables which take values in \mathcal{X} , Y_i 's in \mathcal{Y} .
- We assume that $(X_i, Y_i), i = 1, \dots, n$ are *i.i.d* with the same (**unknown**) distribution \mathbf{P} .

Global performance of a machine f

- For a given **cost function** $\ell : \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R}^+$, we can measure the **global** (for *all possible values* of X and Y) **performance** of a machine $f : \mathcal{X} \rightarrow \mathcal{Y}$ by

$$\ell(Y, f(X))$$

where (X, Y) is independent of (X_i, Y_i) (with same law \mathbf{P}).

- *Problem*: this function is random \implies (very) **difficult to minimize**.

Optimal machine

Risk of a machine

We **measure the performance of a machine** $f : \mathcal{X} \rightarrow \mathcal{Y}$ by *the risk*

$$\mathcal{R}(f) = \mathbf{E}[\ell(Y, f(X))]$$

Theoretical problem

- For the cost function $\ell : \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R}^+$, *theoretical problem* is to find

$$f^* \in \underset{f}{\operatorname{argmin}} \mathcal{R}(f).$$

- Such a function f^* (if it exists) is called *the optimal machine for the cost function ℓ* .

In practice...

- The optimal machine f^* generally depends on the **unknown** probability distribution \mathbf{P} of $(X, Y) \implies f^*$ is **unknown** in practice.
- Statistician's job consists in finding a good **estimate** $f_n = f_n(\cdot, \mathcal{D}_n)$ of $f^* \implies$ we have to find f_n such that $\mathcal{R}(f_n) \approx \mathcal{R}(f^*)$.

Definition

We say that the estimate $(f_n)_n$ is *universally consistant* if for any distribution \mathbf{P}

$$\lim_{n \rightarrow \infty} \mathcal{R}(f_n) = \mathcal{R}(f^*).$$

Choice of the cost function ℓ

- The proposed mathematical framework implies that a machine is performant *with respect to a criterion* (represented by the **cost function** ℓ).
- It means that a machine f could be efficient for a cost function ℓ_1 ($\mathcal{R}_1(f)$ small) but not for another cost function ℓ_2 ($\mathcal{R}_2(f)$ large).

Important conclusion

In practice, it is **crucial** to choose a **relevant cost function** for the problem we are faced.

3 Some criterion for regression and supervised classification

3.1 Regression

Quadratic risk

- In regression ($\mathcal{Y} = \mathbb{R}$), *quadratic cost* is often used. It is defined by

$$\begin{aligned} \ell : \mathbb{R} \times \mathbb{R} &\rightarrow \mathbb{R}^+ \\ (y, y') &\mapsto (y - y')^2 \end{aligned}$$

- *Quadratic risk* for a **machine** or **regression function** $m : \mathcal{X} \rightarrow \mathbb{R}$ is thus defined by

$$\mathcal{R}(m) = \mathbf{E}[(Y - m(X))^2].$$

- The *winner*

$$m^*(x) = \mathbf{E}[Y|X = x]$$

is called the *optimal regression function*.

- Indeed, $\forall m : \mathcal{X} \rightarrow \mathbb{R}$, we have

$$\mathcal{R}(m^*) = \mathbf{E}[(Y - m^*(X))^2] \leq \mathbf{E}[(Y - m(X))^2] = \mathcal{R}(m).$$

Universal consistency

- *Problem:* m^* is unknown in practice. We have to find an estimate $m_n(x) = m_n(x, \mathcal{D}_n)$ such that $m_n(x) \approx m^*(x)$.

Definition

m_n is *universally consistent* if

$$\lim_{n \rightarrow +\infty} \mathcal{R}(m_n) = \mathcal{R}(m^*)$$

for any distribution of (X, Y) .

3.2 Binary classification

Error probability

- Binary classification means that output can only take 2 values ($\mathcal{Y} = \{-1, 1\}$). In this case, we often use the *0-1 loss function*:

$$\begin{aligned} \ell : \{-1, 1\} \times \{-1, 1\} &\rightarrow \mathbb{R}^+ \\ (y, y') &\mapsto \mathbf{1}_{y \neq y'} \end{aligned}$$

- The *risk* for a **classification rule** $g : \mathcal{X} \rightarrow \{-1, 1\}$ is given by

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

- The *winner* (called the *Bayes rule*) is

$$g^*(x) = \begin{cases} -1 & \text{if } \mathbf{P}(Y = -1|X = x) \geq \mathbf{P}(Y = 1|X = x) \\ 1 & \text{otherwise.} \end{cases}$$

- For any classification rule g ,

$$L^* = L(g^*) = \mathbf{P}(g^*(X) \neq Y) \leq \mathbf{P}(g(X) \neq Y) = L(g).$$

Universal consistency

- *Problem:* g^* is unknown in practice. We have to find $g_n(x) = g_n(x, \mathcal{D}_n)$ such that $g_n(x) \approx g^*(x)$.

Definition

$(g_n)_n$ is *universally consistent* if

$$\lim_{n \rightarrow \infty} L(g_n) = L(g^*) = L^*$$

for any distribution of (X, Y) .

\implies *Exercise 1 - IML0.*

3.3 Scoring

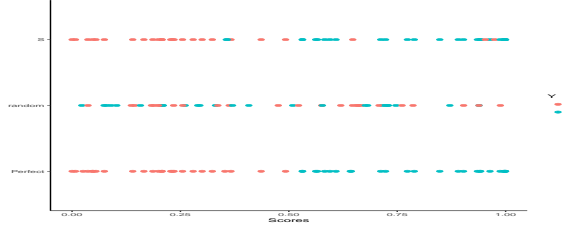
Scoring function

- Always in *binary classification* ($\mathcal{Y} = \{-1, 1\}$).
- But... instead of a classification rule $g : \mathcal{X} \rightarrow \{-1, 1\}$, we want to find a function $S : \mathcal{X} \rightarrow \mathbb{R}$ such that

$$\begin{array}{c} \text{P}(Y = 1) \text{ small} \qquad \qquad \qquad \text{P}(Y = 1) \text{ large} \\ \hline \hspace{10em} \longrightarrow S(x) \end{array}$$

- Such a function is a **score function**: instead of predicting the label y of a new $x \in \mathcal{X}$, we provide a *score* $S(x)$
 - **high** if we think that x is 1;
 - **small** if we think that x is -1.

Perfect and random scores



Definition 1. • *Perfect score:* S is **perfect** if there exists s^* such that

$$\mathbf{P}(Y = 1|S(X) \geq s^*) = 1 \quad \text{and} \quad \mathbf{P}(Y = -1|S(X) < s^*) = 1.$$

• *Random score:* S is **random** if $S(X)$ and Y are independents.

Link between a score and a classification rule

• For a given **score** S and a **threshold** s , we obtain a *classification rule*:

$$g_s(x) = \begin{cases} 1 & \text{if } S(x) \geq s \\ -1 & \text{otherwise.} \end{cases}$$

• We have

	$g_s(X) = -1$	$g_s(X) = 1$
$Y = -1$	OK	E_1
$Y = 1$	E_2	OK

• For any threshold s , we can define 2 errors:

$$\alpha(s) = \mathbf{P}(g_s(X) = 1|Y = -1) = \mathbf{P}(S(X) \geq s|Y = -1)$$

and

$$\beta(s) = \mathbf{P}(g_s(X) = -1|Y = 1) = \mathbf{P}(S(X) < s|Y = 1).$$

We can also define

- *Specificity:* $sp(s) = \mathbf{P}(S(X) < s|Y = -1) = 1 - \alpha(s)$;
- *Sensibility:* $se(s) = \mathbf{P}(S(X) \geq s|Y = 1) = 1 - \beta(s)$.

Performance of a score

Visualize errors $\alpha(s)$ and $\beta(s)$ on a same graph for all **thresholds** s .

ROC curve

• **Idea:** define a 2-dimensionnel graph to represent errors $\alpha(s)$ and $\beta(s)$ for all values of s .

Definition

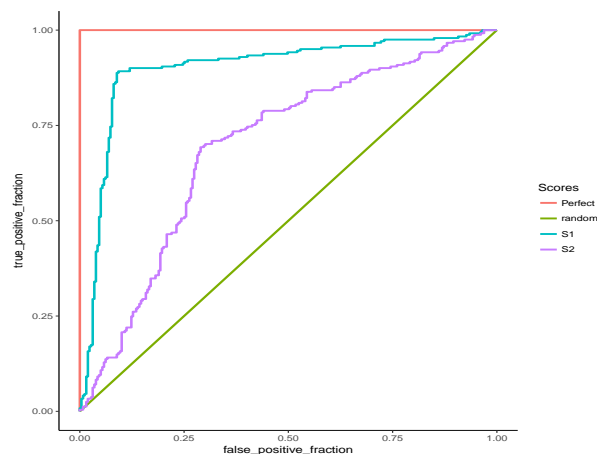
The *ROC curve* of a score S is the *parametrized curve* defined by

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

Remark

• For any score S : $x(-\infty) = y(-\infty) = 1$ and $x(+\infty) = y(+\infty) = 0$.

- For a **perfect score**: $x(s^*) = 0$ and $y(s^*) = 1$.
- For a **random score**: $x(s) = y(s) \forall s$.



Interpretation

We measure performance of a score by its *ability to approach the line $y = 1$* as fast as possible.

AUC

Definition

- *Area Under ROC* for a score S , denoted $AUC(S)$ is often used to measure performance of a S .
- *Perfect score*: $AUC(S) = 1$. **Random score**: $AUC(S) = 1/2$.

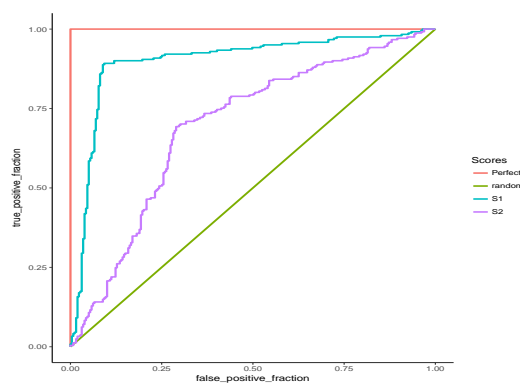
Proposition 3.1. • Let (X_1, Y_1) et (X_2, Y_2) be 2 *i.i.d.* observations. Then

$$AUC(S) = \mathbf{P}(S(X_1) \geq S(X_2) | (Y_1, Y_2) = (1, -1)).$$

Conclusion

$AUC(S)$ measures the probability that S **correctly orders** two observations with **different labels**.

Example



```
> df1 %>% group_by(Scores) %>% summarize(auc(D,M))
1 Perfect 1.0000000
2 random 0.5000000
3 S1 0.8999824
4 S2 0.6957177
```

	Cost $\ell(y, f(x))$	Risk $\mathbf{E}[\ell(Y, f(X))]$	Winner f^*
Regression	$(y - f(x))^2$	$\mathbf{E}[Y - f(X)]^2$	$\mathbf{E}[Y X = x]$
Binary class.	$\mathbf{1}_{y \neq f(x)}$	$\mathbf{P}(Y \neq f(X))$	Bayes rule
Scoring		$AUC(S)$	$\mathbf{P}(Y = 1 X = x)$

Optimal score

- $AUC(S)$ can be seen as a *cost function* for a score S ;
- *Question*: is there an *optimal score* S^* for this cost function?

Theorem 2 ([Cl  men  on et al., 2008]). *Let $S^*(x) = \mathbf{P}(Y = 1|X = x)$, then for any score S we have*

$$AUC(S^*) \geq AUC(S).$$

Consequence

We have to find a "good" estimate $S_n(x) = S_n(x, \mathcal{D}_n)$ of

$$S^*(x) = \mathbf{P}(Y = 1|X = x).$$

Summary

4 Estimating the risk

Reminder

- n observations $(X_1, Y_1), \dots, (X_n, Y_n)$ i.i.d in $\mathcal{X} \times \mathcal{Y}$.

Goal

Given a cost function $\ell : \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R}^+$, we search a machine $f_n(x) = f_n(x, \mathcal{D}_n)$ closed to the optimal machine f^* defined by

$$f^* \in \underset{f}{\operatorname{argmin}} \mathcal{R}(f)$$

where $\mathcal{R}(f) = \mathbf{E}[\ell(Y, f(X))]$.

Question

Given a machine f_n , what can we say about its risk $\mathcal{R}(f_n)$?

Empirical risk

- Since the distribution of (X, Y) is *unknown*, we *can't compute* $\mathcal{R}(f_n) = \mathbf{E}[\ell(Y, f_n(X))]$.
- **First idea**: $\mathcal{R}(f_n)$ is an expectation, estimate it by its *empirical version* (law of large numbers)

$$\mathcal{R}_n(f_n) = \frac{1}{n} \sum_{i=1}^n \ell(Y_i, f_n(X_i)).$$

Problem

- \mathcal{D}_n has *already been used* to construct the machine $f_n \implies$ LLN does not apply!
- *Consequence*: $\mathcal{R}_n(f_n)$ generally *underestimates* $\mathcal{R}(f_n)$.

One solution

Cross validation or bootstrap approaches.

Validation hold out

- The simplest approach.
- It consists in splitting the data \mathcal{D}_n into:
 1. a *learning or training set* $\mathcal{D}_{n,train}$ used to learn a machine f_n ;
 2. a *validation or test set* $\mathcal{D}_{n,test}$ to estimate the risk of f_n .

Algorithm

Inputs. \mathcal{D}_n : data, $\{\mathcal{T}, \mathcal{V}\}$: a partition of $\{1, \dots, n\}$.

1. Learn the machine with $\mathcal{D}_{n,train} = \{(X_i, Y_i) : i \in \mathcal{T}\} \Rightarrow f_{n,train}$;
2. Compute $\widehat{\mathcal{R}}_n(f_n) = \frac{1}{|\mathcal{V}|} \sum_{i \in \mathcal{V}} \ell(Y_i, f_{n,train}(X_i))$.

Comments

n_{train} and n_{test} should be *large enough* to

1. fit $f_{n,train}$;
2. estimate its risk $\mathcal{R}(f_{n,train})$.

K fold cross-validation

- **Idea:** repeat *validation hold out* algorithm on each element of a *data partition*.

Algorithm - CV

Inputs. \mathcal{D}_n : data, K an integer ;

1. Define a random partition $\{\mathcal{I}_1, \dots, \mathcal{I}_K\}$ of $\{1, \dots, n\}$;
2. For $k = 1, \dots, K$
 - (a) $\mathcal{I}_{train} = \{1, \dots, n\} \setminus \mathcal{I}_k$ and $\mathcal{I}_{test} = \mathcal{I}_k$;
 - (b) Learn the machine with $\mathcal{D}_{n,app} = \{(X_i, Y_i) : i \in \mathcal{I}_{app}\} \Rightarrow f_{n,k}$;
 - (c) Let $f_n(X_i) = f_{n,k}(X_i)$ for $i \in \mathcal{I}_{test}$;

3. Output

$$\widehat{\mathcal{R}}_n(f_n) = \frac{1}{n} \sum_{i=1}^n \ell(Y_i, f_n(X_i)).$$

Comments

- More useful than validation hold out when n is *small*.
- K has to be *chosen* by the user (we often set $K = 10$).

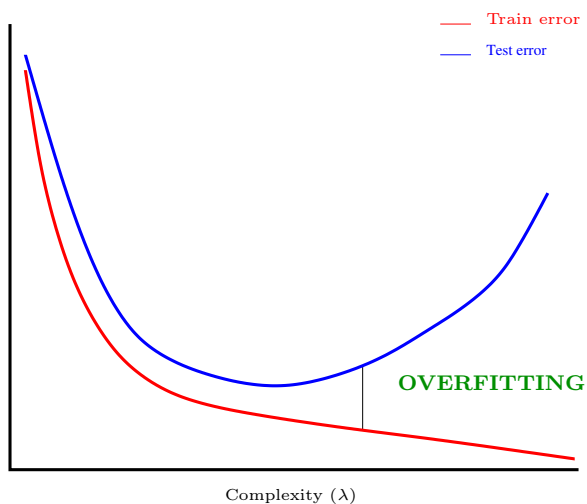
Leave one out

- When $K = n$, we obtain *leave one out* cross validation.
- Risk is estimated by

$$\widehat{\mathcal{R}}_n(f_n) = \frac{1}{n} \sum_{i=1}^n \ell(Y_i, f_n^i(X_i))$$

where f_n^i stands for the machine defined on \mathcal{D}_n after deleted the i th observation.

- *Exercises 1-3, IML1.*



5 Overfitting

- Most of statistical learning algorithms depends on *parameters* (λ).

Examples

- number of input variables in linear and logistic models.
- penalty parameters for lasso and ridge regressions.
- depth for tree algorithms.
- number of nearest neighbors.
- bandwidth of kernel regression estimators.
- number of iterations for boosting algorithms.
- ...
- The choice of these parameters reveals *crucial* for the **performance of the machine**.
- Parameter λ often measures *model complexity*.

