



CentraleSupélec

# Statistique et apprentissage

Chargés de cours (ordre alphabétique) :

Julien Bect, Gilles Faÿ, Ziad Kobeissi, Laurent Le Brusquet,  
Vincent Lescarret, Arshak Minasyan, Arthur Tenenhaus<sup>†</sup> & Xujia Zhu

<sup>†</sup> Coordinateur du cours

1/81

Cours 7/9

Classification : régression logistique.  
Quelques modèles pour l'apprentissage supervisé

## Objectifs du cours 7

- ▶ Réaliser une classification par régression logistique
- ▶ Définir des mesures de performances pour la classification
- ▶ Savoir prédire à l'aide d'arbres de décision
- ▶ Savoir prédire à l'aide de réseaux de neurones

2/81

## Plan du cours

- 1 – Généralités sur la classification
- 2 – Régression logistique [classification]
- 3 – Arbres de décision [régression + classification]
- 4 – Réseaux de neurones [régression + classification]
- 5 – Exercices types
- 6 – Annexes

3/81

## Plan du cours

- 1 – Généralités sur la classification
  - 1.1 – Introduction
  - 1.2 – Fonctions de perte et fonctions de classification optimales
  - 1.3 – Mesures de performances
- 2 – Régression logistique [classification]
- 3 – Arbres de décision [régression + classification]
- 4 – Réseaux de neurones [régression + classification]
- 5 – Exercices types
- 6 – Annexes

# Plan du cours

## 1 – Généralités sur la classification

### 1.1 – Introduction

### 1.2 – Fonctions de perte et fonctions de classification optimales

### 1.3 – Mesures de performances

## 2 – Régression logistique [classification]

## 3 – Arbres de décision [régression + classification]

## 4 – Réseaux de neurones [régression + classification]

## 5 – Exercices types

## 6 – Annexes

# Cadre mathématique et objectifs

## Notations

- ▶  $(X_1, Y_1), \dots, (X_n, Y_n) \stackrel{\text{iid}}{\sim} P^{X,Y}$
- ▶  $P^{X,Y}$  : loi inconnue sur  $\mathcal{X} \times \mathcal{Y}$
- ▶  $\mathcal{X} \subset \mathbb{R}^p$ ,  $\mathcal{Y} = \{0, 1, \dots, K-1\}$
- ▶ sauf mention explicite :  $K = 2$  (**classification binaire**)

## Objectif

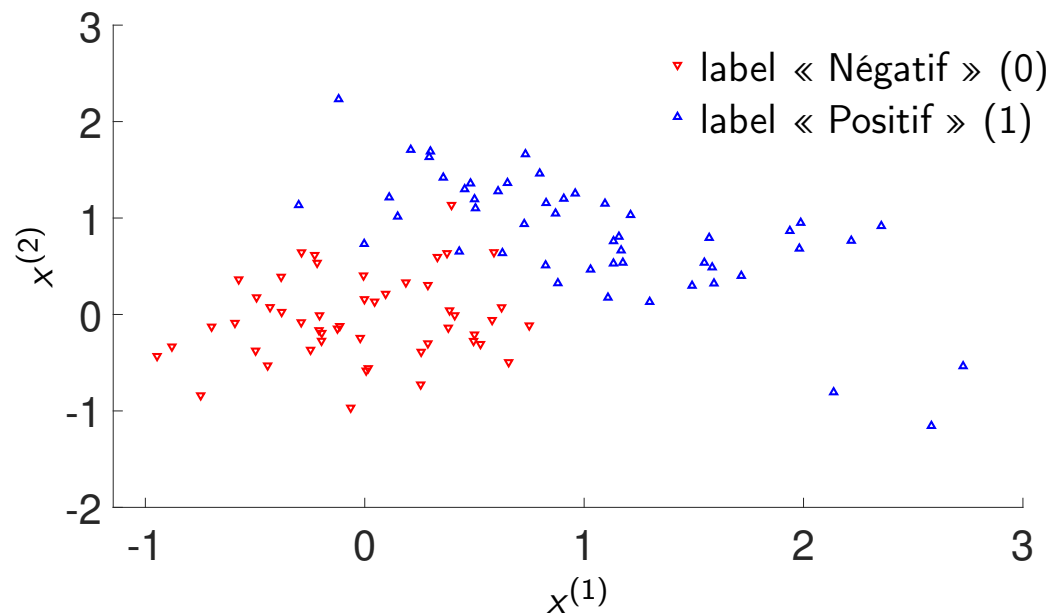
Construire une (bonne) fonction de prédiction  $h : x \mapsto \{0, 1\}$ .

Synonymes : **fonction de classification**, ou « classifieur ».

## Objectifs de cette partie

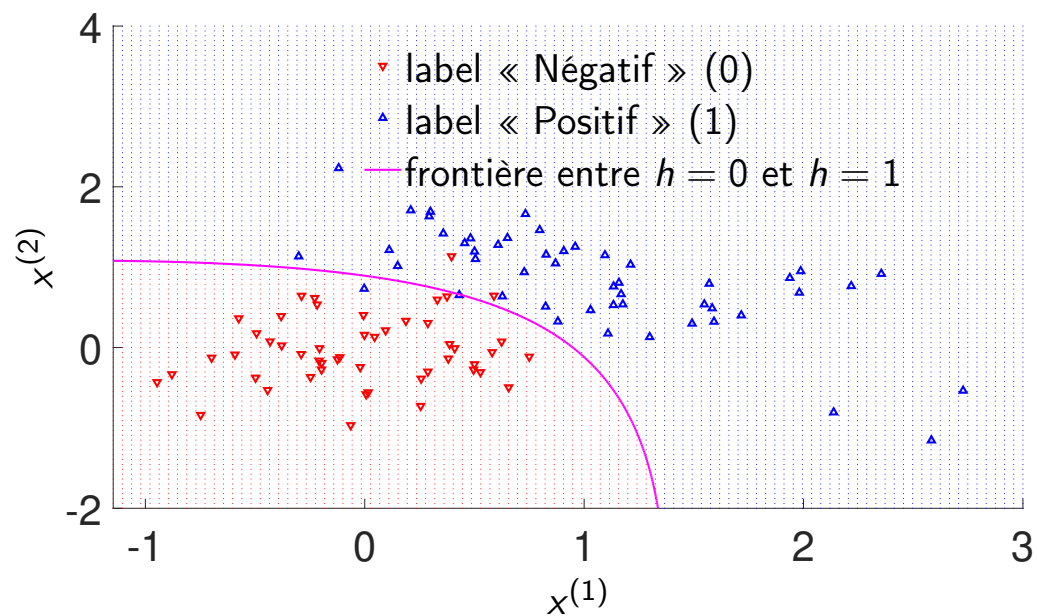
- ▶ introduire la méthode de la **régression logistique**
- ▶ définir des **mesures de risque** appropriées en classification

## Exemple avec deux variables explicatives ( $p = 2$ )



5/81

## Avant-goût de la suite : un classifieur possible



6/81

# Plan du cours

## 1 – Généralités sur la classification

### 1.1 – Introduction

### 1.2 – Fonctions de perte et fonctions de classification optimales

### 1.3 – Mesures de performances

## 2 – Régression logistique [classification]

## 3 – Arbres de décision [régression + classification]

## 4 – Réseaux de neurones [régression + classification]

## 5 – Exercices types

## 6 – Annexes

# Rappel : fonction de perte et risque

### Définition : risque (erreur de généralisation)

Etant donnée une fonction de perte  $L : \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R}_+$  et une fonction de prédiction  $h : \mathcal{X} \rightarrow \mathcal{Y}$ , on définit le **risque**, ou **erreur de généralisation** :

$$R(h) = \mathbb{E} (L(Y, h(X))),$$

l'espérance portant sur le couple  $(X, Y)$ .

 Ce risque **dépend de la loi inconnue** :

$$R(h) = \iint_{\mathcal{X} \times \mathcal{Y}} L(y, h(x)) P^{X,Y}(dx, dy).$$

## Fonctions de classification dures et douces

Considérons un problème à deux classes :  $\mathcal{Y} = \{0, 1\}$ .

### Définition : fonction de classification dure

On appelle **fonction de classification dure** une fonction mesurable  $h : \mathcal{X} \rightarrow \mathcal{Y}$  de l'espace des exemples dans l'espace des étiquettes.

Remarque : cas particulier de fonction de prédiction, cf. cours 6.