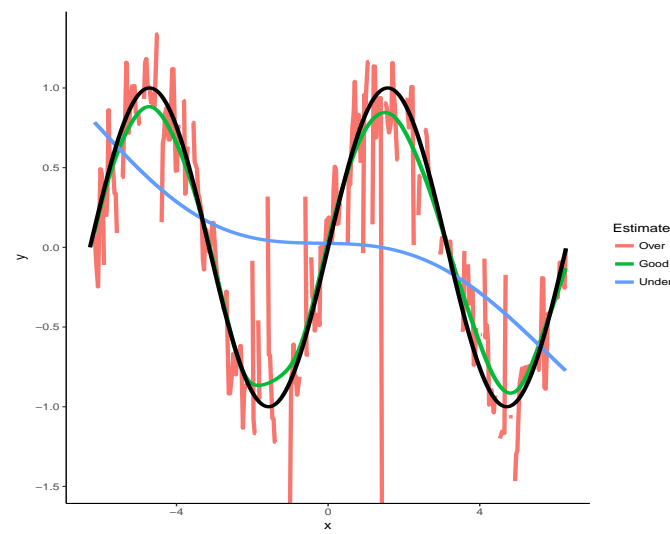
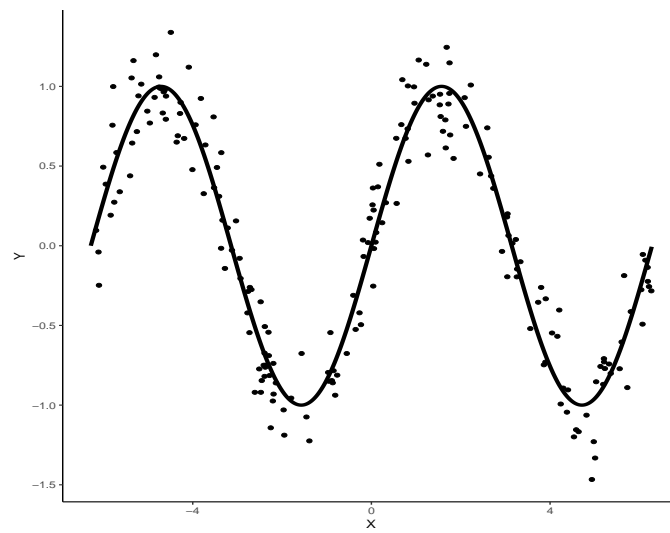
Model complexity

- λ small \implies restrictive model \implies bad fitting \implies bias \nearrow , variance \searrow
- λ large \implies flexible (complex) model \implies **overfitting** \implies bias \searrow , variance \nearrow

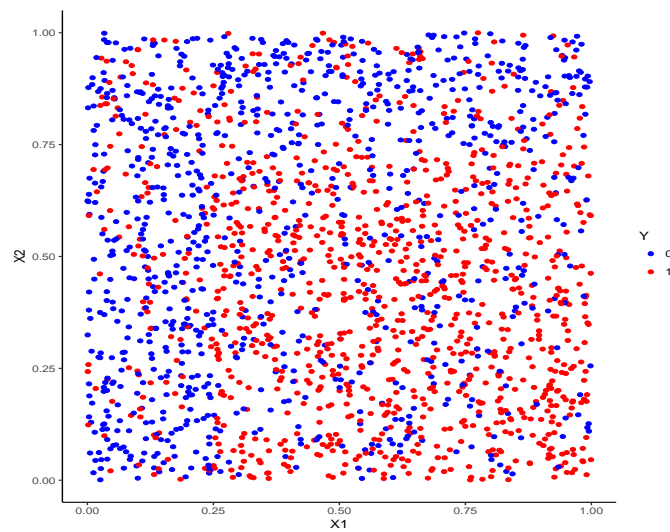
Overfitting

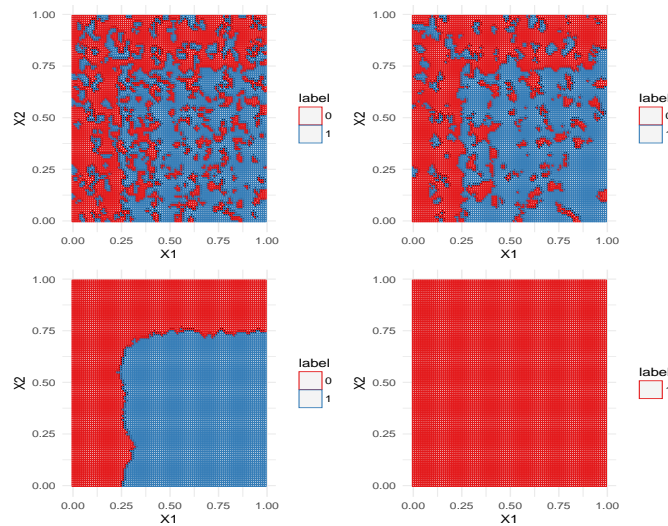
Good fitting on the training data (i.e. $f(X_i) = Y_i$) but **poor predictive** performances on **new individuals**.

Overfitting for regression



Overfitting for supervised classification





- Run application *overfitting.app*.

6 Bibliography

References

Bibliol

- [Besse and Laurent,] Besse, P. and Laurent, B. *Apprentissage Statistique modélisation, prévision, data mining*. INSA - Toulouse. http://www.math.univ-toulouse.fr/~besse/pub/Appren_stat.pdf.
- [Bousquet et al., 2003] Bousquet, O., Boucheron, S., and Lugosi, G. (2003). *Introduction to Statistical Learning Theory*, chapter Advanced Lectures on Machine Learning. Springer.
- [Cléménçon et al., 2008] Cléménçon, S., Lugosi, G., and Vayatis, N. (2008). Ranking and empirical minimization of u-statistics. *The Annals of Statistics*, 36(2):844–874.
- [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.
- [James et al., 2015] James, G., Witten, D., Hastie, T., and Tibshirani, R. (2015). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer.
- [Vapnik, 2000] Vapnik, V. (2000). *The Nature of Statistical Learning Theory*. Springer, second edition.

Part II

Parametric versus nonparametric approaches

Mathematical framework

- n i.i.d observations $(X_1, Y_1), \dots, (X_n, Y_n)$ in $\mathcal{X} \times \mathcal{Y}$.
- $\ell : \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R}^+$ cost function.

Problem

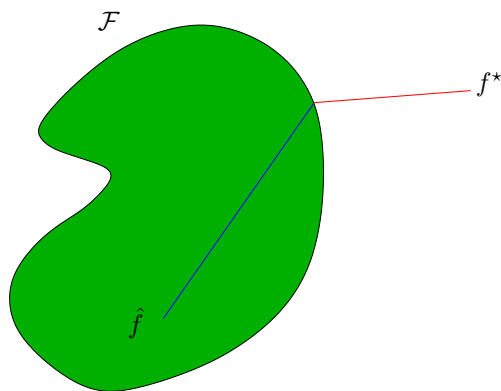
Estimate

$$f^* \in \underset{f}{\operatorname{argmin}} \mathcal{R}(f) = \mathbf{E}[\ell(Y, f(X))]$$

by $f_n(\cdot) = f_n(\cdot, \mathcal{D}_n)$.

Model

- Modelize remains to fix a class of functions \mathcal{F} and to *assume that* $f^* \in \mathcal{F}$.
- Modelize = make *an assumption*.



Given $(X_1, Y_1), \dots, (X_n, Y_n)$, find the *best* machine $f \in \mathcal{F}$.

$$\begin{aligned} \mathcal{R}(\hat{f}) - \mathcal{R}^* &= \mathcal{R}(\hat{f}) - \inf_{f \in \mathcal{F}} \mathcal{R}(f) + \inf_{f \in \mathcal{F}} \mathcal{R}(f) - \mathcal{R}^*. \\ &= \text{Estimation error} + \text{Approximation error}. \end{aligned}$$

Remarks

- These two terms vary in **opposite directions**.
- Statistician's job: **trade-off** between these two terms.

Parametric and non parametric

Definition 3. • If $\mathcal{F} = \{f_\theta : \theta \in \Theta\}$ with Θ of finite dimension, then the model is *parametric*.

- If \mathcal{F} is an infinite dimensional space, then the model is *non-parametric*.

Remark

- **Non-parametric** seems more interesting (since **less restrictive**).
- There is a price to be paid... More **difficult** to estimate for such models.
- *Loss of accuracy* in NP models. In this part, we will study this loss.

1 Some parametric methods

1.1 Linear and logistic models

The linear model

- In *regression* with $\mathcal{X} = \mathbb{R}^d$, the *linear model* is the *parametric* reference model.
- This model makes the assumption that the *regression function is linear*:

$$m^*(x) = \mathbf{E}[Y|X = x] = \beta_1 x_1 + \dots + \beta_d x_d.$$

- Or equivalently

$$Y = \beta_1 X_1 + \dots + \beta_d X_d + \varepsilon$$

where $\mathbf{E}[\varepsilon|X = x] = 0$ and $\mathbf{V}[\varepsilon|X = x] = \sigma^2$.

Remark

Estimate $m^* \iff$ estimate $\beta \in \mathbb{R}^d$ (finite dimension \implies parametric model).

Some properties

- *Least squares* estimates minimize

$$\sum_{i=1}^n \varepsilon_i^2 = \sum_{i=1}^n (Y_i - (\beta_1 X_{i1} + \dots + \beta_d X_{id}))^2.$$

The solution is given by

$$\hat{\beta}_n = (\mathbb{X}^t \mathbb{X})^{-1} \mathbb{X}^t \mathbb{Y}.$$

- Regression function m^* is thus estimated by

$$\hat{m}_n(x) = \hat{\beta}_1 x_1 + \dots + \hat{\beta}_d x_d.$$

Proposition 1.1. *Under some technical assumptions, we prove that*

- $\mathbf{E}[\hat{\beta}] = \beta$ and $\mathbf{V}[\hat{\beta}] = (\mathbb{X}^t \mathbb{X})^{-1} \sigma^2$.

We deduce that (exercise 2, IML0)

$$\mathbf{E}[\|\hat{\beta} - \beta\|^2] = O\left(\frac{1}{n}\right) \quad \text{and} \quad \mathbf{E}[(\hat{m}_n(x) - m^*(x))^2] = O\left(\frac{1}{n}\right).$$

Remark

- Least squares estimates achieve the *parametric rate* ($1/n$).
- Moreover, if *errors terms* $\varepsilon_i, i = 1, \dots, n$ are *Gaussian*, we can compute *the distribution of the least squares estimates* (confidence intervals, test statistics...).
- See [Grob, 2003, Cornillon and Matzner-Løber, 2011] for more information.

Example

- Linear model to explain *ozone concentration*.

```
> model_lin <- lm(V4~V5+V6+V7+V8+V9+V10+V11+V12+V13,data=Ozone)
> summary(model_lin)
Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  59.9517553  38.3286940   1.564  0.119421
V5           -0.0139111   0.0072511  -1.918  0.056527 .
V6             0.0276862   0.1741433   0.159  0.873847
V7             0.0808740   0.0237694   3.402  0.000812 ***
V8             0.1503404   0.0692994   2.169  0.031272 *
V9             0.5253439   0.1247136   4.212  3.87e-05 ***
V10           -0.0010052   0.0003944  -2.549  0.011586 *
V11             0.0049796   0.0147772   0.337  0.736501
V12           -0.1543882   0.1192917  -1.294  0.197140
V13           -0.0033951   0.0048963  -0.693  0.488883
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Logistic model

- *Logistic model* is the "brother" of the linear model in the context of *binary classification* ($\mathcal{Y} = \{-1, 1\}$).
- This model makes the **assumption** that (the logit transformation of) the probability $p(x) = \mathbf{P}(Y = 1|X = x)$ is *linear* :

$$\text{logit } p(x) = \log \frac{p(x)}{1 - p(x)} = \beta_1 x_1 + \dots + \beta_d x_d = x^t \beta.$$

- $\beta = (\beta_1, \dots, \beta_d) \in \mathbb{R}^d \implies$ **parametric model**.
- Unknown parameters β_1, \dots, β_d are estimated by *maximizing the (log)-likelihood*:

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

Some properties

Theorem [Fahrmeir and Kaufmann, 1985]

Under technical assumptions we have

1. the ML estimate $\{\hat{\beta}_n\}_n$ is **consistant**: $\hat{\beta}_n \xrightarrow{\mathbf{P}} \beta$;
2. the ML estimate $\{\hat{\beta}_n\}_n$ is **asymptotically gaussian**:

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

3. **Rate of convergence**:

$$\mathbf{E}[\|\hat{\beta} - \beta\|^2] = O\left(\frac{1}{n}\right).$$

Important remark

Again, the ML estimate *achieves the parametric rate* $(1/n)$.

Example

- In *R*, we can fit a logistic model with the **glm** function.

```
> model_log <- glm(type~., data=spam, family=binomial)
> summary(model_log)$coefficients[1:5,]
              Estimate Std. Error z value Pr(>|z|)
(Intercept) -1.5686144  0.1420362 -11.043767 2.349719e-28
make         -0.3895185  0.2314521  -1.682933 9.238799e-02
address      -0.1457768  0.0692792  -2.104194 3.536157e-02
all           0.1141402  0.1103011  1.034806 3.007594e-01
num3d         2.2515195  1.5070099  1.494031 1.351675e-01
```

1.2 Linear discriminant analysis

Presentation

- Logistic regression *directly modelizes* the parameter of the distribution of $Y|X = x$.
- *Linear discriminant analysis* do the opposite. It consists in
 - *modelizing* the distributions of $X|Y = j$ for $j = 1, \dots, K$ by gaussian distributions $f_j(x)$.
 - calculating the posterior distribution $Y|X = x$ with *Bayes formula* :

$$\mathbf{P}(Y = j|X = x) = \frac{\pi_j f_j(x)}{\sum_{\ell=1}^K \pi_\ell f_\ell(x)}$$

where $\pi_j = \mathbf{P}(Y = j), j = 1, \dots, K$.

Example: Fisher's iris problem

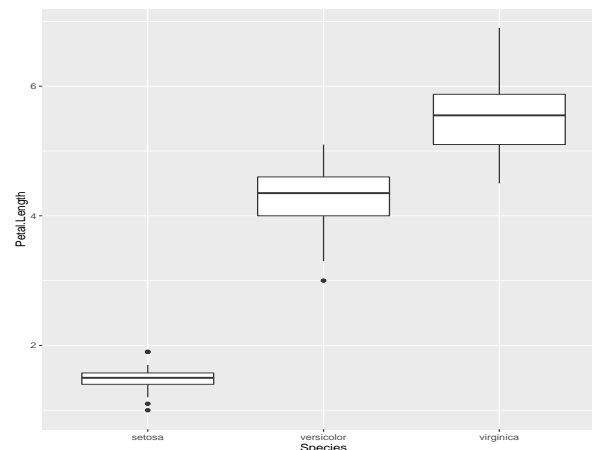
- Explain *iris species* by lengths and widths of petals and sepals.
- 5 variables :
 - the target variable *species* (categorical).
 - lengths and widths of petals and sepals.

```
> summary(iris)
      Sepal.Length   Sepal.Width   Petal.Length   Petal.Width
Min.   :4.300      Min.   :2.000    Min.   :1.000    Min.   :0.100
1st Qu.:5.100      1st Qu.:2.800    1st Qu.:1.600    1st Qu.:0.300
Median :5.800      Median :3.000    Median :4.350    Median :1.300
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
      Species
setosa   :50
versicolor:50
virginica :50
```

1.2.1 Just one explanatory variable

- We first want to explain *Species* by
- We can draw the following *boxplot*.

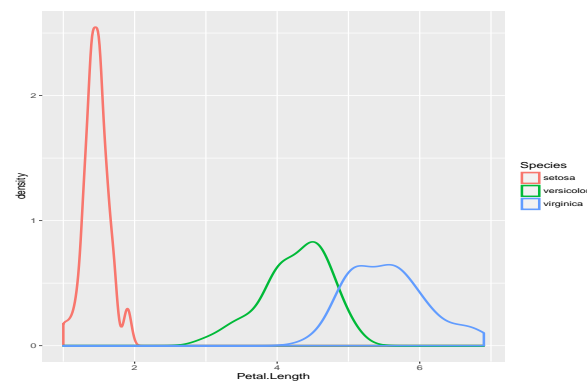
```
> ggplot(iris)+aes(x=Species,y=Petal.Length)+geom_boxplot()+theme_bw()
```



Visualize densities

- *geom_density* allows to visualize conditional distributions of $X|Y = j, j = 1, 2, 3$.

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



A model

- The three densities on the graph look like *Gaussian* densities.
- Let $X = \text{Petal.Length}$ and $Y = \text{Species}$. We assume that distributions of X given $Y = k$ are Gaussians $\mathcal{N}(\mu_k, \sigma^2)$, $k = 1, 2, 3$.
- Densities of $X|Y = k$ are thus given by

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

Estimation

- To obtain *posterior probabilities* $\mathbf{P}(Y = k|X = x)$, we have to estimate:
 - parameters μ_k et σ^2 of the Gaussian distributions;
 - prior probabilities $\pi_k = \mathbf{P}(Y = k)$.

Estimators

These quantities are naturally **estimated** by

$$\hat{\mu}_k = \frac{1}{n_k} \sum_{i:Y_i=k} X_i, \quad \widehat{\sigma^2} = \frac{1}{n-2} \sum_{k=1}^K \sum_{i:Y_i=k} (X_i - \hat{\mu}_k)^2$$
$$\hat{\pi}_k = \frac{n_k}{n} \quad \text{where} \quad n_k = \sum_{i=1}^n \mathbf{1}_{\{Y_i=k\}}.$$

Example with R

```
> library(MASS)
> 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
```

Making predictions

- **predict** function allows to *predict* species of *new iris*:

```
> don_pred
      Sepal.Length Sepal.Width Petal.Length Petal.Width
1         5.0         3.6         1.4         0.2
2         5.5         2.4         3.7         1.0
3         7.1         3.0         5.9         2.1
4         6.7         3.3         5.7         2.5
```

- We just have to enter

```
> 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
```

1.2.2 LDA: general case

- *Goal*: **explain** iris specie by the 4 explanatory variables `Sepal.Length`, `Sepal.Width`, `Petal.Length`, `Petal.Width`. We denote by X_1, X_2, X_3, X_4 these 4 variables and $X = (X_1, X_2, X_3, X_4)$.
- The approach is *similar* to the previous case (1 variable)
 1. We model distributions of $X|Y = k$ by *Gaussian multivariate* distributions.
 2. We use *Bayes formula* to obtain posterior probabilities $\mathbf{P}(Y = k|X = x)$.

LDA: general case

- Distributions of $X|Y = k$ are **are assumed to be Gaussians** $\mathcal{N}(\mu_k, \Sigma)$ where $\mu_k \in \mathbb{R}^p$ and Σ is a $p \times p$ definite positive matrix. Densities of $X|Y = k$ are thus given by:

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

- Posterior probabilities $\mathbf{P}(Y = k|X = x)$ are obtained thanks to the **Bayes formula**

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

where $f(x)$, the density of X , is computed from $f_{X|Y=k}(x)$ and from prior probabilities $\pi_k = \mathbf{P}(Y = k)$.

Estimation

- We again need to estimate *unknown parameters* of the model:
 - mean vectors $\mu_k, k = 1, \dots, K$ and covariance matrix Σ of the Gaussian distributions;
 - prior probabilities $\pi_k = \mathbf{P}(Y = k)$.

Estimators

They are defined by

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

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

Example with R

```
> full_model<- lda(Species~.,data=iris)
> full_model
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
```

Making predictions

- **predict** function allow to *predict* species for *new iris*

```
> 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
```

- We just have to enter

```
> 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
```

Classification rule

- *Reminder*: LDA allows to estimate posterior probabilities:

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

- *Classification rule*: we choose the group which maximizes these probabilities

$$\hat{g}(x) = k \quad \text{if and only if} \quad \mathbf{P}(Y = k|X = x) \geq \mathbf{P}(Y = j|X = x), \quad j \neq k.$$

- *Boundary between 2 groups*: set of points x such that $\mathbf{P}(Y = k|X = x) = \mathbf{P}(Y = j|X = x)$.
- Or

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

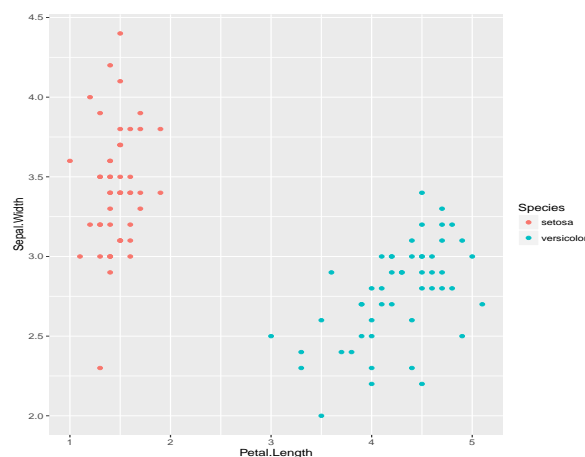
Conclusion

Bondary between 2 groups is **linear**!

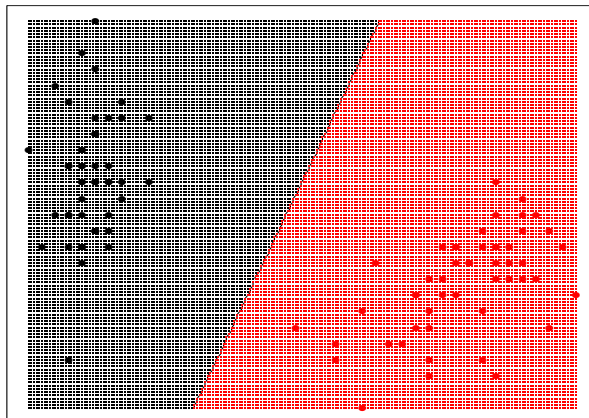
Example

- *Boundary* between "Setosa" and "Versicolor" for 2 variables.

```
> iris1 <- iris[iris$Species%in%c("setosa","versicolor"),c(3,2,5)]
> ggplot(iris1)+aes(x=Petal.Length,y=Sepal.Width,color=Species)+geom_point()
```



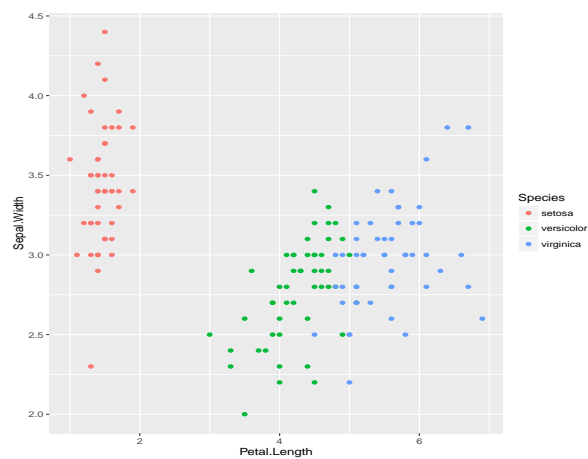
Boundary two classes



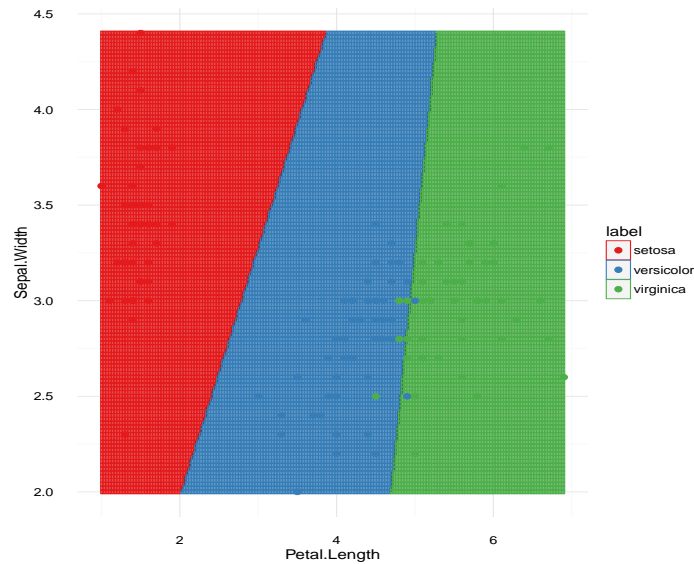
Example - 3 labels

- We do the same for the 3 species (3 classes).

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



Boundaries



Linear discriminant functions

Definition

Linear discriminant functions are defined by

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

Propriété

Thanks to (1), we deduce

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

Conclusion

Maximising posterior probabilities is similar to maximising linear discriminant functions.

2 Some nonparametric methods

Local averaging

Idea

- Parametric models require *strong assumptions* on the function to estimate.
- Nonparametric approaches try to be less *restrictive*.
- These methods consists of studying the data on a *neighborhood* of the points where we want to estimate the target function.
- For both regression and supervised classification, nonparametric approaches rely on *local averaging*:

$$\hat{f}_n(x) = \sum_{i=1}^n W_{ni}(x) Y_i$$

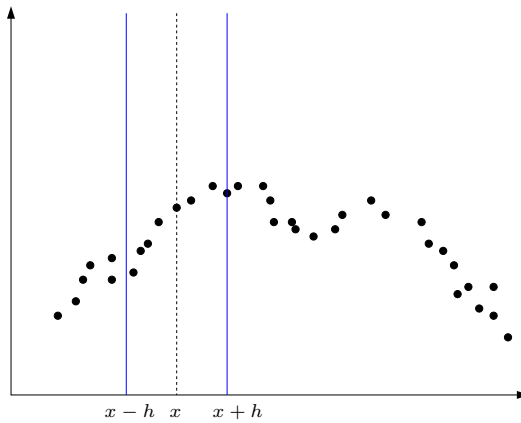
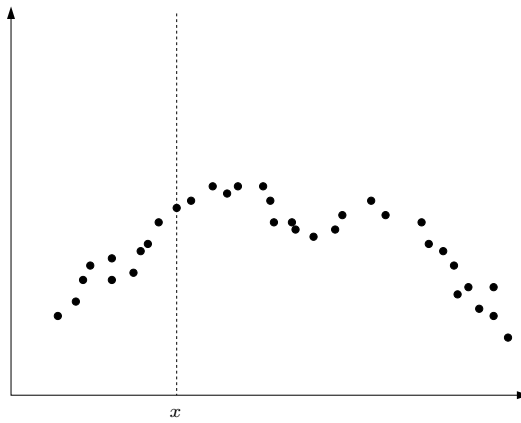
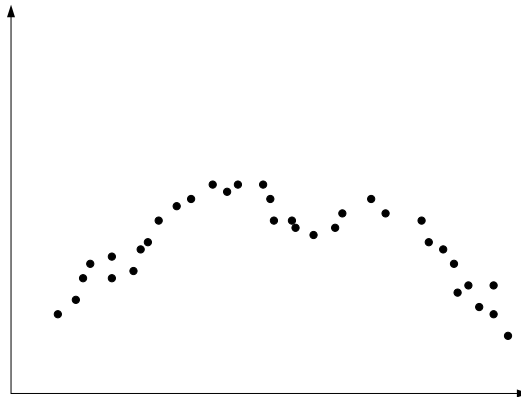
where the weights W_{ni} depend on the algorithm.

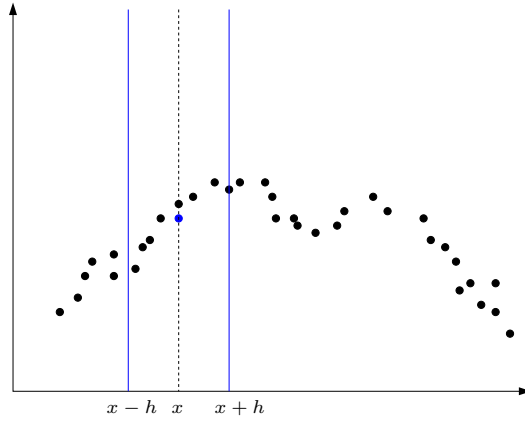
- W_{ni} large if X_i is closed to x .

2.1 Kernel and nearest neighbors methods

Kernel estimate

- $(X_1, Y_1), \dots, (X_n, Y_n)$ i.i.d. with the same law as (X, Y) .
- *Goal:* estimate $m^*(x) = \mathbf{E}[Y|X = x]$.





- The *estimator*

$$\hat{m}_n(x) = \text{Average}(Y_i : X_i \in [x-h, x+h]) = \frac{\sum_{i=1}^n \mathbf{1}_{x-h \leq X_i \leq x+h} Y_i}{\sum_{i=1}^n \mathbf{1}_{x-h \leq X_i \leq x+h}}.$$

Definition 4. Let $h > 0$ and $K : \mathcal{X} \rightarrow \mathbb{R}^+$. The **kernel estimate with bandwidth h and kernel K** is defined by

$$\hat{m}_n(x) = \frac{\sum_{i=1}^n K\left(\frac{X_i - x}{h}\right) Y_i}{\sum_{i=1}^n K\left(\frac{X_i - x}{h}\right)}.$$

Choice of the bandwidth

- *Usual kernels* when $\mathcal{X} = \mathbb{R}^d$:

1. **Uniform**: $K(x) = \mathbf{1}_{\|x\| \leq 1}$;
2. **Gaussian**: $K(x) = \exp(-\|x\|^2)$;
3. **Epanechnikov**: $K(x) = \frac{3}{4}(1 - \|x\|^2)\mathbf{1}_{\|x\| \leq 1}$.

\implies provide weights according to the distance of x .

- The choice of the bandwidth h reveals *crucial* for the performance of the estimate:

1. **h large**: steady estimator, low variance, high bias;
2. **h small**: unsteady estimator ("overfitting"), high variance, small bias.

Conclusion

h governs the *complexity* of the estimate.

Example

- We generate data $(X_i, Y_i), i = 1, \dots, n = 200$ according to the model

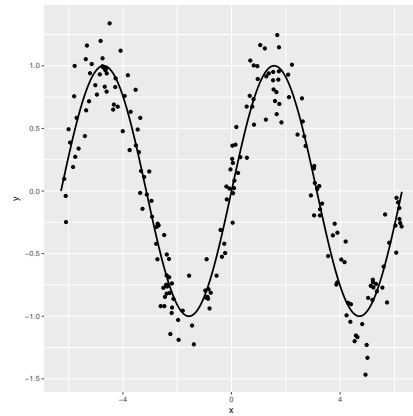
$$Y_i = \sin(X_i) + \varepsilon_i, \quad i = 1, \dots, n$$

where X_i has a uniform distribution on $[-2\pi, 2\pi]$, ε_i has a Gaussian distribution $\mathcal{N}(0, 0.2^2)$.

```

> n <- 200; set.seed(1234)
> X <- runif(n,-2*pi,2*pi)
> set.seed(5678)
> eps <- rnorm(n,0,0.2)
> Y <- sin(X)+eps
> df <- data.frame(X=X,Y=Y)
> x <- seq(-2*pi,2*pi,by=0.01)
> df1 <- data.frame(x=x,y=sin(x))
> ggplot(df1)+aes(x=x,y=y)+
  geom_line(size=1)+
  geom_point(data=df,aes(x=X,y=Y))

```

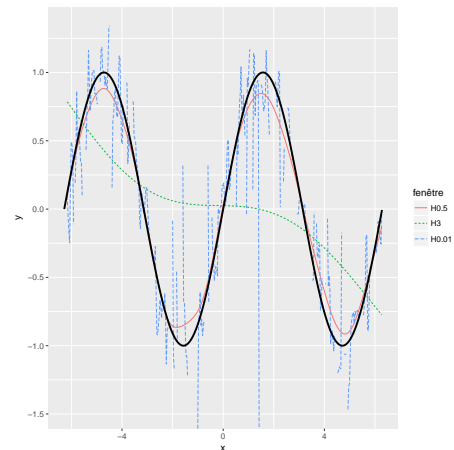


- **locpoly** function from **kernSmooth** package allows to fit kernel estimates.

```

> h1 <- 0.5;h2 <- 3;h3 <- 0.01
> fx1 <- locpoly(X,Y,bandwidth=h1)
> fx2 <- locpoly(X,Y,bandwidth=h2)
> fx3 <- locpoly(X,Y,bandwidth=h3)
> df1 <- data.frame(x=x,y=sin(x))
> df2 <- data.frame(x=fx1$x,
  "H0.5"=fx1$y,"H3"=fx2$y,
  "H0.01"=fx3$y)
> df22 <- melt(df2,id.vars=1)
> names(df22)[2:3] <- c("fenêtre",
  "y")
> ggplot(df22)+aes(x=x,y=y)+
  geom_line(aes(color=fenêtre,
    lty=fenêtre))+geom_line
  (data=df1,aes(x=x,y=y),size=1)

```



- *Exercise 4-IML1.*

Nearest neighbors algorithm

Definition 5. Let $k \leq n$ an integer. The k -nearest neighbors estimate is defined by

$$\hat{m}_n(x) = \frac{1}{k} \sum_{i \in \text{knn}(x)} Y_i$$

where for $x \in \mathcal{X}$

$$\text{knn}(x) = \{i : X_i \text{ is among the knn of } x \text{ among } \{X_1, \dots, X_n\}\}.$$

Remark

Once again, the choice of k reveals **crucial** for the performance of the:

1. **k large**: steady estimate, low variance, high bias;
2. **k small**: "overfitting", large variance, small bias.

$\implies k$ governs the *complexity* of the model.

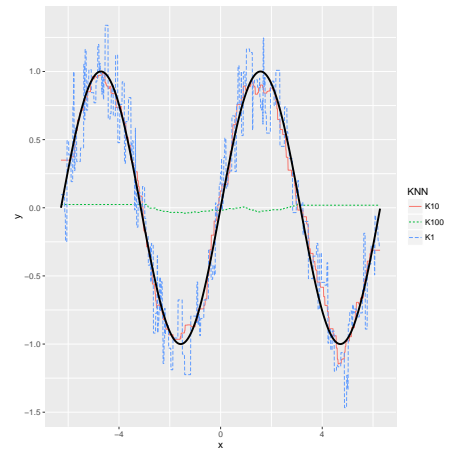
Example

- **knn.reg** function from **FNN** package allows to fit k -nearest neighbors estimate.

```

> k1 <- 10; k2 <- 100; k3 <- 1
> fx1 <- knn.reg(X,as.matrix(x),y=Y,k=k1)
> fx2 <- knn.reg(X,as.matrix(x),y=Y,k=k2)
> fx3 <- knn.reg(X,as.matrix(x),y=Y,k=k3)
> df1 <- data.frame(x=x,y=sin(x))
> df2 <- data.frame(x=x,"K10"=fx1$pred,
  "K100"=fx2$pred,"K1"=fx3$pred)
> df22 <- melt(df2,id.vars=1)
> names(df22)[2:3] <- c("KNN","y")
> ggplot(df22)+aes(x=x,y=y)+
  geom_line(aes(color=KNN,lty=KNN))+
  geom_line(data=df1,aes(x=x,y=y),size=1)