### Définition : fonction de classification douce

On appelle **fonction de classification douce** une fonction mesurable  $h : \mathcal{X} \rightarrow [0, 1]$  de l'espace des exemples dans  $[0, 1]$ .

Partant d'une fonction de classification douce  $h : \mathcal{X} \rightarrow [0, 1]$ , on peut construire une famille de fonctions de classification dures, de la forme  $x \mapsto \mathbb{1}_{h(x) \geq \delta}$ , pour  $\delta \in [0, 1]$ .

8/81

## Fonctions de perte usuelles

### Définition : Perte 0/1 pour la classification dure

$$\begin{aligned} L : \mathcal{Y} \times \mathcal{Y} &\rightarrow \mathbb{R}_+ \\ (y, \tilde{y}) &\mapsto L(y, \tilde{y}) = \mathbb{1}_{y \neq \tilde{y}}. \end{aligned}$$

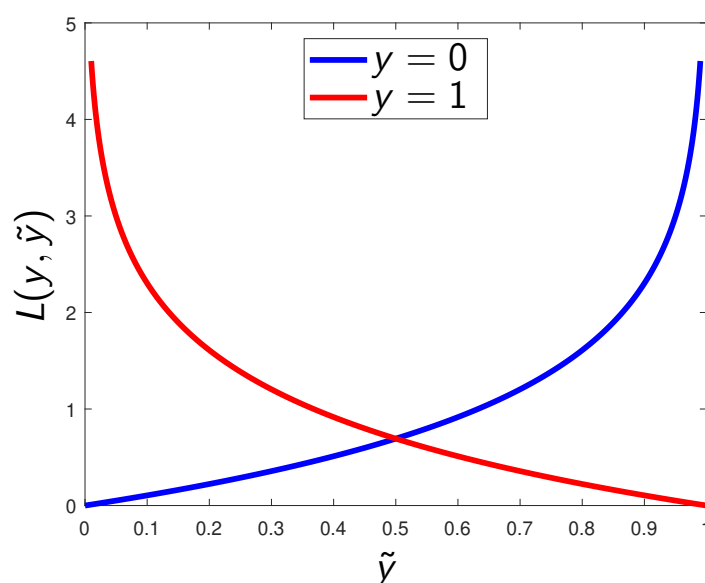
►  $R(h) = \mathbb{P}(Y \neq h(X))$  est la **probabilité de mauvais classement**.

### Définition : Perte logarithmique pour la classification douce

$$\begin{aligned} L : \mathcal{Y} \times [0, 1] &\rightarrow \mathbb{R}_+ \cup \{+\infty\} \\ (y, \tilde{y}) &\mapsto L(y, \tilde{y}) = \begin{cases} -\ln(\tilde{y}) & \text{si } y = 1, \\ -\ln(1 - \tilde{y}) & \text{si } y = 0. \end{cases} \end{aligned}$$

9/81

## Fonctions de perte usuelles (suite)



Remarque : pour les deux fonctions de pertes,

- ▶  $L(y, \tilde{y}) \geq 0$ ,
- ▶  $L(y, \tilde{y}) = 0 \Leftrightarrow \tilde{y} = y$ .

10/81

## Fonctions de classification optimales

### Proposition

$h : \mathcal{X} \rightarrow \mathcal{Y}$  est optimale pour la **perte 0/1** ssi,  $P^X$ -pp,

- ▶  $h(x) = 1$  lorsque  $P(Y = 1 | X = x) > \frac{1}{2}$ ,
- ▶  $h(x) = 0$  lorsque  $P(Y = 1 | X = x) < \frac{1}{2}$ .

où l'on a introduit la notation  $P(A | X = x) = \mathbb{E}(\mathbf{1}_A | X = x)$ .

➡ démonstration : voir TD 7

Par exemple,  $x \mapsto \mathbf{1}_{P(Y=1|X=x) \geq \frac{1}{2}}$  est optimale.

**Remarque** : une formule plus générale peut être établie pour une perte asymétrique ( $L(0, 1) \neq L(1, 0)$ ). Voir polycopié.

11/81

## Fonctions de classification optimales (suite)

### Proposition

$h : \mathcal{X} \rightarrow [0, 1]$  est optimale pour la **perte logarithmique** ssi,  $P^X$ -pp,

$$h(x) = P(Y = 1 \mid X = x).$$

▮ démonstration : exercice 1

Remarque : puisque  $Y$  est à valeurs dans  $\{0, 1\}$ , on a :

$$P(Y = 1 \mid X = x) = \mathbb{E}(Y \mid X = x).$$

☞ classification douce + perte logarithmique  $\approx$  **régression**.

(D'où le nom de la prochaine méthode que nous allons étudier !)

12/81

## Plan du cours

### 1 – Généralités sur la classification

#### 1.1 – Introduction

#### 1.2 – Fonctions de perte et fonctions de classification optimales

#### 1.3 – Mesures de performances

### 2 – Régression logistique [classification]

### 3 – Arbres de décision [régression + classification]

### 4 – Réseaux de neurones [régression + classification]

### 5 – Exercices types

### 6 – Annexes



## Matrice de confusion & grandeurs dérivées

	Réalité Négatif (N)	Réalité Positif (P)
Prédiction Négatif	Vrais Négatifs (VN)	Faux Négatifs (FN)
Prédiction Positif	Faux Positifs (FP)	Vrais Positifs (VP)

### Taux de Vrais Positifs

$$TVP = \frac{VP}{P} = \frac{VP}{VP + FN}$$

(également appelé **sensibilité**)

### Taux de Vrais Négatifs

$$TVN = \frac{VN}{N} = \frac{VN}{VN + FP}$$

(également appelé **spécificité**)

13/81

## Matrice de confusion & grandeurs dérivées (suite)

Autre terminologie, issue du traitement du signal :

- ▶  $1 - TVP$  est le **taux de non-détections** (taux de faux négatifs)
- ▶  $1 - TVN$  est le **taux de fausses alarmes** (taux de faux positifs)

### Application aux fonctions de classification douces

- ▶ Rappel : à toute fonction de classification douce  $h$  on peut associer une famille de fonctions de classification dures

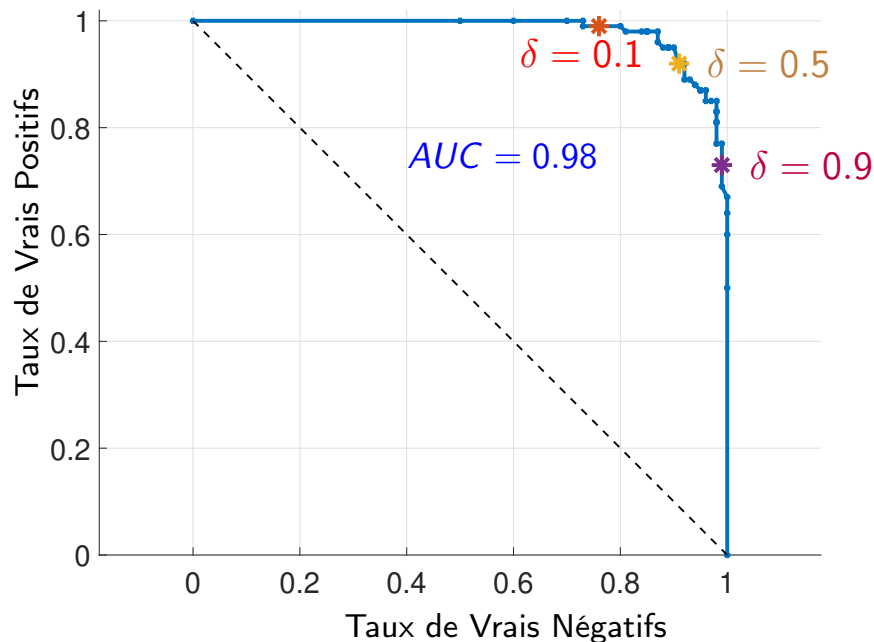
$$h_\delta : x \mapsto \mathbb{1}_{h(x) \geq \delta}, \quad \delta \in [0, 1].$$

- ▶ La valeur de  $\delta$  influe sur le compromis TVN/TVP
  - ▶ quand  $\delta \nearrow$ ,  $TVN \nearrow$ , et  $TVP \searrow$