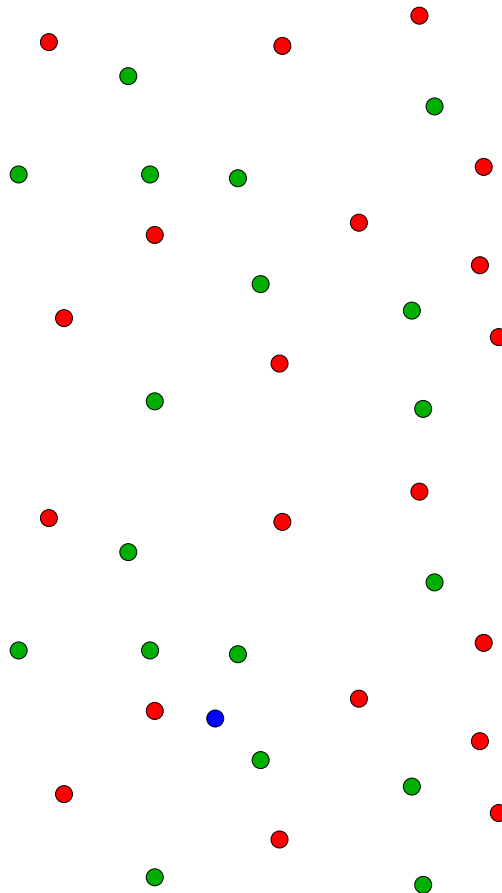
```

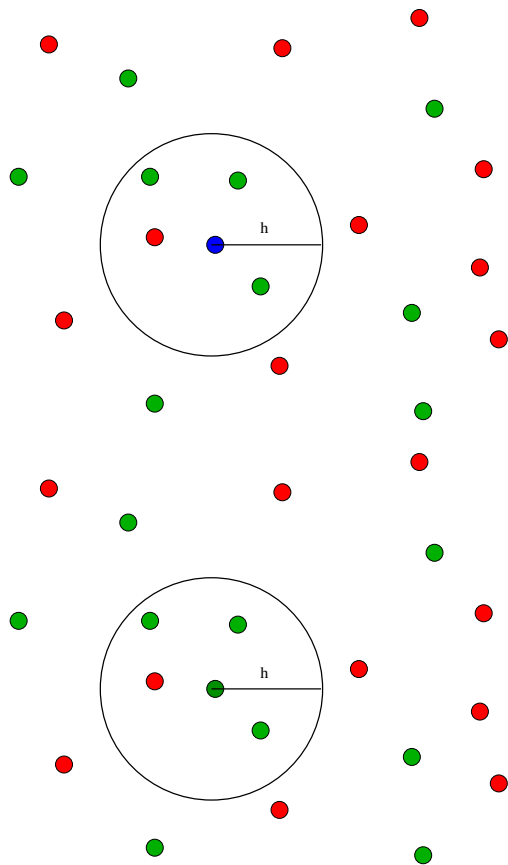


Supervised classification

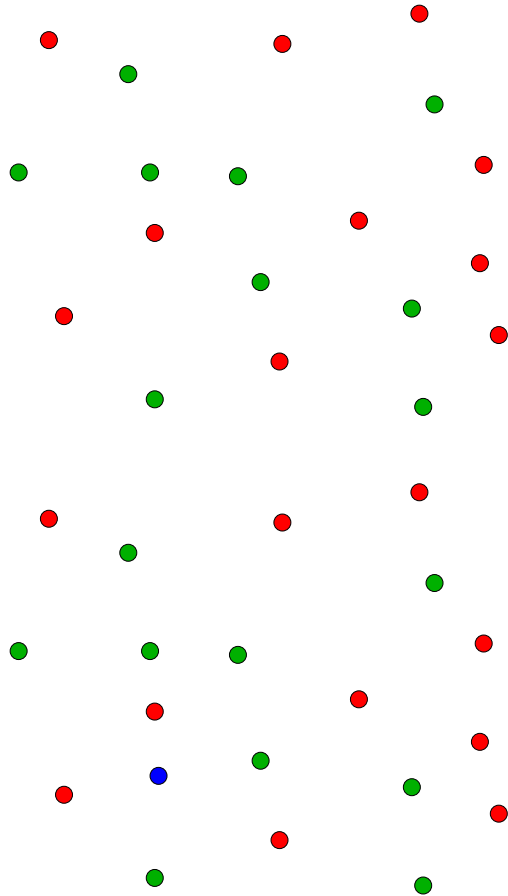
- Kernel and nearest neighbors estimates have been presented in *regression* ($\mathcal{Y} = \mathbb{R}$).
- Approaches are **similar** in supervised classification::
 1. **neighborhoods** are defined in the same way;
 2. (only) *change*: instead of averaging the Y_i in a neighborhood of x , we make a *majority vote*.

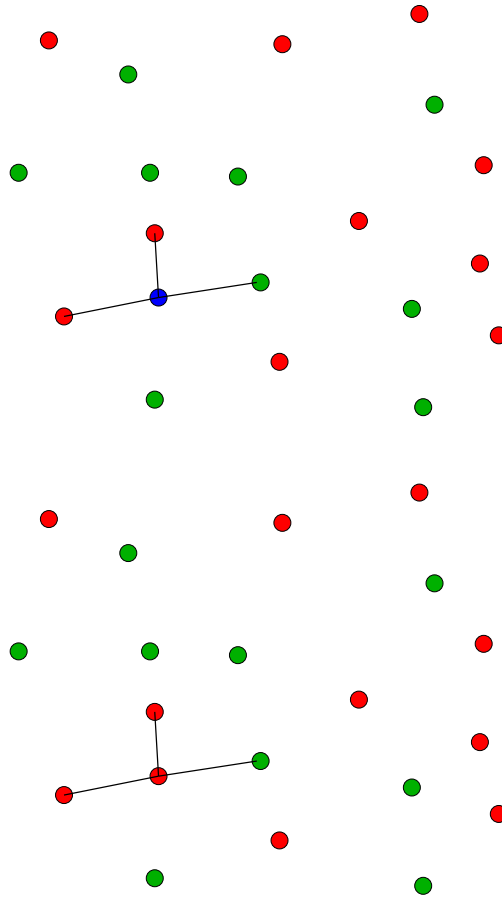
Kernel for supervised classification





k -nn for supervised classification





The k -nn rule

- Let $k \leq n$, the k -nn rule apply a *majority vote* to assess the group of new individuals:

$$\hat{g}_n(x) = MV(Y_i : i \in knn(x)) = \operatorname{argmax}_{k \in \mathcal{Y}} \sum_{i \in knn(x)} \mathbf{1}_{Y_i=k}$$

where $knn(x) = \{i : X_i \text{ is among the } knn \text{ of } x \text{ among } \{X_1, \dots, X_n\}\}$.

Remark

As for regression, the choice of k reveals **crucial** for the performance of the estimate:

- k large**: "steady" estimate, small variance, large bias;
- k small**: "overfitting", large variance, small bias.

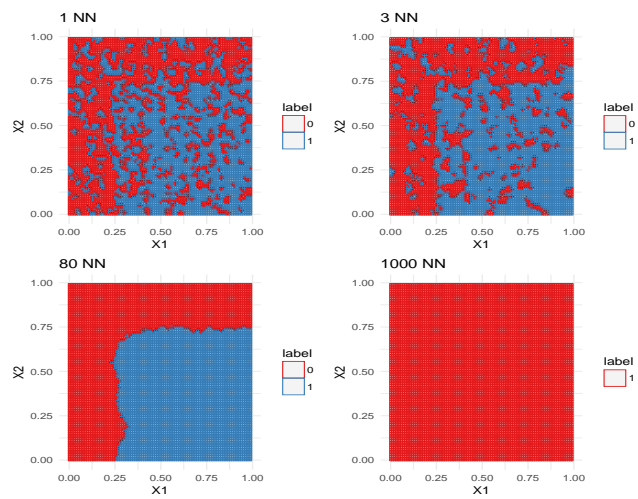
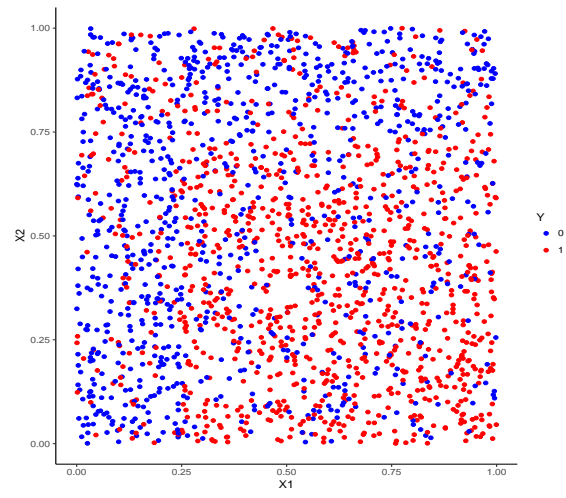
Example

- Goal*: explain a binary variable Y by 2 continuous variables X_1 and X_2 . We have $n = 2000$ observations.

k -nn rules

Conclusion

We clearly visualize how *the choice of k is important*.



Consistency [Györfi et al., 2002]

- For both *regression and supervised classification*, kernel rules and nearest neighbors rules are *universally consistent* (under weak assumptions).

Theorem [Stone, 1977]

If $k \rightarrow \infty$ and $k/n \rightarrow 0$, then the k -nn rule is **universally consistent**.

Theorem [Devroye and Krzyżak, 1989]

If $h \rightarrow 0$ and $nh^d \rightarrow +\infty$, then the kernel rule **universally consistent**.

2.2 The curse of dimensionality

Rate of consistency [Györfi et al., 2002]

Nonparametric methods (always) suffer from *the curse of dimensionality*: as the dimension d increases, we have less and less observations in the neighborhoods of $x \implies$ less and less *accurate* \implies *slower* convergence rate.

Theorem

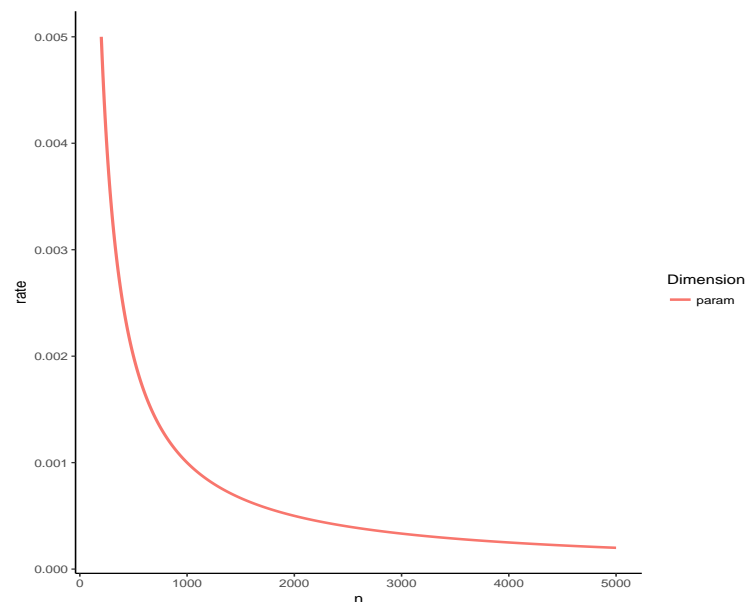
We consider the regression problem (explain Y by X_1, \dots, X_d) and denote by m_n the k -nn estimate. Under technical assumptions, the quadratic risk of m_n satisfies (*see exercise 3-IML0*)

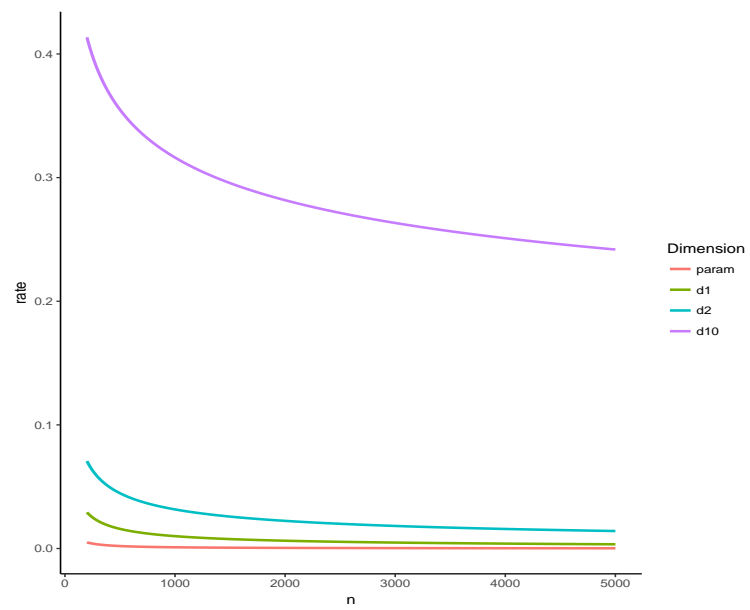
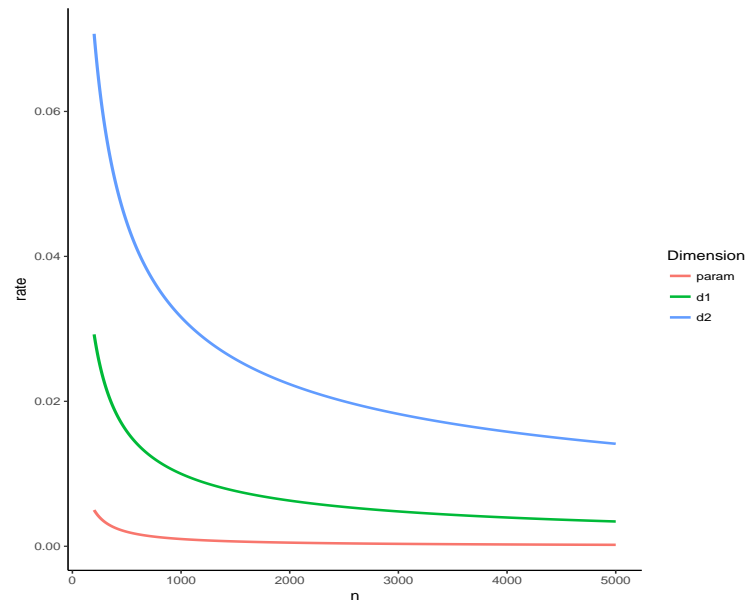
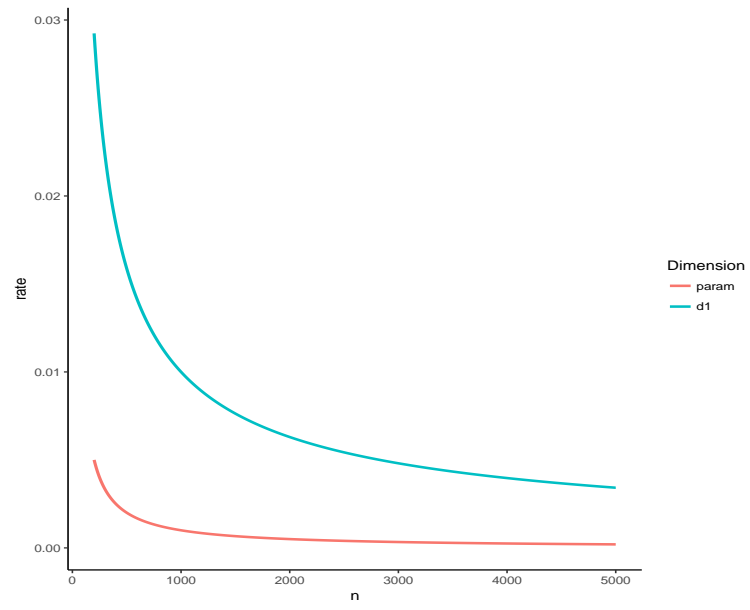
$$\mathcal{R}(m_n) = O\left(n^{-\frac{2}{d+2}}\right).$$

Consequence

- $d = 1$: rate $n^{-2/3}$, $d = 5$: rate $n^{-2/7}$.
- In practice, nonparametric estimates are *not efficient* in **high dimensional spaces**.

Curse of dimensionality (Illustration)





Rules	Parameters
k -nn	k : number of neighbors
kernel	h : bandwidth
trees	$depth$
boosting	$number\ of\ iterations$
...	...

3 Empirical risk minimization

3.1 Setting

Choosing parameters

- Most of the machines depends on *parameters*.
- Selection of these parameters reveals *crucial for the performances of the estimates*.
- *Goal*:
 - define procedures which allow to *automatically* select these parameters;
 - establish theoretical guarantees for these procedures (GB lecture).

ERM strategy

Framework

- \mathcal{F} a collection of machines.
- *Risk* for a machine f : $\mathcal{R}(f) = \mathbf{E}[\ell(Y, f(X))]$.
- *Goal*: select \hat{f} in \mathcal{F} such that

$$\mathcal{R}(\hat{f}) \approx \inf_{f \in \mathcal{F}} \mathcal{R}(f).$$

ERM

- *Estimate the risk* of the machines in \mathcal{F} (validation hold out, cross validation...) $\implies \hat{R}_n(f)$.
- Choose the machine \hat{f} which *minimizes the estimated risk* $\hat{R}_n(f)$.

Selecting k (k -nn rule)

- *Data splitting*:
 - A *learning* or *train* set $\mathcal{D}_m = \{(X_1, Y_1), \dots, (X_m, Y_m)\}$;
 - A *test* set $\mathcal{D}_\ell = \{(X_{m+1}, Y_{m+1}), \dots, (X_n, Y_n)\}$ with $m + \ell = n$.
- *Candidates*: $\mathcal{G}_m = \{g_k, 1 \leq k \leq m\} \rightarrow k$ -nn rules using \mathcal{D}_m .
- *Risk*: $L(g) = \mathbf{P}(g(X) \neq Y)$.

ERM Strategy

Choose \hat{g}_n which minimizes

$$\frac{1}{\ell} \sum_{i=m+1}^n \mathbf{1}_{g_k(X_i) \neq Y_i}.$$

3.2 Caret package

- Classification and regression training.
- This package allows to *select machines* and to *estimate their performances*.
- More than **230 algorithms** are available on caret: <http://topepo.github.io/caret/index.html>
- We just have to specify:
 - the *method* (logistic, *k*-nn, trees, randomForest...)
 - a grid for the parameters to select *parameters* (number of NN...)
 - the *risk* (error probability, AUC, quadratic risk...)
 - how to *estimate the risk* (validation hold out, cross validation, bootstrap...)

Validation hold out

```
> K_cand <- seq(1,500,by=20)
> library(caret)
> ctrl1 <- trainControl(method="LGOCV",number=1,index=list(1:1500))
> KK <- data.frame(k=K_cand)
> e1 <- train(Y~,data=donnees,method="knn",trControl=ctrl1,tuneGrid=KK)
> e1
k-Nearest Neighbors
```

```
2000 samples
 2 predictor
 2 classes: '0', '1'
```

No pre-processing

Resampling: Repeated Train/Test Splits Estimated (1 reps, 75%)

Summary of sample sizes: 1500

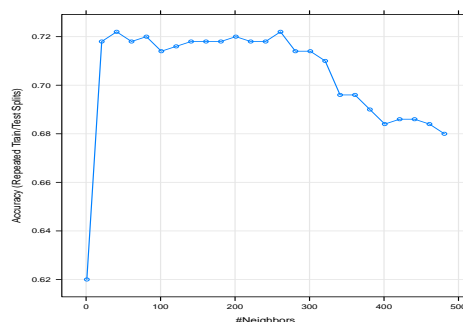
Resampling results across tuning parameters:

k	Accuracy	Kappa
1	0.620	0.2382571
21	0.718	0.4342076
41	0.722	0.4418388
61	0.718	0.4344073
81	0.720	0.4383195
101	0.714	0.4263847
121	0.716	0.4304965
141	0.718	0.4348063
161	0.718	0.4348063
181	0.718	0.4348063
201	0.720	0.4387158
221	0.718	0.4350056
241	0.718	0.4350056
261	0.722	0.4428232
281	0.714	0.4267894
301	0.714	0.4269915
321	0.710	0.4183621
341	0.696	0.3893130
361	0.696	0.3893130
381	0.690	0.3767090
401	0.684	0.3645329
421	0.686	0.3686666
441	0.686	0.3679956
461	0.684	0.3638574
481	0.680	0.3558050

Accuracy was used to select the optimal model using the largest value.

The final value used for the model was k = 261.

```
> plot(e1)
```



Cross validation

```
> library(doMC)
> registerDoMC(cores = 3)
> ctrl2 <- trainControl(method="cv",number=10)
> e2 <- train(Y~.,data=dapp,method="knn",trControl=ctrl2,tuneGrid=KK)
> e2
k-Nearest Neighbors
```

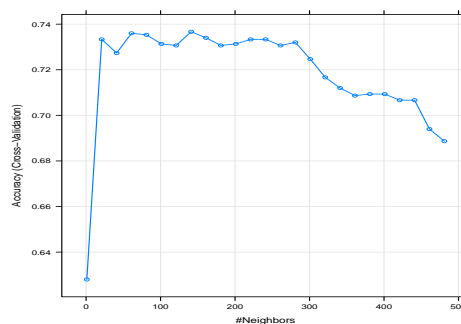
```
1500 samples
  2 predictor
  2 classes: '0', '1'
```

```
No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 1350, 1350, 1350, 1350, 1350, 1350, ...
Resampling results across tuning parameters:
```

k	Accuracy	Kappa
1	0.6280000	0.2519051
21	0.7333333	0.4623213
41	0.7273333	0.4503384
61	0.7360000	0.4682891
81	0.7353333	0.4673827
101	0.7313333	0.4596395
121	0.7306667	0.4584747
141	0.7366667	0.4703653
161	0.7340000	0.4654675
181	0.7306667	0.4585136
201	0.7313333	0.4597224
221	0.7333333	0.4638243
241	0.7333333	0.4637789
261	0.7306667	0.4581189
281	0.7320000	0.4604955
301	0.7246667	0.4452185
321	0.7166667	0.4283226
341	0.7120000	0.4183438
361	0.7086667	0.4109784
381	0.7093333	0.4121146
401	0.7093333	0.4117108
421	0.7066667	0.4057889
441	0.7066667	0.4047529
461	0.6940000	0.3782209
481	0.6886667	0.3662798

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was k = 141.

```
> plot(e2)
```



Repeated cross-validation

```
> ctrl3 <- trainControl(method="repeatedcv",repeats=5,number=10)
> e3 <- train(Y~.,data=dapp,method="knn",trControl=ctrl3,tuneGrid=KK)
> e3
k-Nearest Neighbors
```

```
1500 samples
  2 predictor
  2 classes: '0', '1'
```

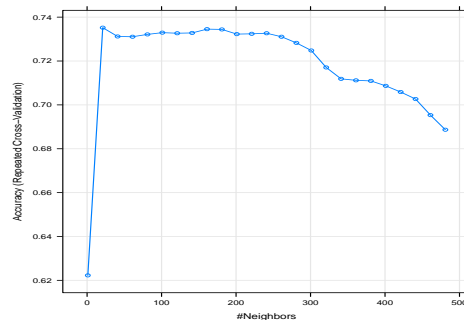
```
No pre-processing
Resampling: Cross-Validated (10 fold, repeated 5 times)
Summary of sample sizes: 1350, 1350, 1350, 1350, 1350, 1350, ...
Resampling results across tuning parameters:
```

k	Accuracy	Kappa
---	----------	-------

1	0.6222667	0.2416680
21	0.7352000	0.4661220
41	0.7312000	0.4580125
61	0.7310667	0.4580882
81	0.7321333	0.4606022
101	0.7329333	0.4626718
121	0.7326667	0.4623496
141	0.7328000	0.4628236
161	0.7345333	0.4663240
181	0.7344000	0.4660110
201	0.7322667	0.4616271
221	0.7324000	0.4619926
241	0.7326667	0.4624912
261	0.7310667	0.4591799
281	0.7282667	0.4530797
301	0.7248000	0.4454653
321	0.7170667	0.4292033
341	0.7118667	0.4181330
361	0.7112000	0.4163210
381	0.7109333	0.4154893
401	0.7086667	0.4104291
421	0.7058667	0.4043432
441	0.7026667	0.3972028
461	0.6953333	0.3813444
481	0.6886667	0.3664347

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was k = 21.

```
> plot(e3)
```



Minimizing AUC

```
> donnees1 <- donnees
> names(donnees1)[3] <- c("Class")
> levels(donnees1$Class) <- c("G0", "G1")
> ctrl11 <- trainControl(method="LGOCV", number=1, index=list(1:1500),
                        classProbs=TRUE, summary=twoClassSummary)
> e4 <- train(Class~., data=donnees1, method="knn", trControl=ctrl11,
              metric="ROC", tuneGrid=KK)
> e4
k-Nearest Neighbors
```

```
2000 samples
2 predictor
2 classes: 'G0', 'G1'
```

No pre-processing
Resampling: Repeated Train/Test Splits Estimated (1 reps, 75%)
Summary of sample sizes: 1500
Resampling results across tuning parameters:

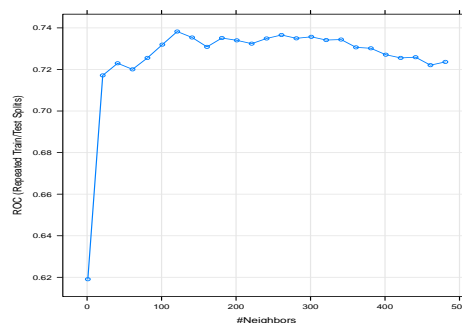
k	ROC	Sens	Spec
1	0.6190866	0.5983264	0.6398467
21	0.7171484	0.6903766	0.7432950
41	0.7229757	0.6861925	0.7547893
61	0.7200500	0.6945607	0.7394636
81	0.7255567	0.6945607	0.7432950
101	0.7319450	0.6903766	0.7356322
121	0.7382452	0.6945607	0.7356322
141	0.7353757	0.7029289	0.7318008
161	0.7308549	0.7029289	0.7318008
181	0.7351272	0.7029289	0.7318008
201	0.7340050	0.7029289	0.7356322
221	0.7324099	0.7071130	0.7279693
241	0.7349028	0.7071130	0.7279693
261	0.7365780	0.7071130	0.7356322

281	0.7349749	0.6987448	0.7279693
301	0.7356963	0.7029289	0.7241379
321	0.7341493	0.6861925	0.7318008
341	0.7343898	0.6527197	0.7356322
361	0.7306385	0.6527197	0.7356322
381	0.7301816	0.6359833	0.7394636
401	0.7270957	0.6276151	0.7356322
421	0.7255487	0.6317992	0.7356322
441	0.7258933	0.6192469	0.7471264
461	0.7220619	0.6150628	0.7471264
481	0.7236330	0.6108787	0.7432950

ROC was used to select the optimal model using the largest value.

The final value used for the model was $k = 121$.

```
> getTrainPerf(e4)
      TrainROC TrainSens TrainSpec method
1 0.7382452 0.6945607 0.7356322      knn
> plot(e4)
```



Summary

- *Parametric*: **strong** assumption but **fast rates** ($1/n$).
- *Non parametric*: **less restrictive** but **slow rates** plus **curse of dimensionality** ($1/n^{2/(d+2)}$).
- *ERM strategy*: select (**automatically**) parameters which minimizes the **estimated risk**.
- *Exercise 5, IML1*.

4 Bibliography

References

Biblio2

- [Breiman et al., 1984] Breiman, L., Friedman, J., Olshen, R., and Stone, C. (1984). *Classification and regression trees*. Wadsworth & Brooks.
- [Cornillon and Matzner-Løber, 2011] Cornillon, P. and Matzner-Løber, E. (2011). *Régression avec R*. Springer.
- [Devroye et al., 1996] Devroye, L., Györfi, L., and Lugosi, G. (1996). *A Probabilistic Theory of Pattern Recognition*. Springer.
- [Devroye and Krzyżak, 1989] Devroye, L. and Krzyżak, A. (1989). An equivalence theorem for l_1 convergence of the kernel regression estimate. *Journal of statistical Planning Inference*, 23:71–82.
- [Fahrmeir and Kaufmann, 1985] Fahrmeir, L. and Kaufmann, H. (1985). Consistency and asymptotic normality of the maximum likelihood estimator in generalized linear models. *The Annals of Statistics*, 13:342–368.
- [Grob, 2003] Grob, J. (2003). *Linear regression*. Springer.

- [Györfi et al., 2002] Györfi, L., Kohler, M., Krzyzak, A., and Harro, W. (2002). *A Distribution-Free Theory of Nonparametric Regression*. Springer.
- [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.
- [Stone, 1977] Stone, C. J. (1977). Consistent nonparametric regression. *Annals of Statistics*, 5:595–645.

Part III

Linear model: variable selection and regularization

Framework

- $(X_1, Y_1), \dots, (X_n, Y_n)$ i.i.d. observations with the same distribution as (X, Y) which takes values in $\mathcal{X} \times \mathcal{Y}$;
- In this *part*, we assume $\mathcal{X} = \mathbb{R}^d$ and $\mathcal{Y} = \mathbb{R}$ or $\{-1, 1\}$.

Linear and logistic models

1. If $\mathcal{Y} = \mathbb{R}$,

$$m(x) = \mathbf{E}[Y|X = x] = \beta_0 + \beta_1 x_1 + \dots + \beta_d x_d = x^t \beta.$$

2. If $\mathcal{Y} = \{-1, 1\}$,

$$\text{logit } p(x) = \beta_0 + \beta_1 x_1 + \dots + \beta_d x_d = x^t \beta$$

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

Some limits

- 2 *drawbacks* in some situations:
 1. **prediction accuracy**: LS and ML estimates can have *large variance* (especially when d is large) and thus poor prediction accuracy.
 2. **interpretation**: when d is large, we don't know what are the *most important variables*.

Goals

- Since we have *more and more data*, these drawbacks are occurring more and more often.
- We need to develop new *automatic procedures* to *select important variables*.

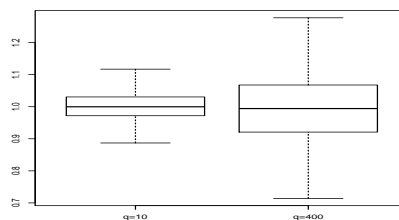
An example

- We generate observations $(x_i, y_i), i = 1, \dots, 500$ according to

$$Y = 1X_1 + 0X_2 + \dots + 0X_{q+1} + \varepsilon$$

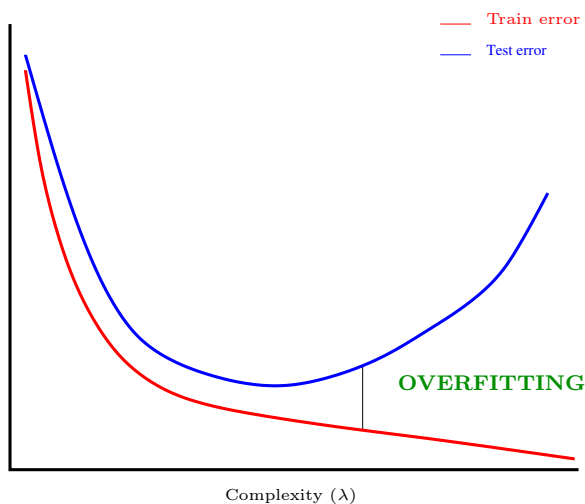
where $X_2, X_{q+1}, \dots, \varepsilon$ are i.i.d. with law $\mathcal{N}(0, 1)$.

- We compute the *LS estimator* of β_1 for 1000 replications. We draw boxplot of these estimators for $q = 10$ and $q = 400$.



Conclusion

Large **variance** (thus **loss of accuracy**) when the number of *unnecessary variables* increases.



Size of the model

Conclusion

The **size** of the model governs the bias/variance trade-off.

1 Subset selection

Best subset selection

- $(X_1, Y_1), \dots, (X_n, Y_n)$ i.i.d. with the same law as (X, Y) which takes values in $\mathbb{R}^d \times \mathbb{R}$;
- d input variables $\implies 2^d$ candidate models.

The idea

1. Fit the 2^d models;
2. Choose the one which optimizes a given *criterion*.

Algorithm : best subset selection

1. for $k = 0, \dots, d$:
 - (a) Fit the $\binom{d}{k}$ linear models with k variables;
 - (b) Choose the model with the higher R^2 . Denote \mathcal{M}_k this model.
2. Select, among $\mathcal{M}_0, \mathcal{M}_1, \dots, \mathcal{M}_d$, the best model according to a given **criterion**.

Some criteria

- **AIC**: Akaike Information Criterion

$$-2\mathcal{L}_n(\hat{\beta}) + 2d.$$

- **BIC**: Bayesian Information Criterion

$$-2\mathcal{L}_n(\hat{\beta}) + \log(n)d.$$

- **Adjusted R^2** :

$$R_a^2 = 1 - \frac{n-1}{n-d+1}(1-R^2) \quad \text{where} \quad R^2 = \frac{SSR}{SST} = \frac{\|\hat{\mathbf{Y}} - \bar{\mathbf{Y}}\mathbf{1}\|^2}{\|\mathbf{Y} - \bar{\mathbf{Y}}\mathbf{1}\|^2}.$$

- **Mallo's C_p** :

$$C_p = \frac{1}{n} \left(\sum_{i=1}^n (Y_i - \hat{Y}_i)^2 + 2d\hat{\sigma}^2 \right).$$

R user

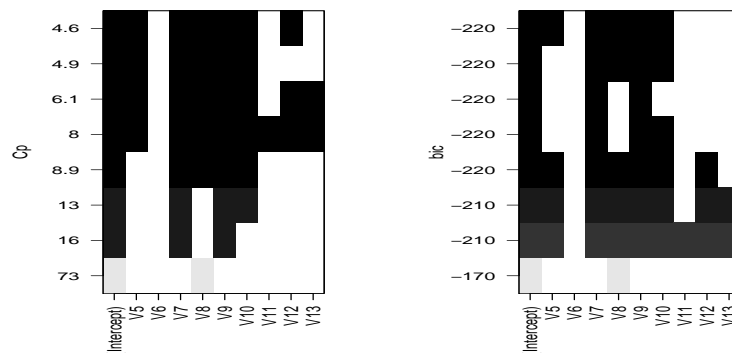
- `regsubsets` from **leaps** package allows to make *best subset selection*.

```
> library(leaps)
> reg.fit <- regsubsets(V4~V5+V6+V7+V8+V9+V10+V11+V12+V13,data=0zone)
> summary(reg.fit)
1 subsets of each size up to 8
Selection Algorithm: exhaustive
      V5  V6  V7  V8  V9  V10 V11 V12 V13
1 ( 1 ) " " " " " " "*" " " " " " " " "
2 ( 1 ) " " " " "*" " " " "*" " " " " " "
3 ( 1 ) " " " " "*" " " " "*" "*" " " " "
4 ( 1 ) " " " " "*" "*" "*" "*" " " " "
5 ( 1 ) "*" " " " "*" "*" "*" "*" " " " "
6 ( 1 ) "*" " " " "*" "*" "*" "*" " " "*"
7 ( 1 ) "*" " " " "*" "*" "*" "*" " " "*"
8 ( 1 ) "*" " " " "*" "*" "*" "*" "*" "*"

```

```
> plot(reg.fit,scale="Cp")
> plot(reg.fit,scale="bic")

```



- Mallows's C_p selects:

$$Y = \beta_0 + \beta_1 V_5 + \beta_2 V_7 + \beta_3 V_8 + \beta_4 V_9 + \beta_5 V_{10} + \beta_6 V_{12} + \varepsilon.$$

- BIC selects:

$$Y = \beta_0 + \beta_1 V_5 + \beta_2 V_7 + \beta_3 V_8 + \beta_4 V_9 + \beta_5 V_{10} + \varepsilon.$$

Stepwise selection

- BSS considers *all models* (advantage).
- *Drawback*: it becomes *infeasible* (too long computational time) when d is large ($d \geq 40$).
- When d is large, we can seek a *good path* through all possible subsets.
- *Stepwise selection procedures* define recursive models by adding or deleting one variable at each step.

Forward stepwise selection

1. Let \mathcal{M}_0 the **null model** (only the intercept);
2. for $k = 0, \dots, d - 1$:
 - (a) Define the $d - k$ models by adding one variable in \mathcal{M}_k ;
 - (b) Choose, among those $d - k$ models, the one which maximizes the R^2 . Denote \mathcal{M}_{k+1} this model.
3. Select, among $\mathcal{M}_0, \dots, \mathcal{M}_d$, the best model according to a given **criterion**.

Backward stepwise selection

1. Let \mathcal{M}_d the **full model** (d variables);
2. For $k = d, \dots, 1$:
 - (a) Define the k models by deleting one variable in \mathcal{M}_k ;
 - (b) Choose, among those k models, the one which maximizes R^2 . Denote \mathcal{M}_{k-1} this model.
3. Select, among $\mathcal{M}_0, \dots, \mathcal{M}_d$, the best model according to a given **criterion**.

R user

- We just have to add the argument **method="forward"** or **method="backward"** in *regsubsets* to make *subset selection*.

```
> reg.fit.for <- regsubsets(V4~V5+V6+V7+V8+V9+V10+V11+V12+V13,data=Ozone,
  method="forward")
> reg.fit.back <- regsubsets(V4~V5+V6+V7+V8+V9+V10+V11+V12+V13,data=Ozone,
  method="backward")
```

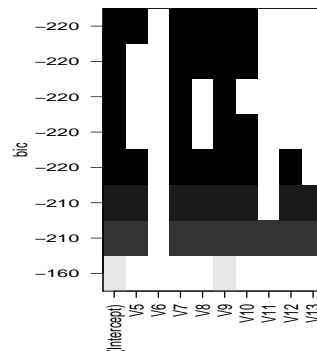
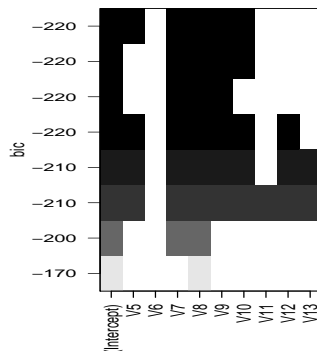
```
> summary(reg.fit.for)
```

```
      V5 V6 V7 V8 V9 V10 V11 V12 V13
1 ( 1 ) " " " " " " " " " " " " " "
2 ( 1 ) " " " " " " " " " " " " " "
3 ( 1 ) " " " " " " " " " " " " " "
4 ( 1 ) " " " " " " " " " " " " " "
5 ( 1 ) " " " " " " " " " " " " " "
6 ( 1 ) " " " " " " " " " " " " " "
7 ( 1 ) " " " " " " " " " " " " " "
8 ( 1 ) " " " " " " " " " " " " " "
```

```
> summary(reg.fit.back)
```

```
      V5 V6 V7 V8 V9 V10 V11 V12 V13
1 ( 1 ) " " " " " " " " " " " " " "
2 ( 1 ) " " " " " " " " " " " " " "
3 ( 1 ) " " " " " " " " " " " " " "
4 ( 1 ) " " " " " " " " " " " " " "
5 ( 1 ) " " " " " " " " " " " " " "
6 ( 1 ) " " " " " " " " " " " " " "
7 ( 1 ) " " " " " " " " " " " " " "
8 ( 1 ) " " " " " " " " " " " " " "
```

```
> plot(reg.fit.for,scale="bic")
> plot(reg.fit.back,scale="bic")
```



Remark

For this example, forward and backward selection provide the same model (it's not always the case).

Binary classification

- *Best subset* and *stepwise* selection have been proposed for **regression** ($\mathcal{Y} = \mathbb{R}$).
- These approaches are exactly the same for **binary classification** ($\mathcal{Y} = \{-1, 1\}$).
- With R, we can use:
 - *bestglm* function from the *bestglm* package for **best subset selection**.
 - *step* function for **stepwise selection**.
- *Exercise 1-2, IML2*.

2 Penalized regression

- For large values of d , least square estimates in the linear model

$$Y = \beta_1 X_1 + \dots + \beta_d X_d + \varepsilon$$

often exhibits **high variance (overfitting)**.

Penalized regression: the idea

- **Constraint** the values of the LS estimates to **reduce the variance** (even if we **increase the bias**).
- **How?** By imposing a **constraint** on the size of the coefficients:

$$\hat{\beta}^{pen} = \underset{\beta}{\operatorname{argmin}} \sum_{i=1}^n \left(y_i - \sum_{j=1}^d x_{ij} \beta_j \right)^2$$

subject to $\|\beta\|_? \leq t$.

Questions

- Which *norm* for the constraint?
- How should we *select* t ?
 - t small \implies **strong** constraint ($\hat{\beta}_j \approx 0$) ;
 - t large \implies **small** constraint ($\hat{\beta}_j \approx \hat{\beta}_{j,LS}$).

2.1 Ridge regression

- *Ridge regression* shrinks the regression coefficients by constraining the **euclidean norm** of the parameters.

Definition

1. Ridge estimates $\hat{\beta}^R$ minimize

$$\sum_{i=1}^n \left(y_i - \beta_0 - \sum_{j=1}^d x_{ij} \beta_j \right)^2 \quad \text{subject to} \quad \sum_{j=1}^d \beta_j^2 \leq t \quad (2)$$

2. or *equivalently* by imposing a penalty on the size of the coefficients

$$\hat{\beta}^R = \underset{\beta}{\operatorname{argmin}} \left\{ \sum_{i=1}^n \left(y_i - \beta_0 - \sum_{j=1}^d x_{ij} \beta_j \right)^2 + \lambda \sum_{j=1}^d \beta_j^2 \right\}. \quad (3)$$

Some remarks

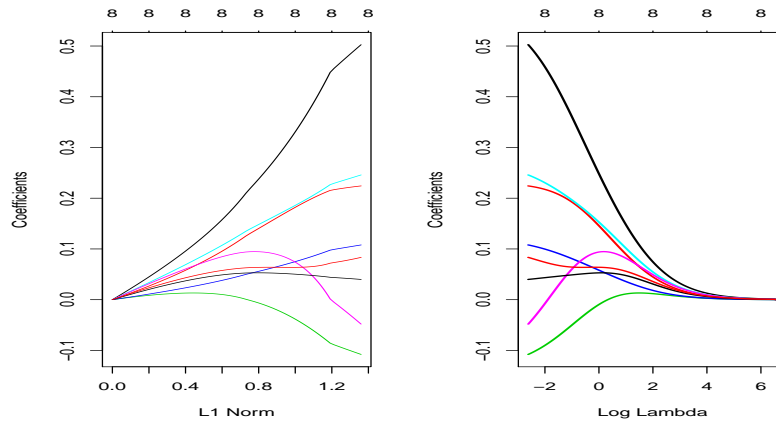
- (2) are (3) *the same* in the sense that there is a **one-to-one correspondence** between t and λ .
- Ridge estimate *depends* on t (or λ) : $\hat{\beta}^R = \hat{\beta}^R(t) = \hat{\beta}^R(\lambda)$.
- Input variables are generally *standardized* to make the variables at the same **scale** (it is automatic in classical softwares).

An example

- *The problem:* explain the level of **prostate specific antigen** by a number (8) of clinical measures.
- $n = 100$ data available at <https://web.stanford.edu/~hastie/ElemStatLearn/>
- Package *glmnet* allows to make ridge regression on R.

UseR

```
> reg.ridge <- glmnet(prostate.data2[,2:9],prostate.data2[,10],alpha=0)
> plot(reg.ridge,label=TRUE)
> plot(reg.ridge,xvar="lambda",label=TRUE,lwd=2)
```



Some properties of ridge estimates

Proposition 2.1. 1. Solution of (3) is given by

$$\hat{\beta}^R = \hat{\beta}^R(\lambda) = (\mathbb{X}^t \mathbb{X} + \lambda \mathbb{I})^{-1} \mathbb{X}^t \mathbb{Y}.$$

2. It follows that

$$\text{bias}(\hat{\beta}^R) = -\lambda(\mathbb{X}^t \mathbb{X} + \lambda \mathbb{I})^{-1} \beta$$

and

$$\mathbf{V}(\hat{\beta}^R) = \sigma^2 (\mathbb{X}^t \mathbb{X} + \lambda \mathbb{I})^{-1} \mathbb{X}^t \mathbb{X} (\mathbb{X}^t \mathbb{X} + \lambda \mathbb{I})^{-1}.$$

Remarks

- For $\lambda = 0$, we obtain **LS** estimates.
- $\lambda \nearrow \implies \text{bias} \nearrow$ and variance \searrow and conversely as $\lambda \searrow$.

Choice of λ

- This choice of λ reveals *crucial* for the performance: if $\lambda \approx 0$ then $\hat{\beta}^R \approx \hat{\beta}^{MCO}$, if λ "large" then $\hat{\beta}^R \approx 0$.
- The procedure to select λ is usual:
 1. **Estimation of a criterion** for a grid of λ ;
 2. We choose the value of λ which **minimizes** the estimated criterion.
- **Example:** *cv.glmnet* selects the value of λ which minimizes the *quadratic risk*:

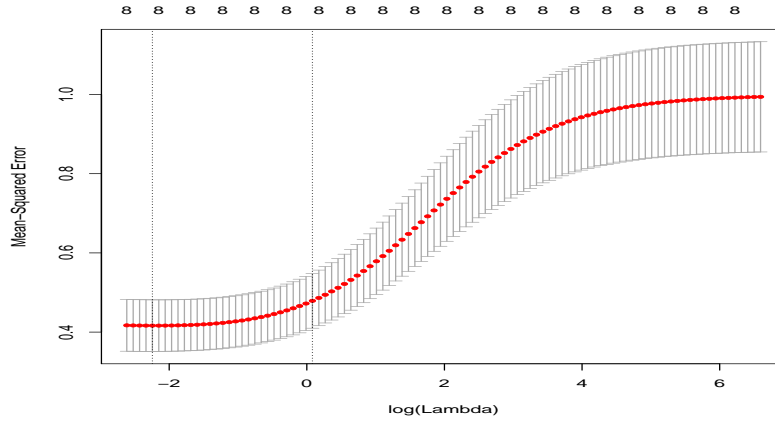
$$\mathbf{E}[(Y - X^t \hat{\beta}^R(\lambda))^2]$$

estimated by *cross validation*.