14/81

## Courbe ROC (Receiver Operating Characteristic)

- ▶ un outil d'**aide à la décision** (choix de  $\delta$ )
- ▶ un outil de **comparaison de classifieurs**
- ▶ grandeur dérivée : **AUC** = Area Under the Curve



15/81

## Plan du cours

### 1 – Généralités sur la classification

### 2 – Régression logistique [classification]

2.1 – Un modèle linéaire pour la classification douce

2.2 – Apprentissage : sélection des coefficients

2.3 – Retour à l'exemple introductif

2.4 – Extensions

### 3 – Arbres de décision [régression + classification]

### 4 – Réseaux de neurones [régression + classification]

### 5 – Exercices types

### 6 – Annexes

# Plan du cours

## 1 – Généralités sur la classification

## 2 – Régression logistique [classification]

### 2.1 – Un modèle linéaire pour la classification douce

### 2.2 – Apprentissage : sélection des coefficients

### 2.3 – Retour à l'exemple introductif

### 2.4 – Extensions

## 3 – Arbres de décision [régression + classification]

## 4 – Réseaux de neurones [régression + classification]

## 5 – Exercices types

## 6 – Annexes

# Un modèle linéaire pour la classification douce

On considère un problème de **classification** (binaire)

►  $\mathcal{X} \subset \mathbb{R}^p$ ,  $\mathcal{Y} = \{0, 1\}$ .

On choisit la **perte logarithmique** :

► l'objectif est d'approcher la fonction de classification optimale

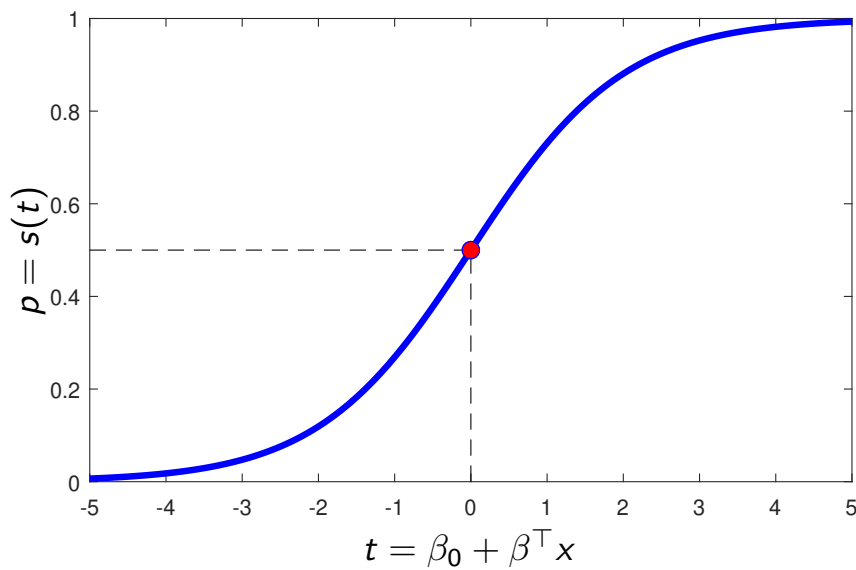
$$h^*(x) = \mathbb{P}(Y = 1 \mid X = x).$$

La **régression logistique** consiste à utiliser des fonctions de classification douces de la forme

$$h(x) = s\left(\beta_0 + \beta^\top x\right),$$

avec  $\beta_0 \in \mathbb{R}$ ,  $\beta \in \mathbb{R}^p$ , et  $s(t) = e^t / (1 + e^t)$  la **fonction logistique**.

## La fonction logistique



⇒ définit une correspondance :  $\beta_0 + \beta^\top x \in \mathbb{R} \longleftrightarrow \text{proba } p \in ]0, 1[$

Aussi connue sous le nom de **fonction sigmoïde**.

17/81

## Un modèle linéaire pour la classification douce (suite)

De manière équivalente,

$$\text{logit}(h(x)) = \beta_0 + \beta^\top x$$

avec

$$\begin{aligned} \text{logit} : ]0, 1[ &\rightarrow \mathbb{R} \\ p &\mapsto \ln\left(\frac{p}{1-p}\right) \end{aligned}$$

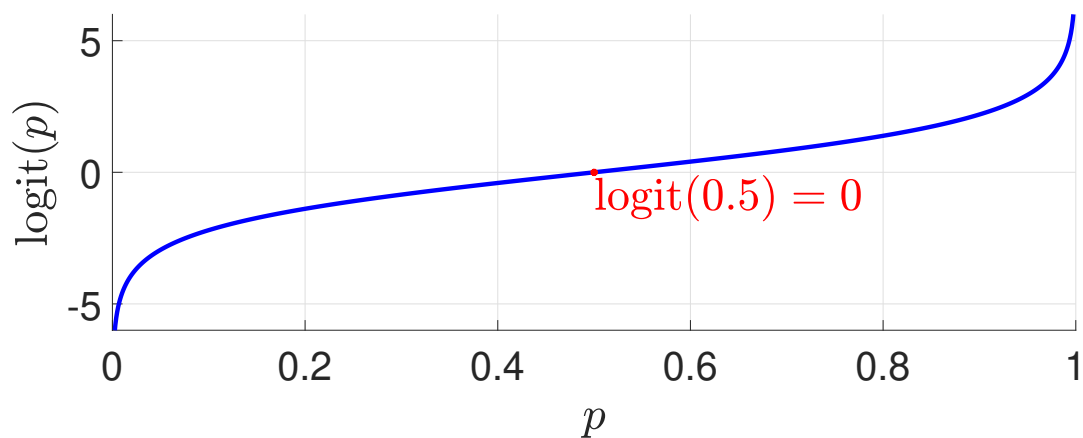
la “**fonction logit**”.

### Propriétés

- ▶ La **fonction logistique**  $s$  est une bijection strictement croissante et  $C^\infty$  de  $\mathbb{R}$  dans  $]0, 1[$ .
- ▶ La **fonction logit** est la bijection réciproque : elle est strictement croissante et  $C^\infty$  de  $]0, 1[$  dans  $\mathbb{R}$ .

18/81

## La fonction logit



⇒ définit une correspondance : proba  $p \in ]0, 1[ \longleftrightarrow \beta_0 + \beta^\top x \in \mathbb{R}$

19/81

## De la classification douce à la classification dure

Etant donné une fonction de **classification douce** de la forme

$$h(x) = s(\beta_0 + \beta^\top x),$$

et un **seuil de décision**  $\delta \in [0, 1]$ , on pose :

$$h_\delta(x) = \mathbb{1}_{h(x) \geq \delta}.$$

⇒  $h_\delta$  sépare les classes dans  $\mathcal{X}$  par un **hyperplan affine** :

$$h_\delta(x) = 1 \iff \beta_0 + \beta^\top x \geq \text{logit}(\delta)$$

Pour la perte 0-1, la valeur  $\delta = \frac{1}{2}$  est généralement utilisée.

20/81

# Plan du cours

## 1 – Généralités sur la classification

## 2 – Régression logistique [classification]

2.1 – Un modèle linéaire pour la classification douce

2.2 – Apprentissage : sélection des coefficients

2.3 – Retour à l'exemple introductif

2.4 – Extensions

## 3 – Arbres de décision [régression + classification]

## 4 – Réseaux de neurones [régression + classification]

## 5 – Exercices types

## 6 – Annexes

# Minimisation du risque empirique

Allègement des notations :  $x \rightarrow \begin{pmatrix} 1 \\ x \end{pmatrix}$  et  $\beta \rightarrow \begin{pmatrix} \beta_0 \\ \beta \end{pmatrix}$

$$\Rightarrow h(x) = \frac{\exp(\beta^\top x)}{1 + \exp(\beta^\top x)}$$

On choisit  $\beta$  par **minimisation du risque empirique** :

$$\hat{\beta} = \operatorname{argmin}_{\beta \in \mathbb{R}^{p+1}} \frac{1}{n} \sum_{i=1}^n L(Y_i, h(X_i)),$$

où  $L$  désigne la **perte logarithmique**.

## Minimisation du risque empirique (suite)

Risque empirique calculé avec la perte logarithmique

$$\sum_{i=1}^n L(y_i, h(x_i)) = -\ln \left( \prod_{i=1}^n (h(x_i))^{y_i} (1 - h(x_i))^{1-y_i} \right)$$

Risque empirique calculé avec la perte logarithmique

$$\sum_{i=1}^n L(y_i, h(x_i)) = -\ln \left( \underbrace{\prod_{i=1}^n (h(x_i))^{y_i} (1 - h(x_i))^{1-y_i}}_{\text{vraisemblance } \mathcal{L}(\beta; \underline{X}, \underline{Y})} \right)$$

$$\text{On a donc : } \underbrace{\frac{1}{n} \sum_{i=1}^n L(Y_i, h(X_i))}_{\text{risque empirique}} = -\underbrace{\frac{1}{n} \ln(\mathcal{L}(\beta; \underline{X}, \underline{Y}))}_{\text{log-vraisemblance}}.$$

$\hat{\beta}$  est donc l'**EMV** dans le modèle paramétrique :

22/81

## Log-vraisemblance

Log-vraisemblance

(voir TD 7)

$$\begin{aligned} \ell(\beta) &= \ln \mathcal{L}(\beta; \underline{X}, \underline{Y}) \\ &= \sum_{i=1}^n \left\{ Y_i \beta^\top X_i - \ln \left( 1 + \exp(\beta^\top X_i) \right) \right\} \end{aligned}$$

### Maximisation de $\ell$

Elle s'effectue à l'aide d'un **algorithme d'optimisation** numérique

➡ par exemple, l'algorithme de Newton-Raphson

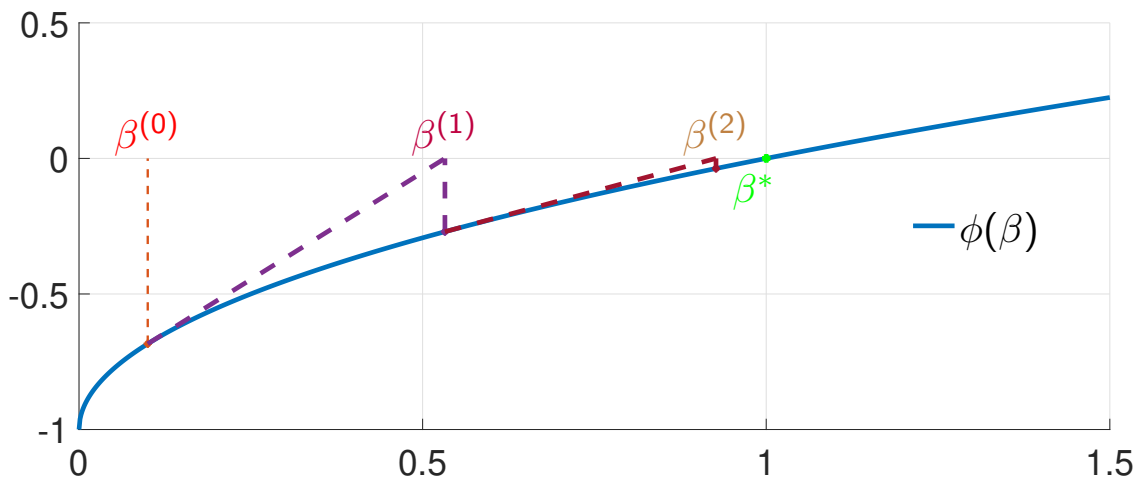
23/81

## Rappel : Algorithme de Newton-Raphson unidimensionnel

Soit  $\phi : \mathbb{R} \rightarrow \mathbb{R}$ . On cherche  $\beta^*$  tel que  $\phi(\beta^*) = 0$ .

L'algorithme de Newton-Raphson est itératif :

- ▶ initialisation :  $\beta^{(0)}$
- ▶ récurrence :  $\beta^{(k+1)} = \beta^{(k)} - \frac{\phi(\beta^{(k)})}{\phi'(\beta^{(k)})}$



24/81

## Maximisation de $\ell$ par la méthode de Newton-Raphson

Même algorithme mais cette fois en dimension  $p + 1$ , avec

- ▶  $\phi \rightarrow \nabla_{\beta} \ell$ ,
- ▶  $\phi' \rightarrow H_{\beta} \ell$ , matrice hessienne  $((p + 1) \times (p + 1))$  de  $\ell$ .

D'où la récurrence :

$$\beta^{(k+1)} = \beta^{(k)} - \left[ H_{\beta} \ell \left( \beta^{(k)} \right) \right]^{-1} \nabla_{\beta} \ell \left( \beta^{(k)} \right).$$

Sous les conditions suivantes :

- ▶  $H_{\beta} \ell$  est une application lipschitzienne,
- ▶  $H_{\beta} \ell \left( \beta^{(0)} \right)$  est inversible,
- ▶  $h_0 = \left[ H_{\beta} \ell \left( \beta^{(0)} \right) \right]^{-1} \nabla_{\beta} \ell \left( \beta^{(0)} \right)$  est suffisamment<sup>†</sup> petit,

l'algorithme converge vers un point  $\beta^*$  tel que  $\nabla_{\beta} \ell (\beta^*) = 0$ .

<sup>†</sup> voir la [page Wikipédia](#) sur le théorème de Kantorovich pour une quantification du « suffisamment ».

25/81



# Plan du cours

## 1 – Généralités sur la classification

## 2 – Régression logistique [classification]

2.1 – Un modèle linéaire pour la classification douce

2.2 – Apprentissage : sélection des coefficients

2.3 – Retour à l'exemple introductif

2.4 – Extensions

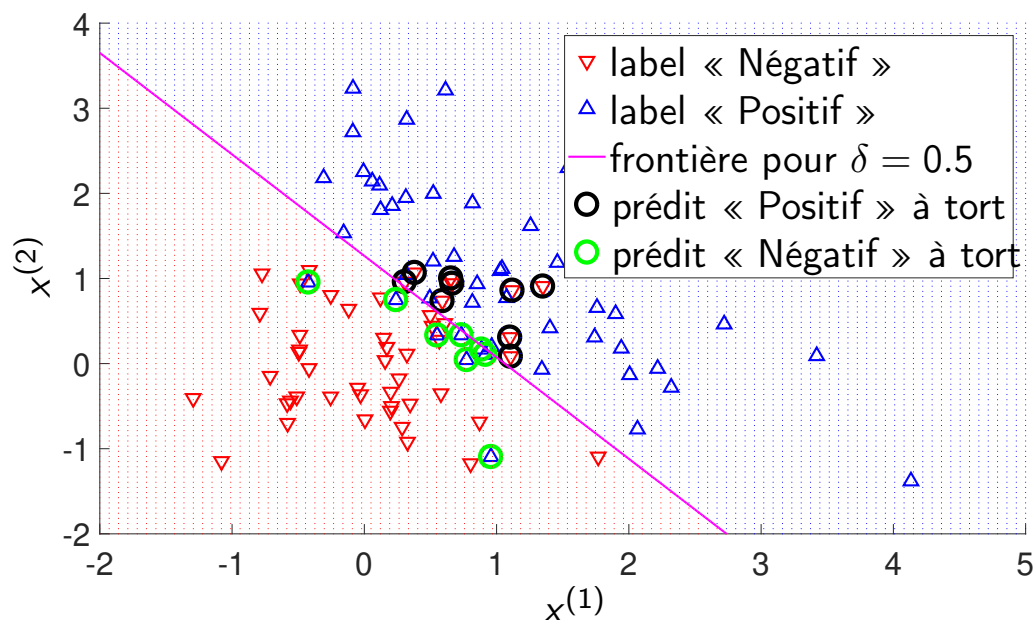
## 3 – Arbres de décision [régression + classification]