```

> reg.cvridge <- cv.glmnet(prostate.data2[,2:9],prostate.data2[,10],alpha=0)
> bestlam <- reg.cvridge$lambda.min
> bestlam
[1] 0.1060069
> plot(reg.cvridge)

```



2.2 Lasso regression

- *Lasso regression* shrinks the regression coefficients by constraining the L_1 norm of the parameters.

Definition [Tibshirani, 1996]

1. *Lasso estimates* $\hat{\beta}^L$ minimize

$$\sum_{i=1}^n \left(Y_i - \beta_0 - \sum_{j=1}^d X_{ij} \beta_j \right)^2 \quad \text{subject to} \quad \sum_{j=1}^d |\beta_j| \leq t \quad (4)$$

2. or *equivalently* by imposing a penalty on the size of the coefficients

$$\hat{\beta}^L = \underset{\beta}{\operatorname{argmin}} \left\{ \sum_{i=1}^n \left(Y_i - \beta_0 - \sum_{j=1}^d X_{ij} \beta_j \right)^2 + \lambda \sum_{j=1}^d |\beta_j| \right\}. \quad (5)$$

Comparison Ridge-Lasso

- If \mathbb{X} is an *orthonormal input matrix*, we have an *explicit solution* for ridge and lasso.

Proposition

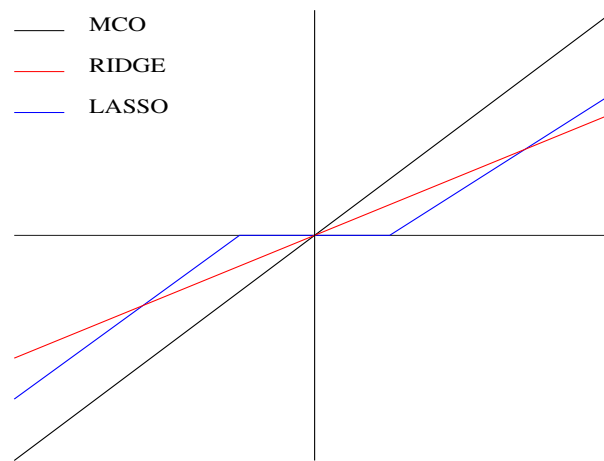
If \mathbb{X} is orthonormal, then

$$\hat{\beta}_j^R = \frac{\hat{\beta}_j}{1 + \lambda} \quad \text{and} \quad \hat{\beta}_j^L = \begin{cases} \operatorname{sign}(\hat{\beta}_j)(|\hat{\beta}_j| - \lambda) & \text{if } |\hat{\beta}_j| \geq \lambda \\ 0 & \text{otherwise.} \end{cases}$$

where $\hat{\beta}_j$ is the LS of β_j .

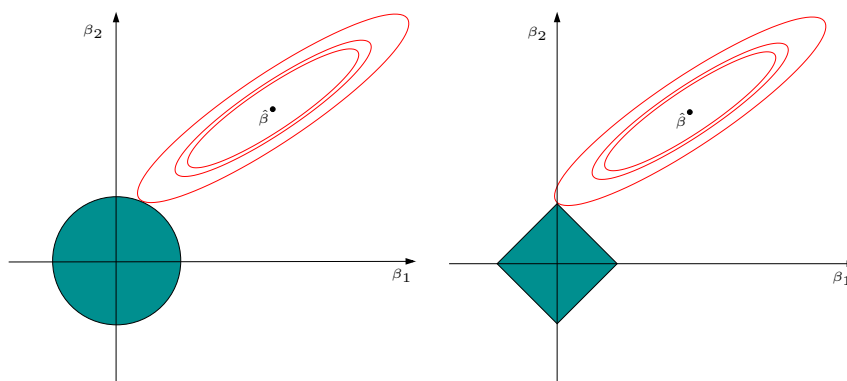
Comments

- **Ridge** does a **proportional shrinkage**;
- **Lasso** *translates* each coefficient by a factor λ , **truncating at 0** (when it is small).



Conclusion

Lasso put *small coefficients to 0* \implies variables with small coefficients are *excluded* from the model.



Relationship between ridge and lasso

Both methods find the first point where the *elliptical contours hit the constraint region*:

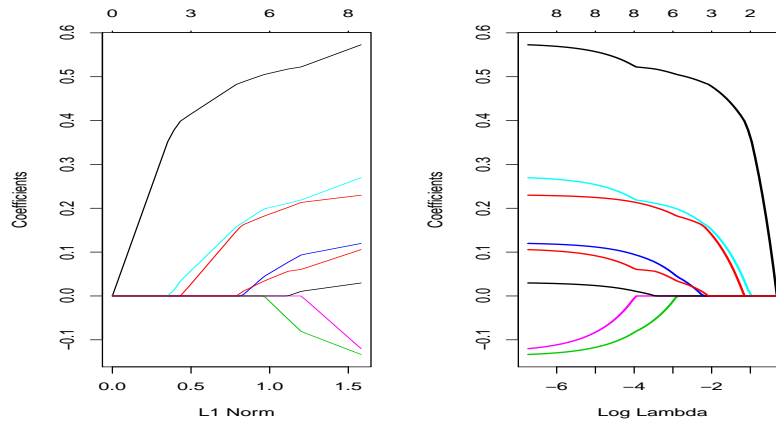
1. L_2 for *ridge* and L_1 norm for *lasso*.
2. The diamonds (L_1) has corner \implies the constraint region is often *hit at a corner*.

Some remarks

- As for ridge:
 - input variables X_1, \dots, X_d are generally *standardized* before the analysis.
 - $\lambda \nearrow \implies$ bias \nearrow and variance \searrow and reciprocally as $\lambda \searrow$.
 - Choice of λ reveals *crucial* (minimization of an estimated criterion).
- **BUT**, unlike ridge: $\lambda \nearrow \implies$ *some estimated parameters equal 0* for lasso ([Bühlmann and van de Geer, 2011]).

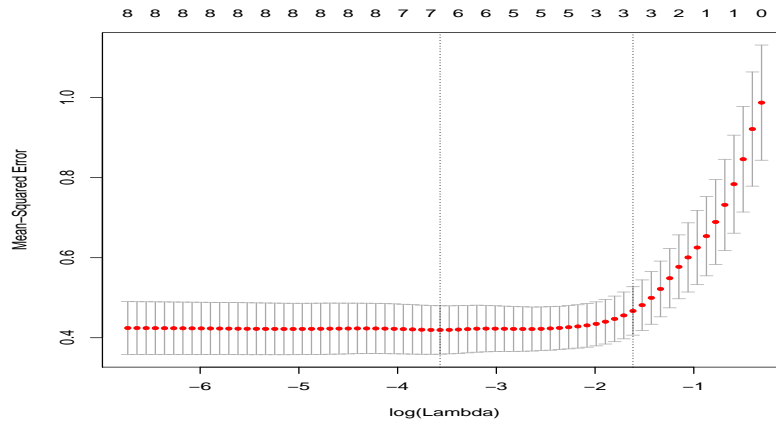
UseR

```
> reg.lasso <- glmnet(prostate.data2[,2:9],prostate.data2[,10],alpha=1)
> plot(reg.lasso,label=TRUE)
> plot(reg.lasso,xvar="lambda",label=TRUE,lwd=2)
```



Choice of λ

```
> reg.cvlasso <- cv.glmnet(prostate.data2[,2:9],prostate.data2[,10],alpha=1)
> bestlam <- reg.cvlasso$lambda.min
> bestlam
[1] 0.02815637
> plot(reg.cvlasso)
```



2.3 Supervised classification

Binary classification

- *Ridge and lasso* have been presented for regression.
- It is not difficult to adjust these methods to the *logistic model* $\mathcal{Y} = \{-1, 1\}$.
- *Penalty terms are the same.*
- **Only change:** least square criterion is replaced by *likelihood*.

Lasso and Ridge for logistic regression

Definition

Let $\tilde{y}_i = (y_i + 1)/2$ ($\tilde{y}_i = 0$ or 1).

- *Ridge estimates* for logistic regression are defined by

$$\hat{\beta}^R = \underset{\beta}{\operatorname{argmin}} \left\{ - \sum_{i=1}^n (\tilde{y}_i x_i^t \beta - \log(1 + \exp(x_i^t \beta))) + \lambda \sum_{j=1}^d \beta_j^2 \right\}.$$

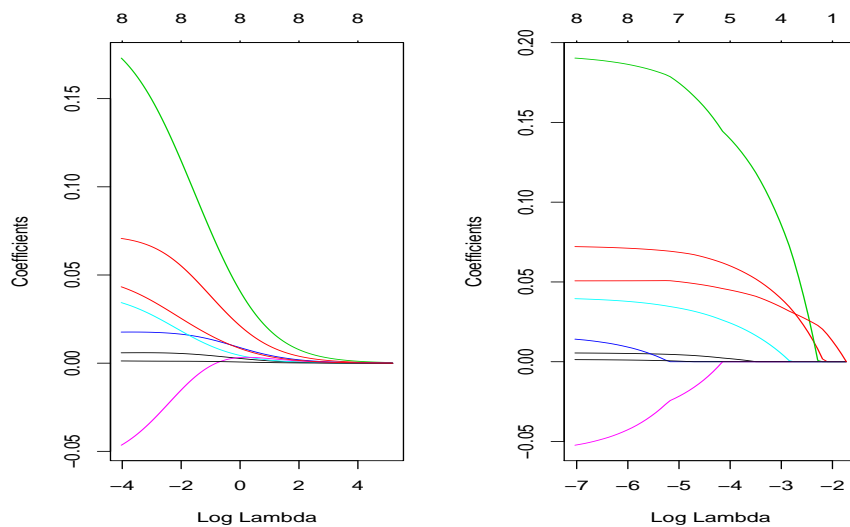
- *Lasso estimates* for logistic regression are defined by

$$\hat{\beta}^L = \underset{\beta}{\operatorname{argmin}} \left\{ - \sum_{i=1}^n (\tilde{y}_i x_i^t \beta - \log(1 + \exp(x_i^t \beta))) + \lambda \sum_{j=1}^d |\beta_j| \right\}.$$

UseR

- To make ridge or lasso for logistic regression, we just have to add `family=binomial` in `glmnet` function.
- *It is the only change* (coefficient paths, choice of λ are the same...).

```
> colnames(donnees)
[1] "sbp"      "tobacco"  "ldl"      "adiposity" "typea"    "obesity"
[7] "alcohol"  "age"      "chd"
> log.ridge <- glmnet(donnees[,1:8], donnees[,9], family="binomial", alpha=0)
> log.lasso <- glmnet(donnees[,1:8], donnees[,9], family="binomial", alpha=1)
> plot(log.ridge, xvar="lambda")
> plot(log.lasso, xvar="lambda")
```



Elastic net

- [Zou and Hastie, 2005] have proposed to *combine ridge and lasso* with the following penalty term (called *elastic net penalty*)

$$\lambda \sum_{j=1}^d ((1 - \alpha) \beta_j^2 + \alpha |\beta_j|)$$

where $\alpha \in [0, 1]$.

- α measures the trade-off *ridge/lasso* :
 - $\alpha = 1 \implies$ Lasso;
 - $\alpha = 0 \implies$ Ridge.
 - This parameter corresponds (obviously) to the *alpha* parameter in `glmnet` function.
- *Advantage*: more flexible since elastic net includes ridge and lasso.
- *Drawback*: we have to select both α and λ (you can use `caret` to do that).

Summary

- LASSO and ridge regressions allow to make *efficient* linear models when the **classical linear model is defective**:
 - *high correlations* between inputs;
 - *high dimension* (large number of inputs).
- When the linear model is efficient, we don't need to use these methods.
- *Exercise 3-4, IML2.*

3 Bibliography

Références

Biblio3

- [Bühlmann and van de Geer, 2011] Bühlmann, P. and van de Geer, S. (2011). *Statistics for high-dimensional data*. Springer.
- [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.
- [Tibshirani, 1996] Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society, Series B*, 58:267–288.
- [Zou and Hastie, 2005] Zou, H. and Hastie, T. (2005). Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society, Series B*, 67:301–320.

Part IV

Trees

Presentation

- Tree algorithms are statistical learning algorithms for both *regression and supervised classification*.
- *Popular* method, not (too) difficult to understand, *visualization* tool.
- Tree algorithms are not generally the most performant algorithms... *but* a lot of efficient algorithms are defined from trees (random forest, gradient tree boosting...).
- There are *different ways* to build trees.
- We focus on the *CART* algorithm [Breiman et al., 1984] which is the most widely used algorithm to define trees.

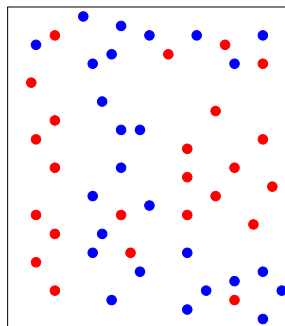
1 Binary trees

Notations

- *The problem*: **explain** output Y by p inputs X_1, \dots, X_p .
- Y might be categorical (binary or not) or continuous and X_1, \dots, X_p *categorical or continuous*.
- For simplicity (to make figures), we first assume that Y is binary (-1 ou 1) and that $p = 2$ (2 inputs X_1 and X_2 continuous).

Data visualization

- n observations $(X_1, Y_1), \dots, (X_n, Y_n)$ where $X_i \in \mathbb{R}^2$ and $Y_i \in \{-1, 1\}$.

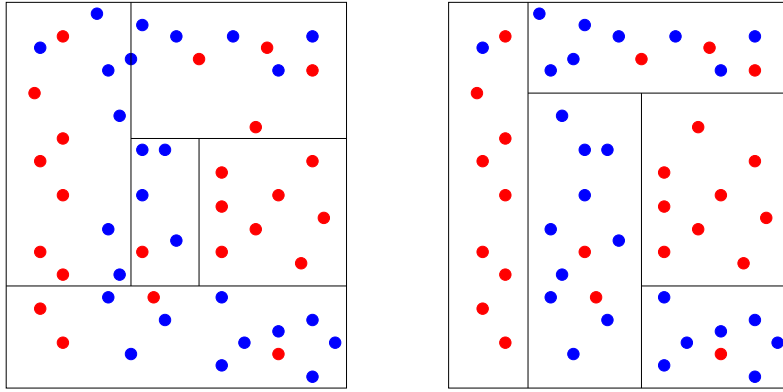


Tree partitions

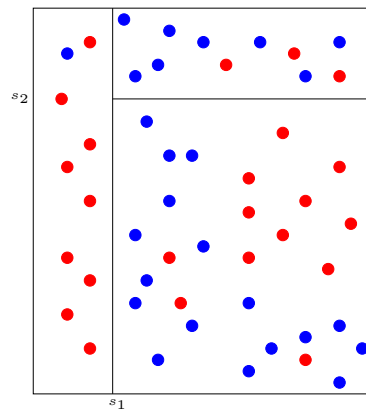
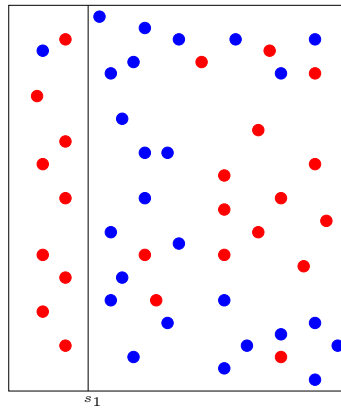
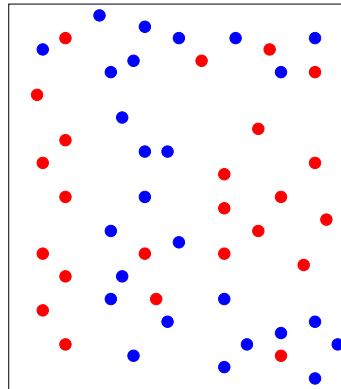
Find a **partition** of the feature space into a set of rectangles which **divides** points according to their color.

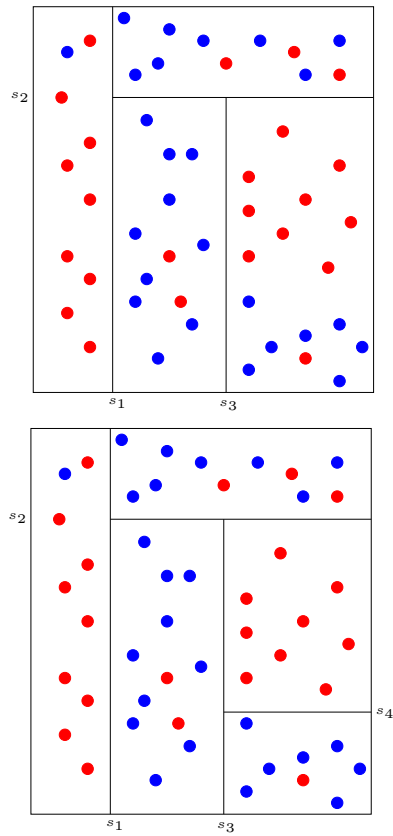
Binary partitions

- *CART algorithm* restricts attention to **recursive** binary partitions.
- 2 examples:

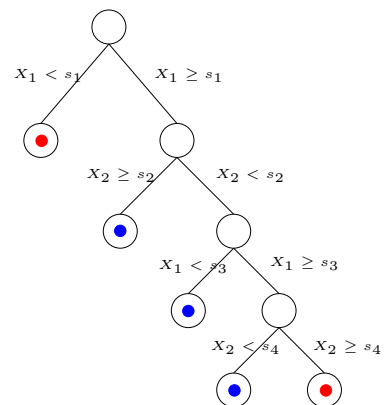
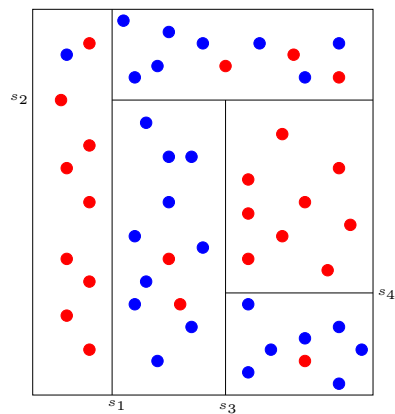
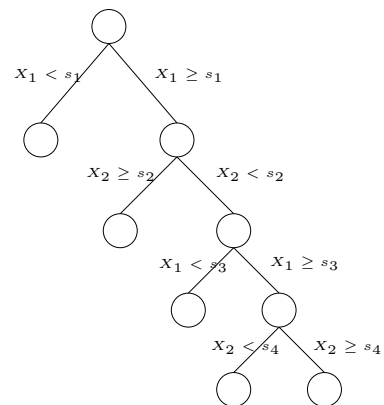
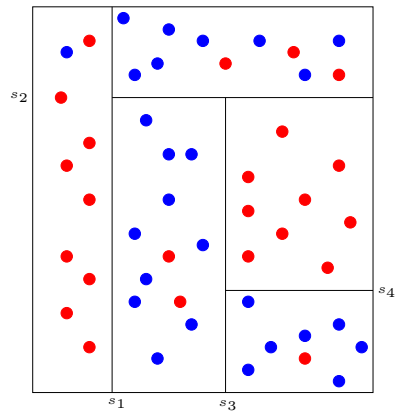


- At each step, the method splits the data into two regions according to a *split variable* and a *split point*.





A tree partition



Classification rule

At the end, we do a **majority vote** in each cell of the partition (in each rectangle).

Definitions

Definitions

- Each elements of the partition are called *terminal nodes*.
- \mathbb{R}^p (the first node) is the *root node*.
- Each split (each question) defines two child nodes, the *left and right child nodes*.

Question

- Tree process is **recursive**: we just have to know how to split a node.
- How to define a good split (or find a good question)?

2 Choice of the split

Question

How to choose a split?

- At each step, we *have to find* (j, s) which split a node \mathcal{N} into two children nodes

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

- (j, s) is selected by **minimizing a criterion** which measures the *impurity* of the two children nodes.

Impurity

- Impurity of a node should be
 1. **small** when the node is homogeneous: values of Y are *closed* to each other in the node.
 2. **large** when the node is heterogeneous: values of Y are *different* from each other in the node.

The idea

For a given impurity measure \mathcal{I} , we choose the split (j, s) which **minimizes**

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

where $\mathbf{P}(\mathcal{N}_k)$ stands for the proportion of observations in $\mathcal{N}_k, k = 1, 2$

2.1 Regression

- In regression (Y continuous), we usually use the *variance* to measure the impurity in the node

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

where $\bar{Y}_{\mathcal{N}}$ is the mean of Y_i in \mathcal{N} .

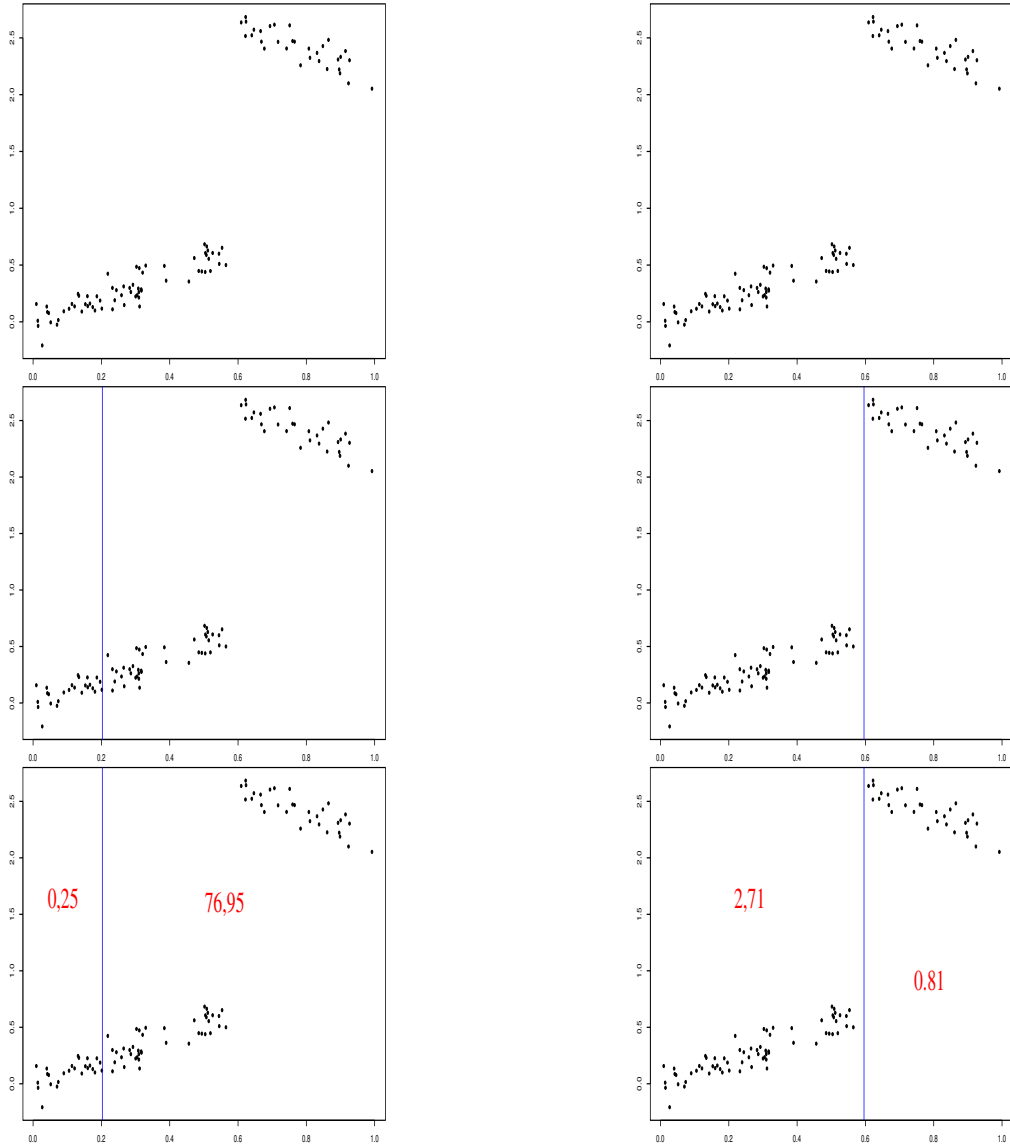
Split for regression

At each step, we choose (j, s) which minimizes

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

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

Example



Conclusion

We choose the **right** split.

2.2 Supervised classification

- $Y_i, i = 1, \dots, n$ take values in $\{1, \dots, K\}$.
- We search an impurity function \mathcal{I} such $\mathcal{I}(\mathcal{N})$ is
 - *small* if one label appears in *majority* in \mathcal{N} , if we can clearly differentiate one label from the other;
 - *large* otherwise.

Definition

Impurity of \mathcal{N} is defined by

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

where

- $p_j(\mathcal{N})$ stands for the proportion of class j in \mathcal{N} .
- f is a concave function $[0, 1] \rightarrow \mathbb{R}^+$ such that $f(0) = f(1) = 0$.

Examples of functions f

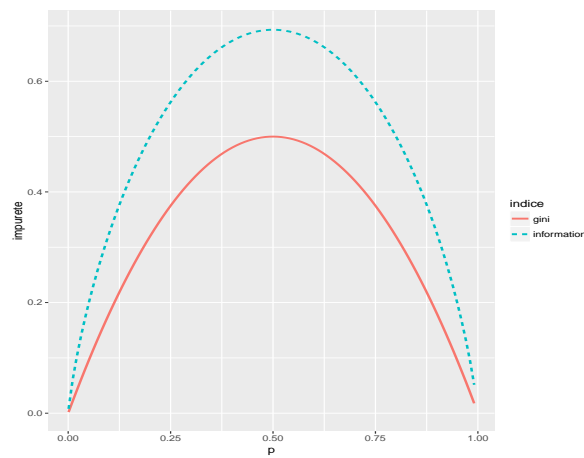
- If \mathcal{N} is *pur*, we expect that $\mathcal{I}(\mathcal{N}) = 0 \implies$ that's why $f(0) = f(1) = 0$.
- The two classical impurity functions are
 1. *Gini*: $f(p) = p(1 - p)$;
 2. *Information*: $f(p) = -p \log(p)$.

Binary case

We have

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

where p stands for the proportion of 1 (or -1) in \mathcal{N} .



Split for supervised classification

- Recall that for a given node \mathcal{N} and (j, s) , the two child nodes are defined by

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

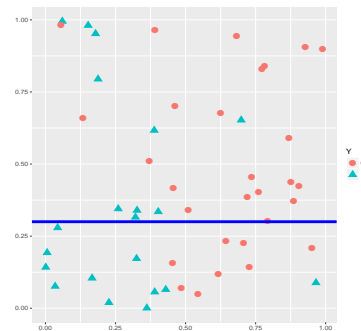
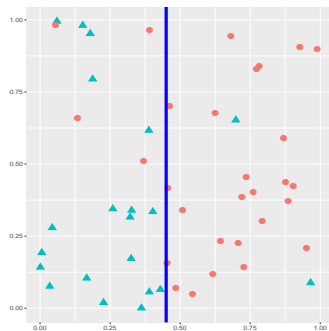
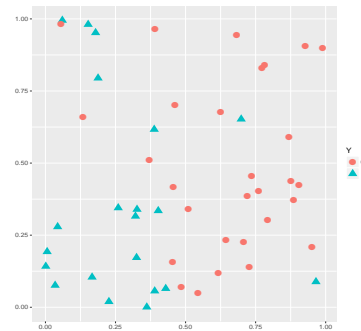
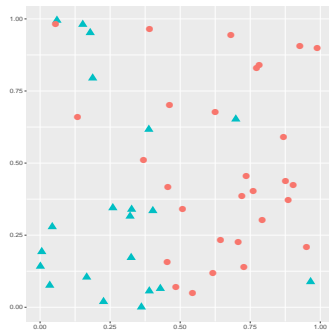
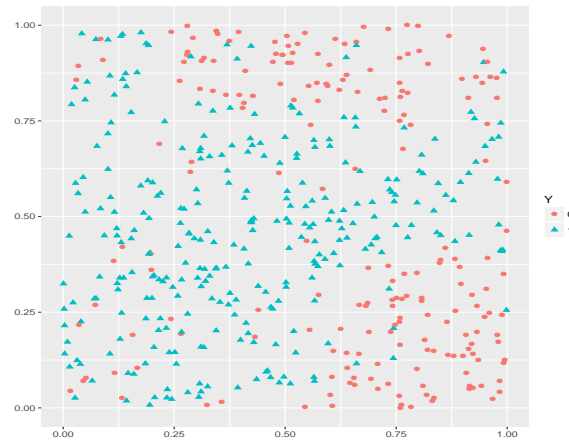
Choice of (j, s)

For a given impurity measure \mathcal{I} , we choose (j, s) which **minimizes**:

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

Example

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



	$\mathcal{I}(\mathcal{N}_1)$	$\mathcal{I}(\mathcal{N}_2)$	Crit.
Left	0.287	0.137	0.2061
Right	0.488	0.437	0.4562

Conclusion

We select the **left** split. (*Exercise 1,2,3-IML3.*)

3 Pruning a tree

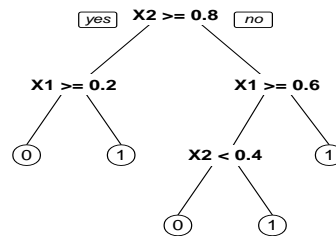
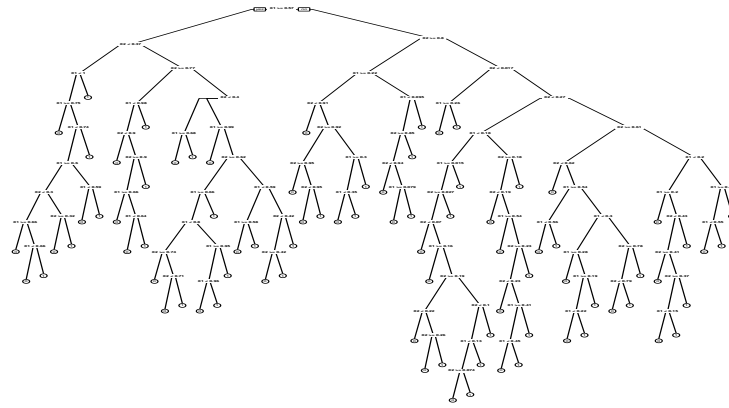
Questions

- How to select an *efficient tree*?
- Do we choose the *maximum or deeper* tree? (split the nodes until one observation by node).
- Grow a *large tree* and then **prune** this tree (select a subtree of this large tree)?

An example for binary classification

Optimal tree?

Intuitively, we are tempted to choose *5 or 6 terminal nodes*.



"Deeper" tree

```

> library(rpart)
> library(rpart.plot)
> tree1 <- rpart(Y~.,data=my_data,cp=0.0001,minsplit=2)
> prp(tree1)

```

A smaller tree

```

> tree2 <- rpart(Y~.,data=my_data)
> prp(tree2)

```

Comparison

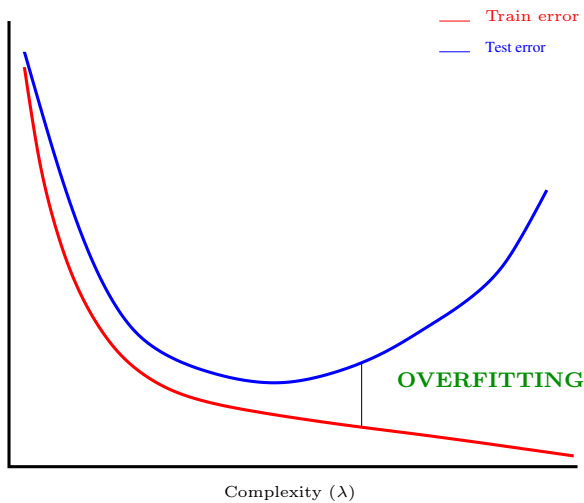
- We estimate the *misclassification error* of these two trees on a **test set**.

```

> prev1 <- predict(tree1,newdata=dtest,type="class")
> prev2 <- predict(tree2,newdata=dtest,type="class")
> round(mean(prev1!=dtest$Y),3)
[1] 0.157
> round(mean(prev2!=dtest$Y),3)
[1] 0.115

```

Conclusion



- Performance is *not always improved* by the *size* of the tree.
- Tree size is a tuning parameter which governs the *model's complexity*. We have to **select** this parameter.

Overfitting

Remark

Complexity is governed by the *depth* (or *size*) of the tree.

Bias and variance

Depth controls the tradeoff bias/variance :

1. **Small tree** \implies steady (robust) tree \implies **small variance**... but... *large bias*.
2. **Large tree** \implies unsteady tree \implies **small bias**... but... *large variance (overfitting)*.

Pruning [Breiman et al., 1984]

Instead of stopping the splitting process, we

1. grow a *large tree* (very deep tree) \mathcal{T}_{max} ;
2. then select a sequence of *nested subtrees*:

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

3. finally *select one subtree* in this sequence.

Nested trees

Construction of the sequence

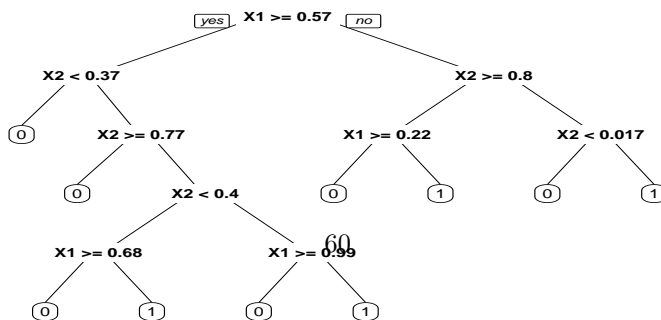
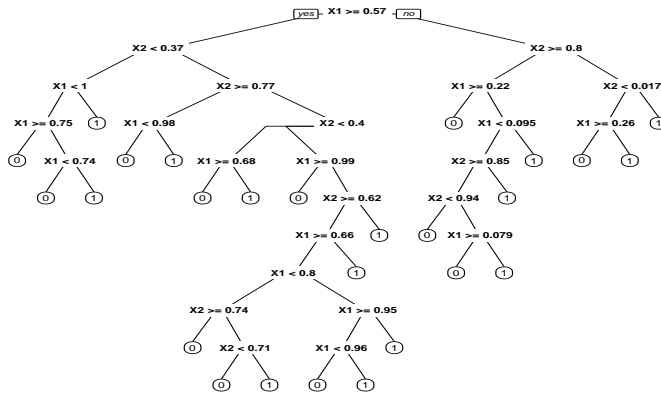
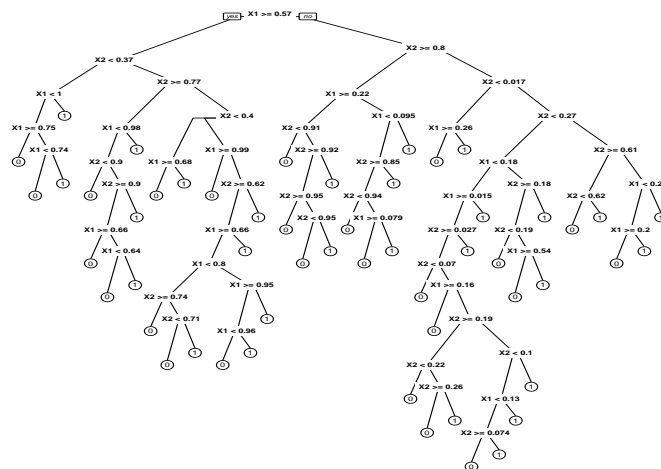
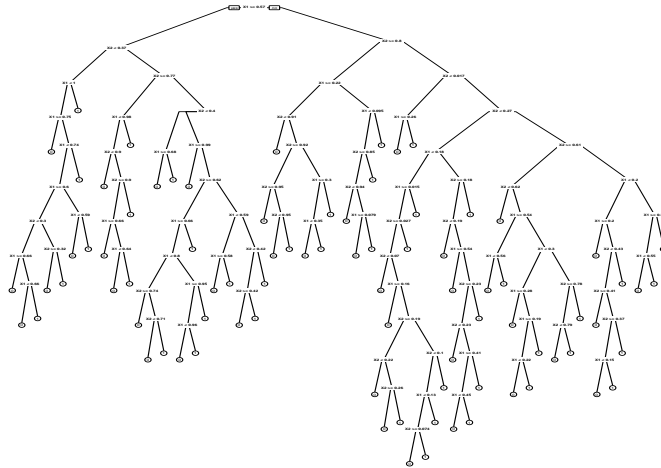
- Let T be a tree with $|T|$ terminal nodes $\mathcal{N}_1, \dots, \mathcal{N}_{|T|}$.
- Define $R(\mathcal{N})$ the risk (error) in node \mathcal{N} :

– *Regression*:

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

– *Classification*:

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



Definition

For $\alpha > 0$,

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

is the *cost complexity criterion* of T .

The idea

- $C_\alpha(T)$ measures both the **fitting** and the **complexity** of the tree.
- The **idea** is to find the subtree T_α which minimizes $C_\alpha(T)$ for a safe choice of α .

Remark

- $\alpha = 0 \implies T_\alpha = T_0 = T_{max}$.
- $\alpha = +\infty \implies T_\alpha = T_{+\infty} = \text{tree without split}$.
- α is called the *complexity parameter*.

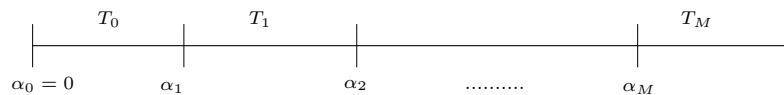
Theorem [Breiman et al., 1984]

There exists a *finite* sequence $\alpha_0 = 0 < \alpha_1 < \dots < \alpha_M$ with $M < |T_{max}|$ and a sequence of *nested trees*

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

such that $\forall \alpha \in [\alpha_m, \alpha_{m+1}[$

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



Important consequence

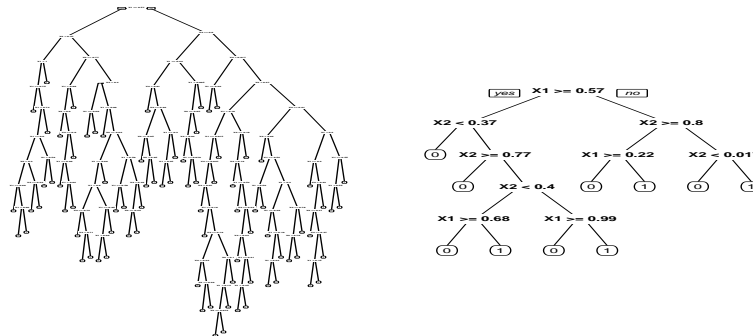
- We now are faced with a **finite** sequence of nested trees.
- We have to choose one tree in this sequence (or **one value of α**).

Example