## 4 – Réseaux de neurones [régression + classification]

## 5 – Exercices types

## 6 – Annexes

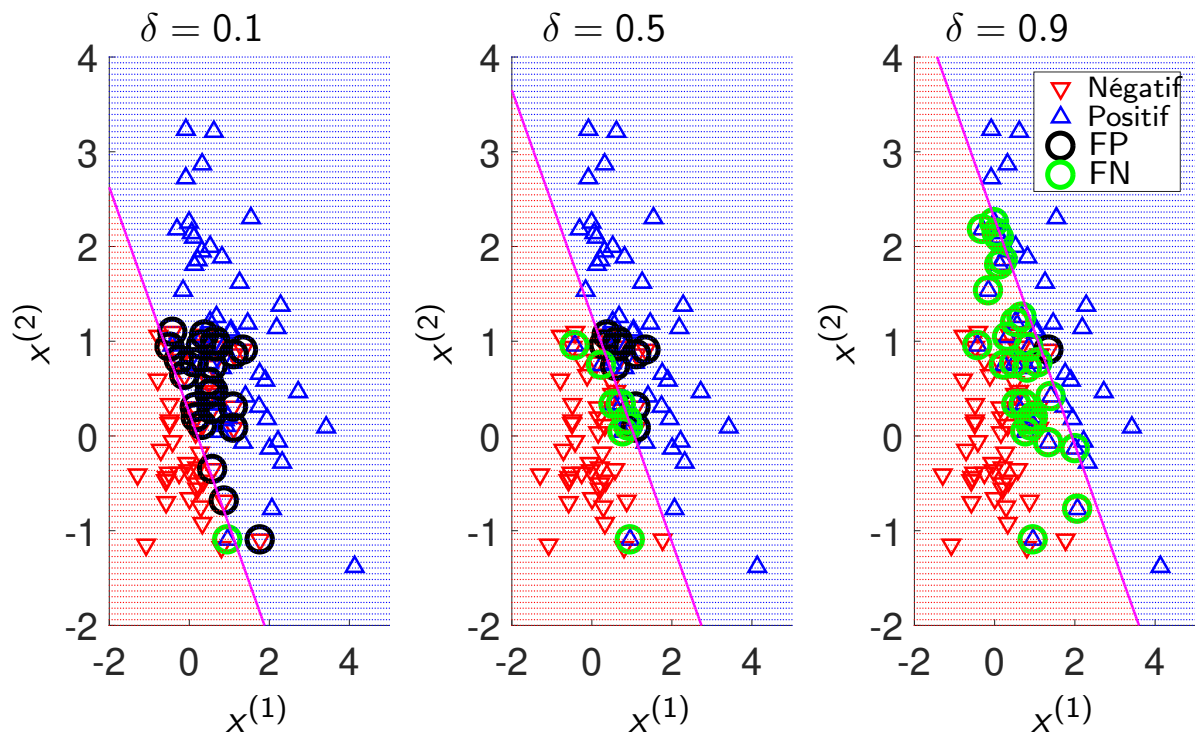
### RL sur l'exemple avec deux variables explicatives



#### Erreurs de prédiction (rappel) :

- ▶ prédire « Positif » un objet de classe « Négatif »
- ▶ prédire « Négatif » un objet de classe « Positif »

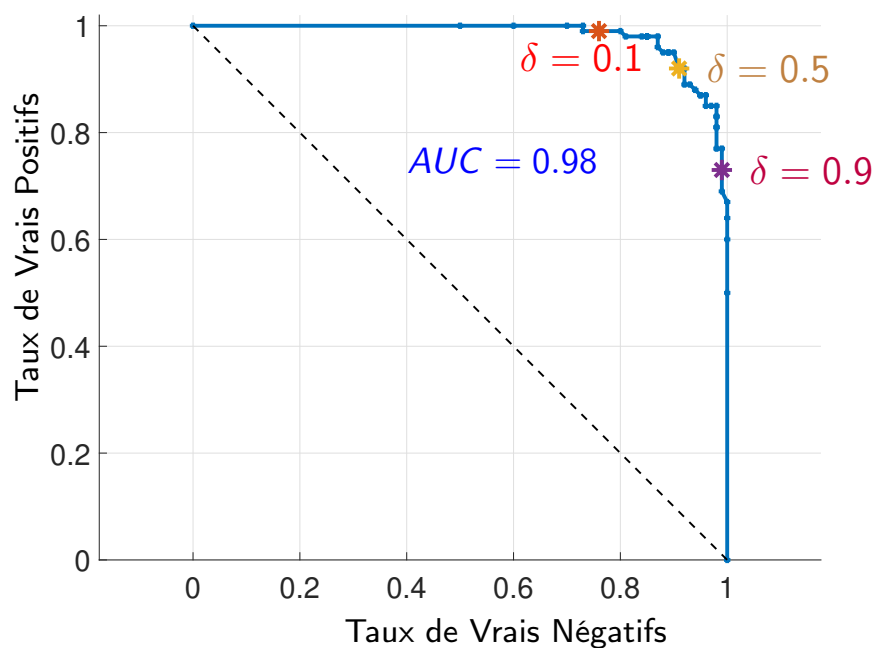
## Influence de $\delta$



27/81

## Courbe ROC (Receiver Operating Characteristic)

- un outil d'**aide à la décision** (choix de  $\delta$ )
- un outil de **comparaison de classifieurs**
- grandeur dérivée : **AUC** = Area Under the Curve



28/81

# Plan du cours

## 1 – Généralités sur la classification

## 2 – Régression logistique [classification]

2.1 – Un modèle linéaire pour la classification douce

2.2 – Apprentissage : sélection des coefficients

2.3 – Retour à l'exemple introductif

2.4 – Extensions

## 3 – Arbres de décision [régression + classification]

## 4 – Réseaux de neurones [régression + classification]

## 5 – Exercices types

## 6 – Annexes

# Extension : grand nombre de variables

## Gérer le cas où $p$ est grand

On **pénalise** la log-vraisemblance :

►  $L^1$  :  $\hat{\beta} = \arg \max_{\beta} [\ell(\beta) - \lambda \|\beta\|_1]$  (LASSO),

►  $L^2$  :  $\hat{\beta} = \arg \max_{\beta} [\ell(\beta) - \lambda \|\beta\|^2]$  (ridge).

⇒ voir cours 8

$p$  est « grand » si  $p \gg n$ , ou même simplement  $p \approx n$

## Extension : plus de deux classes

### Classification multi-classes

Soit  $\{0, 1, \dots, K - 1\}$  l'ensemble des labels (classes),  $K \geq 3$ .

On réalise  $K - 1$  régressions logistiques binaires en considérant une classe (ici « 0 ») comme référence :

$$\begin{cases} \ln \left( \frac{P(Y=\mathbf{1}|X=x)}{P(Y=\mathbf{0}|X=x)} \right) &= \beta_{\mathbf{1},0} + \beta_{\mathbf{1}}^T x \\ \vdots & \\ \ln \left( \frac{P(Y=\mathbf{K-1}|X=x)}{P(Y=\mathbf{0}|X=x)} \right) &= \beta_{\mathbf{K-1},0} + \beta_{\mathbf{K-1}}^T x \end{cases}$$

pour plus de détails : exercice 2

30/81

## Plan du cours

1 – Généralités sur la classification

2 – Régression logistique [classification]

3 – Arbres de décision [régression + classification]

3.1 – Deux exemples introductifs

3.2 – Partitionnement récursif

3.3 – Fonction de prédiction

4 – Réseaux de neurones [régression + classification]

5 – Exercices types

6 – Annexes

# Plan du cours

- 1 – Généralités sur la classification
- 2 – Régression logistique [classification]
- 3 – Arbres de décision [régression + classification]
  - 3.1 – Deux exemples introductifs
  - 3.2 – Partitionnement récursif
  - 3.3 – Fonction de prédiction
- 4 – Réseaux de neurones [régression + classification]
- 5 – Exercices types
- 6 – Annexes