```
> printcp(tree)
Classification tree:
rpart(formula = Y ~ ., data = my_data, 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(tree,cp=0.005)
> prp(tree)
> prp(arbre1)
```



Remark

We have to **select** one tree in the sequence

$$T_{max} = T_0 \supset T_1 \supset \dots \supset T_M.$$

The final tree

Risk estimation

We choose the **final tree** by minimizing a risk $\mathcal{R}(T_m) = \mathbb{E}[\ell(Y, T_m(X))]$ (as usual). For instance,

1. **quadratic risk** $\mathbb{E}[(Y - T_m(X))^2]$ in *regression* ;
2. **misclassification error** $\mathbb{P}(Y \neq T_m(X))$ in *supervised classification*.

This risk is unknown and is generally estimated by **cross validation**.

Select the optimal tree

The approach consists in

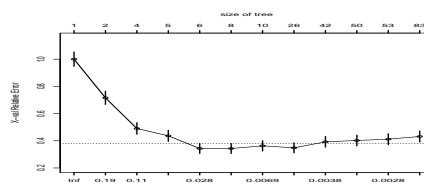
1. *estimating the risk* for each subtree.
2. selecting the subtree which *minimizes the estimated risk*.

- Estimations of $\mathcal{R}(m)$ are in the column *xerror* of the function **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

- We can look at the estimated error for each subtree with **plotcp**

```
> plotcp(tree3)
```

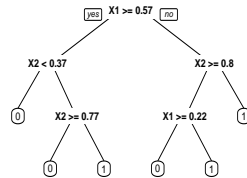


Conclusion

We choose the tree with **5 splits**.

Visualisation of the final tree

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



Classification rule and score for a tree

- Final tree \mathcal{T} consists of a partition of \mathbb{R}^p into $|\mathcal{T}|$ terminal nodes $\mathcal{N}_1, \dots, \mathcal{N}_{|\mathcal{T}|}$.

- Classification rule:**

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

where $\mathcal{N}(x)$ stands for the terminal node which contains x .

- Score:**

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

Predict function

- `predict` function (or `predict.rpart`) allows to *estimate* the **label** or the **score** of a new observation:

```
> 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
```

Conclusion

- "Simple" method for both *regression* and *supervised classification*.
- We can *interpret* the model (plot the tree) if the tree is not too large.
- One drawback:** due to the recursive process, the algorithm is *not stable*, affected by small perturbations of the sample.
- This drawback will become an advantage for *bootstrap aggregating* \implies **random forest**.
- Exercise 4-IML3.*

4 Bibliography

References

Biblio4

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

1	2	3	4	5	6	7	8	9	10
---	---	---	---	---	---	---	---	---	----

Part V

Bagging and random forests

1 Bagging

- *Bagging* is a set of algorithms introduced by Léo Breiman [Breiman, 1996].
- *Bagging* comes from **Bootstrap Aggregating**.

The idea

- Instead of fitting one "sophisticated" machine, fit a lot of **simple** machines and *aggregate* them.
- **Example:**

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

where $\hat{m}_1(x), \dots, \hat{m}_B(x)$ are simple machines.

Questions

- How to define the *simple* machines?
- Do we choose **efficient** simple machines? **Not efficient** (large bias, large variance) machines?
- How many machines?
- **One constraint:** we want to fit simple machines in a similar way (only trees for instance).
- *Problem:* if you run the same algorithm on the same dataset $(X_1, Y_1), \dots, (X_n, Y_n)$, all simple machines will be the same and

$$\hat{m}(x) = \frac{1}{B} \sum_{k=1}^B \hat{m}_k(x) = \hat{m}_1(x)$$

\Rightarrow aggregation is *useless*.

- *Solution:* run the same algorithm on **different** datasets.

Bootstrap sample

- We have at hand *one* dataset $\mathcal{D}_n = (X_1, Y_1), \dots, (X_n, Y_n)$.
- We will not create or invent data!

Bootstrap

- Define new datasets by *randomly* draw dataset with **replacement** from the training data.

Bootstrap: example

- The sample:
- *Bootstrap* samples:
- We finally *aggregate*:

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

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
	\vdots								\vdots	
7	10	3	4	9	10	10	8	6	1	m_B

Bagging algorithm

- Estimates m_k are not fitted on the original dataset $\mathcal{D}_n = (X_1, Y_1), \dots, (X_n, Y_n)$ but on *bootstrap samples*.

Bagging

Inputs:

- a "simple machine" (a tree, 1NN rule...)
- B a positive integer.

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

- Draw a bootstrap sample from \mathcal{D}_n .
- Fit the simple machine on this bootstrap sample: $m_k(x)$.

Output: the aggregate estimate $\hat{m}_B(x) = \frac{1}{B} \sum_{k=1}^B m_k(x)$.

How to choose B?

- 2 parameters have to be chosen: *the number of iterations B and the simple machine*.
- From the Law of Large Numbers, we can prove that

$$\lim_{B \rightarrow +\infty} \hat{m}_B(x) = \lim_{B \rightarrow +\infty} \frac{1}{B} \sum_{k=1}^B m_k(x) = \bar{m}(x, \mathcal{D}_n) \quad a.s | \mathcal{D}_n.$$

- As B increases, \hat{m}_B stabilizes.

Important conclusion

- B is **not** an **important** parameter, we have to choose it as large as possible (often 500).
- Bagging is **random** but it is less random when B is large.

Some properties

Bias and variance

For regression, we have $\mathbf{E}[\hat{m}_B(x)] = \mathbf{E}[m_k(x)]$, $\forall k = 1, \dots, B$ and

$$\mathbf{V}[\hat{m}_B(x)] \approx |\rho(x)| \mathbf{V}[m_k(x)]$$

where $\rho(x) = \text{corr}(m_k(x), m_{k'}(x))$ for $k \neq k'$.

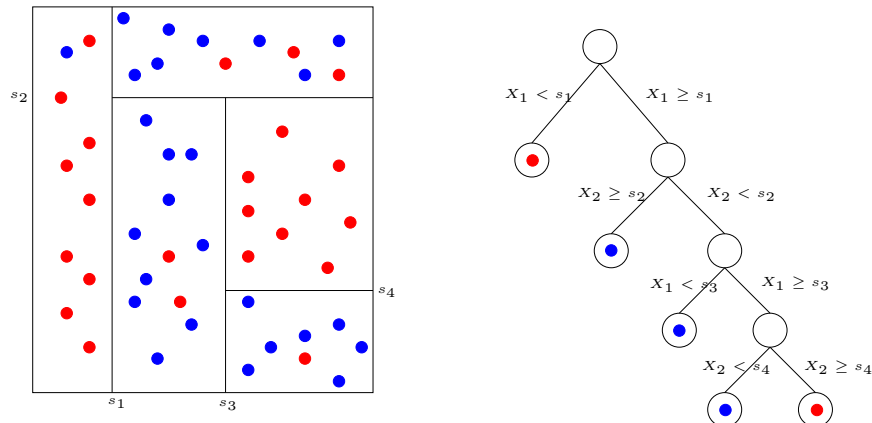
Remarks

- Bias is *not affected* by the bagging process.
- Variance of the bagging estimate reduces when *correlation* between the simple machines **decreases**.
- Consequence:** we need simple machines *sensitive to small disturbances of the data*.
- Trees* are known to satisfy this property (**drawback becomes an advantage**...).

2 Random forests

2.1 The algorithm

Tree (reminder)



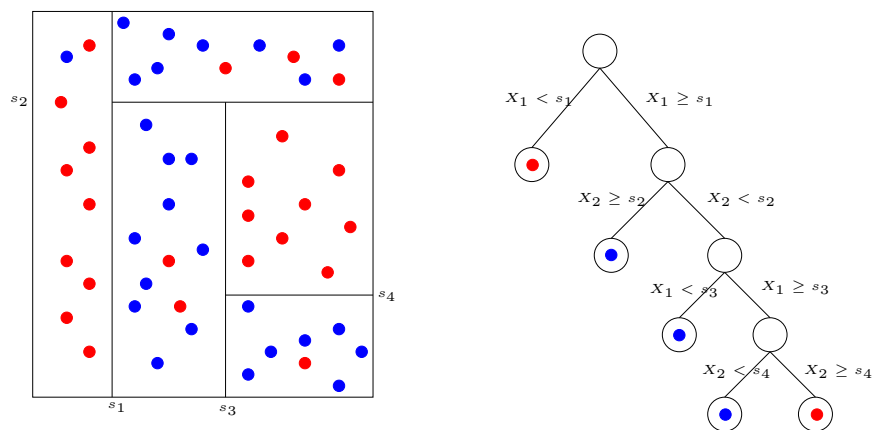
Important parameter: depth

- **small**: bias \nearrow , variance \searrow
- **large**: bias \searrow , variance \nearrow
- A random forest = a collection of trees.
- These algorithms have been studied by *Léo Breiman* (2000).
- References

<http://www.stat.berkeley.edu/~breiman/RandomForests/>

Robin Genuer's phd thesis [[Genuer, 2010](#)].

- Trees are fitted as for the CART process (no pruning) with only *one small variation*.



Trees for the forest

- At each step, the best split is selected among $m_{try} \leq d$ inputs *randomly* chosen among the d inputs.
- *Goal*: try to **reduce correlations** between the trees, to make the trees more different from each other.

Random forest algorithm

Inputs:

- B size of the forest;
- $mtry \in \{1, \dots, d\}$ number of candidate inputs for each split.

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

1. Draw a *bootstrap* sample from \mathcal{D}_n ;
2. Fit a tree according to the CART process, each split is chosen among $mtry$ variables randomly chosen among the d input variables. Denote by $T_k(x)$ the tree.

Output: the random forest $\hat{T}_B(x) = \frac{1}{B} \sum_{k=1}^B T_k(x)$.

Comments

- The algorithm is for both *regression* and *binary classification*:
 1. for regression, the RF estimates $m^*(x) = \mathbf{E}[Y|X = x]$;
 2. for binary classification, the RF estimates $S^*(x) = \mathbf{P}(Y = 1|X = x)$.
- Simple algorithm. On R , you can use *randomForest* function from the **randomForest** package or the *ranger* function from the **ranger** package.
- Estimate known to be *efficient* for complex data and *robust* (wrt to the choice of its parameter).

Choice of the parameter

- B : large.

Remind

Bagging decreases the variance:

$$\mathbf{V}[\hat{T}_B(x)] \approx |\rho(x)| \mathbf{V}[T_k(x)].$$

Consequence

- Bias is not improved by the *bagging process*, it is recommended to use trees with *small bias* and **large variance**.
- Trees for forest are *deep trees* with a small number of observations in each terminal node.
- By default *randomForest* fit trees with (only) **5** observations in terminal nodes for regression and **1** for supervised classification.

Choice of $mtry$

- This parameter (slightly) governs the bias/variance trade-off of the forest.

Conclusion

- We can look at the performances of the forest for *many values of $mtry$* .
- By default $mtry = d/3$ for regression and \sqrt{d} for supervised classification.

Application on the spam dataset

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

```
> library(randomForest)
> forest1 <- randomForest(type~.,data=spam)
> forest1

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
```

2.2 OOB error

Random forest performance

- As for other machine learning algorithms, we need *criteria* to measure *performances* of a **random forest**.
- **Examples:**
 - Quadratic risk $\mathbf{E}[(Y - \hat{T}_B(X))^2]$ for regression;
 - Misclassification error $\mathbf{P}(Y \neq \hat{T}_B(X))$ for supervised classification.
- These criteria can be estimated by *validation hold out* or *cross validation*.
- *Bootstrap step* in bagging algorithms proposes another way to estimate these criteria: **OOB (Out Of Bag)**.

Out Of Bag error

- For each (X_i, Y_i) , construct its *random forest* predictor by averaging only those trees corresponding to bootstrap samples in which (X_i, Y_i) does *not* appear:

$$\hat{Y}_i = \frac{1}{|\mathcal{I}_B|} \sum_{k \in \mathcal{I}_B} T_k(X_i)$$

where \mathcal{I}_B is the set of trees such that (X_i, Y_i) is **Out Of Bag**.

Out Of Bag estimates

- OOB **quadratic risk**: $\frac{1}{n} \sum_{i=1}^n (\hat{Y}_i - Y_i)^2$.
- OOB **misclassification error**: $\frac{1}{n} \sum_{i=1}^n \mathbf{1}_{\hat{Y}_i \neq Y_i}$.

Example

- (X_1, Y_1) does not appear in bootstrap samples 2, 3 and 5, thus

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

- We do the same for *all the observations* $\implies \hat{Y}_2, \dots, \hat{Y}_n$.
- We obtain the *OOB quadratic risk*:

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

Example

- Spam dataset with $mtry = 1$:

```
> forest2 <- randomForest(Y~.,data=spam,mtry=1)
> forest2

Call:
randomForest(formula = Y ~ ., data = dapp, 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

OOB misclassification error: 8.04% for $mtry = 1$ and 5.26% for $mtry = 7$.

2.3 Variable importance

- *Single trees* are highly interpretable.
- Linear combinations of trees (random forests) *lose* this important features.
- There exists a *score* which measures *importance* of each inputs.
- As for OOB error, this score is based on the fact for some observations *does not appear in bootstrap samples*.
- Let OOB_k denotes the *OOB* sample of the k -th tree.
- Let E_{OOB_k} the quadratic error of the k -th tree measured on OOB_k :

$$E_{OOB_k} = \frac{1}{|OOB_k|} \sum_{i \in OOB_k} (T_k(X_i) - Y_i)^2.$$

- *Permute* (randomly) the values of input j in $OOB_k \Rightarrow OOB_k^j$ and compute the quadratic error on this dataset:

$$E_{OOB_k}^j = \frac{1}{|OOB_k^j|} \sum_{i \in OOB_k^j} (T_k(X_i^j) - Y_i)^2,$$

Definition

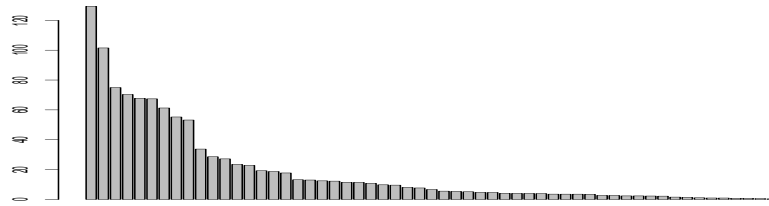
The **variable importance score** for the j variable is defined by

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

Example

- It is easy to obtain variable importance score with `randomForest`

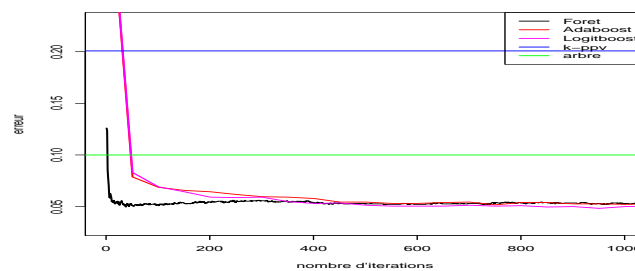
```
> imp <- importance(forest1)
> imp1 <- sort(imp,decreasing=TRUE)
> ord <- order(imp,decreasing=TRUE)
> ord
[1] 52 53 55  7 56 16 21 25 57  5 24 19 26 23 46 27 11  8 50 12 37  3 18  6 45
[26] 17 10  2 28 42 49 35  1 36 39 13 54  9 30 33 22 51 29 14 43 44 31 20 48 15
[51] 40  4 41 34 32 38 47
> barplot(imp1,beside=TRUE)
```



Comparison - spam dataset

- We make a comparison between some statistical learning algorithms on the spam dataset.
- To do that, we split the data into a
 - a *training set* of size 2300 to **fit and calibrate** the models;
 - a *test set* of size 2301 to **estimate misclassification error** of each model

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



Method	M. error
Random Forest	0.050
Adaboost	0.052
Logitboost	0.048
k -NN	0.200
Tree	0.100

- *Exercise 5-IML3*

3 Bibliography

References

Biblio5

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

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