## Classification binaire : détection de spam

Données recueillies sur 4601 messages électroniques

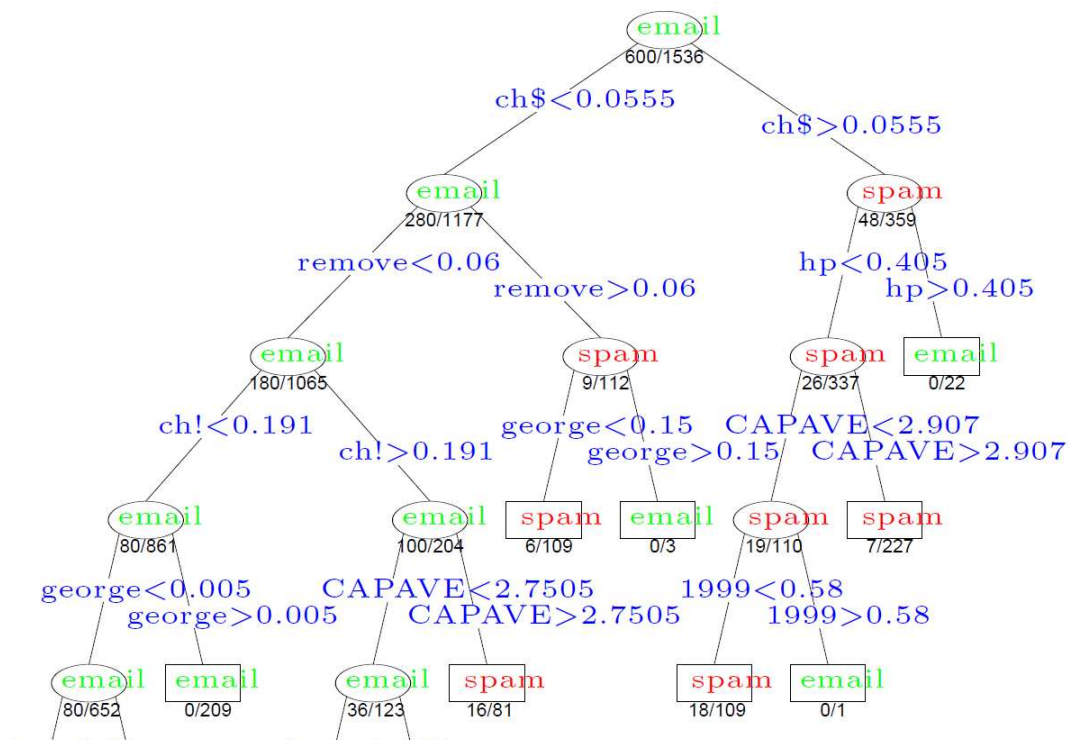
- ▶ variables explicatives : fréquences relatives des 57 mots les plus fréquents
- ▶ variable à expliquer : étiquette « Spam » ou « Email »
  - ➡ variable **qualitative** (binaire ici)

**TABLE 1.1.** Average percentage of words or characters in an email message equal to the indicated word or character. We have chosen the words and characters showing the largest difference between spam and email.

	george	you	your	hp	free	hpl	!	our	re	edu	remove
spam	0.00	2.26	1.38	0.02	0.52	0.01	0.51	0.51	0.13	0.01	0.28
email	1.27	1.27	0.44	0.90	0.07	0.43	0.11	0.18	0.42	0.29	0.01

**TABLE 9.3.** Spam data: confusion rates for the 17-node tree (chosen by cross-validation) on the test data. Overall error rate is 9.3%.

True	Predicted	
	email	spam
email	57.3%	4.0%
spam	5.3%	33.4%



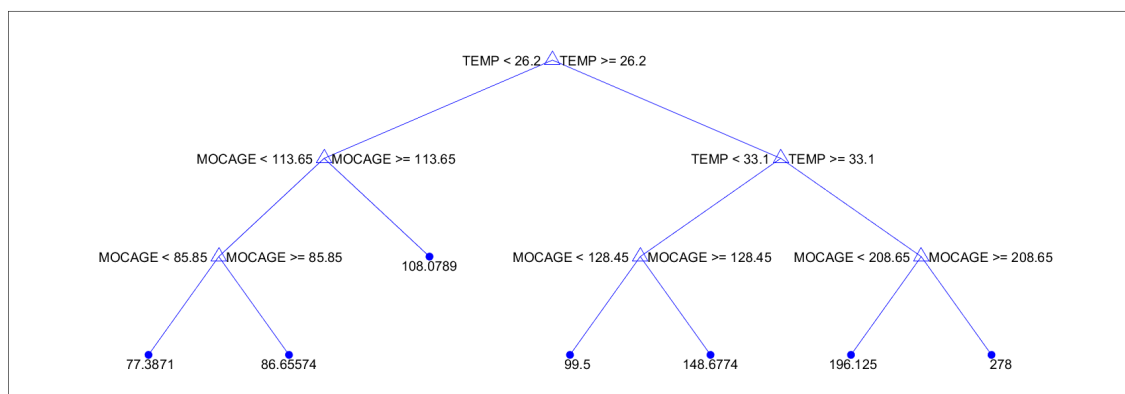
**FIGURE 9.5.** The pruned tree for the **spam** example. The split variables are shown in blue on the branches, and the classification is shown in every node. The numbers under the terminal nodes indicate misclassification rates on the test data.

32/81

## Arbre de régression : exemple « Ozone »

**Exemple simplifié** (afin de permettre des visualisations 2D et 3D)

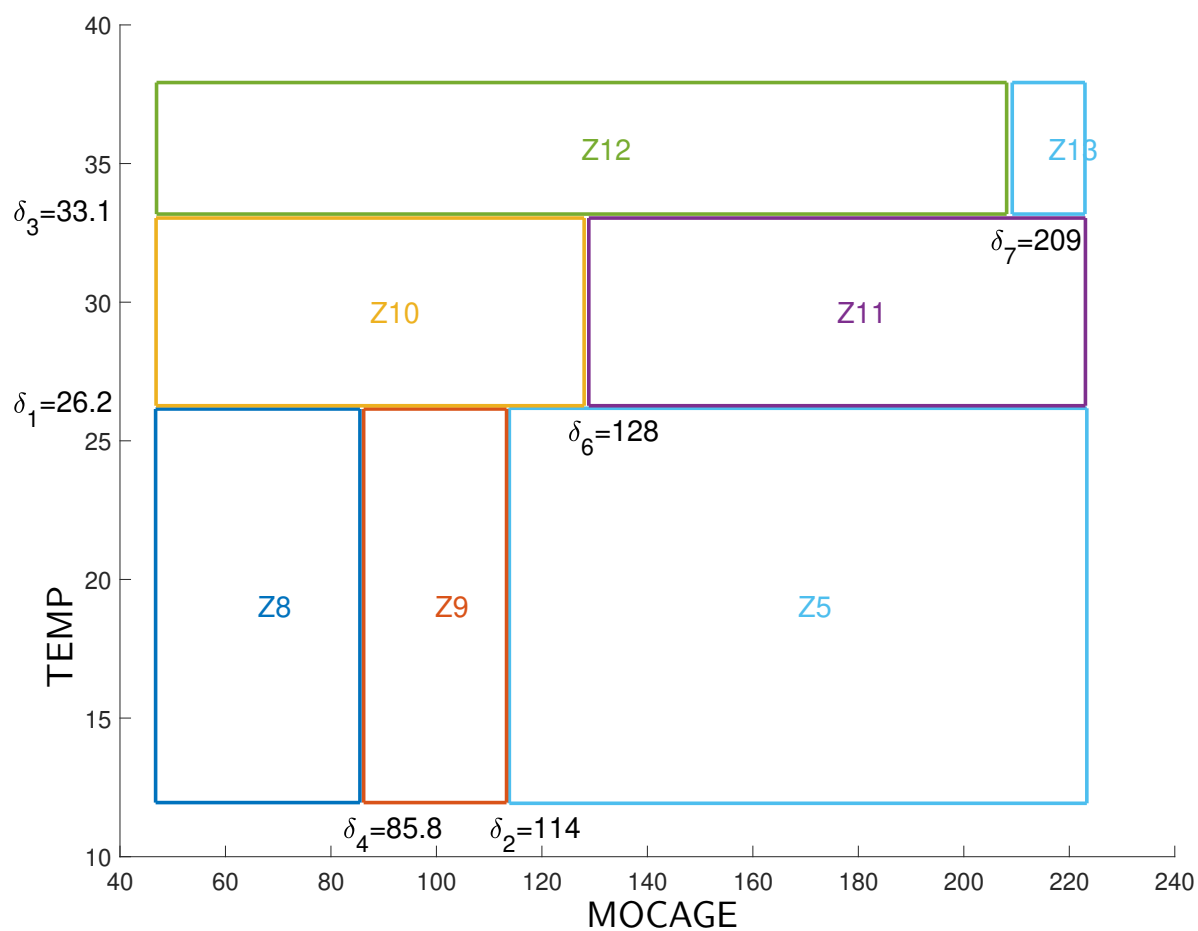
- ▶ prédire la variable O3 (variable **quantitative**)
- ▶ à partir des variables MOCAGE et TEMP



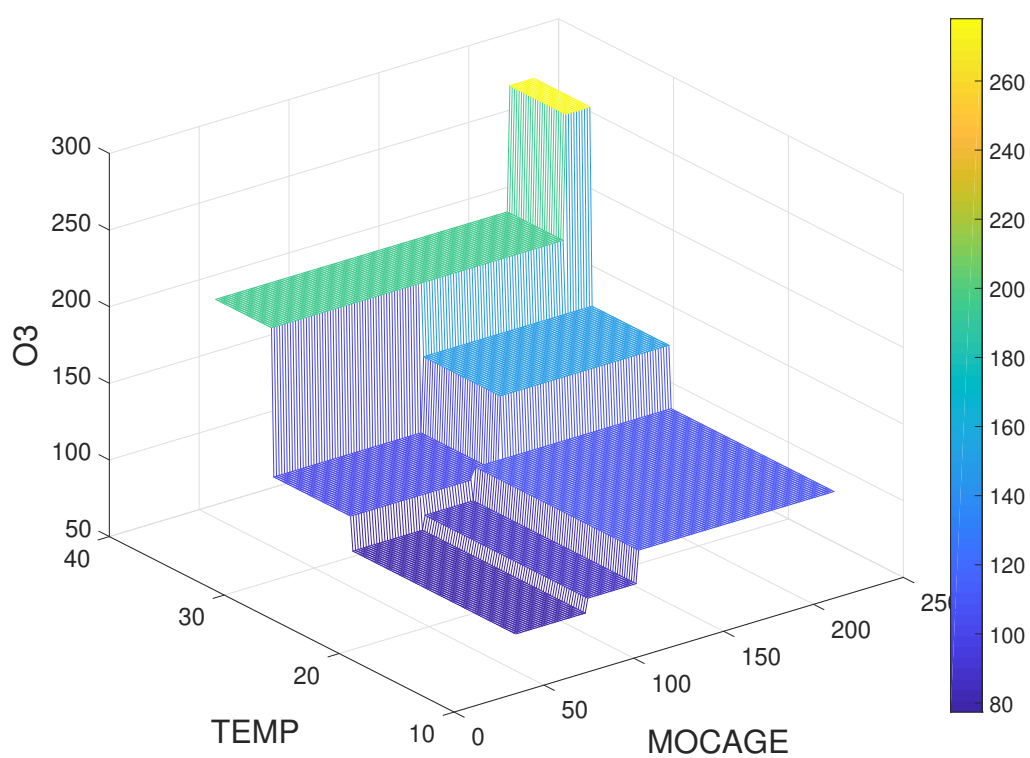
**Vocabulaire.** Quand la variable à prédire est

- ▶ **quantitative**, on parle d'**arbre de régression**,
- ▶ **qualitative**, on parle d'**arbre de classification**.

33/81



## Arbre de régression : exemple « Ozone »



# Plan du cours

1 – Généralités sur la classification

2 – Régression logistique [classification]

3 – Arbres de décision [régression + classification]

3.1 – Deux exemples introductifs

3.2 – Partitionnement récursif

3.3 – Fonction de prédiction

4 – Réseaux de neurones [régression + classification]

5 – Exercices types

6 – Annexes

## Partitionnement récursif : principe général

### Objectif

Construire une partition de  $\mathcal{X}$  à partir des données  $(\underline{X}, \underline{Y})$ .

Principe : construction **itérative** d'une suite  $(\mathcal{P}_m)_{m \geq 1}$  de partitions,

►  $\mathcal{P}_m = \{Z_1^{(m)}, \dots, Z_m^{(m)}\}$ , la partition  $\mathcal{P}_m$  contenant  $m$  parties.

Initialisation :  $\mathcal{P}_1 = \{\mathcal{X}\}$ .

$\mathcal{P}_m \rightarrow \mathcal{P}_{m+1}$  : on **divise** une partie  $Z_{k_m}^{(m)}$  **selon une des variables** :

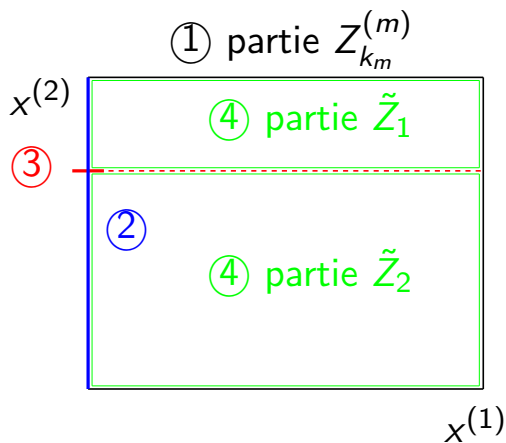
►  $\tilde{Z}_1 = Z_{k_m}^{(m)} \cap \{x \text{ tel que } x^{(j_m)} \leq \delta_m\}$

►  $\tilde{Z}_2 = Z_{k_m}^{(m)} \cap \{x \text{ tel que } x^{(j_m)} > \delta_m\}$

(l'indice  $j_m$  et le seuil  $\delta_m$  restent à préciser)



## Exemple avec $p = 2$



Itération  $\mathcal{P}_m \rightarrow \mathcal{P}_{m+1}$  :

- ▶ ① partie  $Z_{k_m}^{(m)} \in \mathcal{P}_m$
- ▶ ② variable  $x^{(j_m)}$  (ici  $j_m = 2$ )
- ▶ ③ seuil  $\delta_m$
- ▶ ④ création de  $\tilde{Z}_1$  et  $\tilde{Z}_2$

Après division de  $Z_{k_m}^{(m)}$ , on obtient :

$$\mathcal{P}_{m+1} = \mathcal{P}_m \cup \{\tilde{Z}_1, \tilde{Z}_2\} \setminus \{Z_{k_m}^{(m)}\}$$

37/81

## Choisir $k_m$ , $j_m$ et $\delta_m$

Soit  $D(Z)$  une mesure de l'hétérogénéité de la partie  $Z$ .

Exemple (pour  $y$  quantitative)

$$D(Z) = \sum_{i \in Z} (y_i - \bar{y}_Z)^2$$

où  $\bar{y}_Z$  est la moyenne empirique calculée sur la partie  $Z$ .

$k_m$ ,  $j_m$  et  $\delta_m$  sont choisis conjointement de sorte que

$D(Z_{k_m}^{(m)}) - D(\tilde{Z}_1) - D(\tilde{Z}_2)$  soit le plus grand possible

⇒ plus grande décroissance de l'hétérogénéité

(Rappel :  $\tilde{Z}_1$  et  $\tilde{Z}_2$  sont les deux parties créées par division de  $Z_{k_m}^{(m)}$ )

38/81

# Plan du cours

1 – Généralités sur la classification

2 – Régression logistique [classification]

3 – Arbres de décision [régression + classification]

3.1 – Deux exemples introductifs

3.2 – Partitionnement récursif

3.3 – Fonction de prédiction

4 – Réseaux de neurones [régression + classification]

5 – Exercices types

6 – Annexes

## Fonction de prédiction constante par morceaux

Les arbres de décisions réalisent une **prédiction constante par morceaux** sur les éléments de la partition :

$$h_{\beta}(x) = \sum_{k=1}^m \beta_k \mathbb{1}_{Z_k^{(m)}}(x).$$

**Remarque :** **une fois la partition fixée**, il s'agit d'un **modèle linéaire** par rapport aux paramètres  $\beta_1, \dots, \beta_m$ .

## Estimation des coefficients

Principe : pour estimer  $\beta^{(m)} = (\beta_1^{(m)}, \dots, \beta_m^{(m)})$ ,

- ▶ on se donne une **fonction de perte**  $L(y, h_\beta(x))$ ,
- ▶ puis on **minimise le risque empirique**.

Simplification :

$$\begin{aligned}\min_{\beta} \hat{\mathcal{R}}(h_{\beta}) &= \min_{\beta} \sum_{i=1}^n L(y_i, h_{\beta}(x_i)) \\ &= \min_{\beta} \sum_{k=1}^m \sum_{i \in Z_k^{(m)}} L(y_i, \beta_k) \\ &= \sum_{k=1}^m \min_{\beta_k} \sum_{i \in Z_k^{(m)}} L(y_i, \beta_k)\end{aligned}$$

Conséquence :  $\forall k, \hat{\beta}_k^{(m)} = \arg \min_{\beta_k} \sum_{i \in Z_k^{(m)}} L(y_i, \beta_k)$ .

40/81

## Deux cas particuliers importants

### Régression avec la **perte quadratique**

$$\hat{\beta}_k^{(m)} = \operatorname{argmin}_{\beta_k} \sum_{i \in Z_k^{(m)}} (y_i - \beta_k)^2 = \bar{y}_{Z_k^{(m)}}$$

||| démonstration

### Classification binaire avec la **perte logarithmique**

Classification douce :

$$\begin{aligned}\hat{\beta}_k^{(m)} &= \operatorname{argmin}_{\beta_k \in [0,1]} \sum_{i \in Z_k^{(m)}} (-y_i \ln(\beta_k) - (1 - y_i) \ln(1 - \beta_k)) \\ &= \frac{1}{\operatorname{card}(Z_k^{(m)})} \cdot \operatorname{card}(i \in Z_k^{(m)} \text{ tel que } y_i = 1)\end{aligned}$$

||| démonstration

Classification dure : seuiller avec  $\delta = \frac{1}{2}$  (cf. régression logistique).

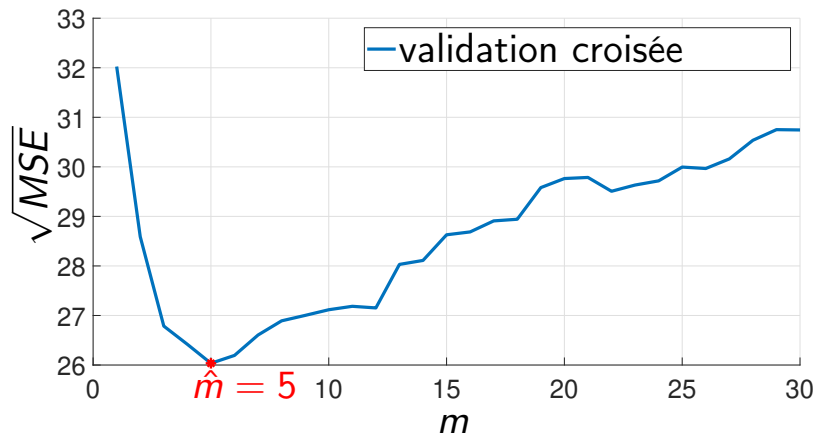
41/81

## Choix de la taille $m$ de la partition

- ▶  $m$  peut être fixé à l'avance (a priori sur la complexité désirée)
- ▶ ou estimé par **validation croisée** (sera vue au cours 8).

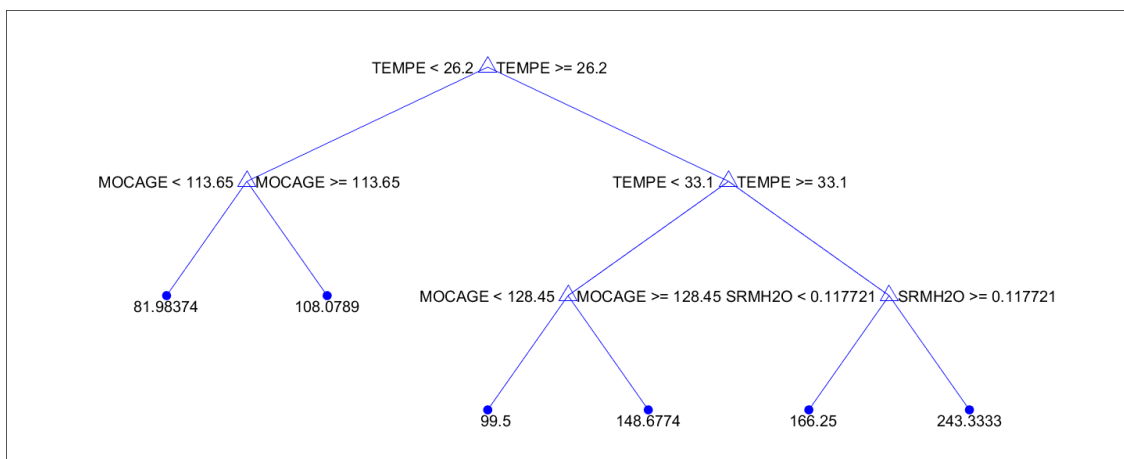
### Exemple « Ozone »

- ▶ Régression de la variable O3 à partir des 7 variables explicatives
- ▶ Choix de  $m$  par validation croisée leave-one-out



42/81

## Arbre de régression : exemple « Ozone »



```
1 if TEMPE<26.2 then node 2 elseif TEMPE>=26.2 then node 3 else 103.433
2 if MOCAGE<113.65 then node 4 elseif MOCAGE>=113.65 then node 5 else 88.1429
3 if TEMPE<33.1 then node 6 elseif TEMPE>=33.1 then node 7 else 153.673
4 fit = 81.9837
5 fit = 108.079
6 if MOCAGE<128.45 then node 8 elseif MOCAGE>=128.45 then node 9 else 138.59
7 if SRMH2O<0.117721 then node 10 elseif SRMH2O>=0.117721 then node 11 else 212.5
8 fit = 99.5
9 fit = 148.677
10 fit = 166.25
11 fit = 243.333
```

43/81

## Pour aller plus loin...

### Défauts des arbres de décision

- ▶ grande sensibilité aux fluctuations de l'échantillon ( $\underline{x}$ ,  $\underline{y}$ )
- ▶ réponse constante sur chaque partie (par définition)  
(y compris lorsque le phénomène à modéliser est régulier)

### Extensions

- ▶ agrégation de modèles (chacun étant un arbre)
  - ▮▮▮ ➔ **Forêts aléatoires**
- ▶ somme pondérée de classifieurs faibles
  - ▮▮▮ ➔ **Boosting** (AdaBoost)

44/81

## Plan du cours

- 1 – Généralités sur la classification
- 2 – Régression logistique [classification]
- 3 – Arbres de décision [régression + classification]
- 4 – Réseaux de neurones [régression + classification]
  - 4.1 – Neurones
  - 4.2 – Le perceptron multi-couches
  - 4.3 – Exemple
  - 4.4 – Autres architectures
- 5 – Exercices types
- 6 – Annexes

# Plan du cours

- 1 – Généralités sur la classification
- 2 – Régression logistique [classification]
- 3 – Arbres de décision [régression + classification]
- 4 – Réseaux de neurones [régression + classification]
  - 4.1 – Neurones
  - 4.2 – Le perceptron multi-couches
  - 4.3 – Exemple
  - 4.4 – Autres architectures
- 5 – Exercices types
- 6 – Annexes

## Le neurone biologique (multipolaire) : axone, dendrites...

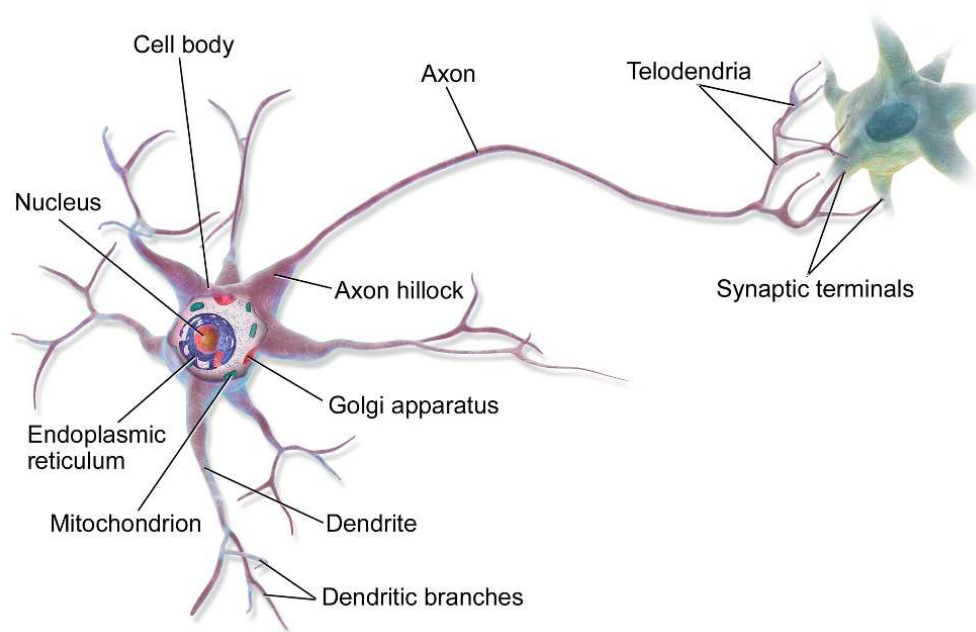


Image : Bruce Blaus, <https://commons.wikimedia.org>, CC BY 3.0

« Un neurone multipolaire est un type de neurone qui possède un seul (généralement long) axone et de nombreux dendrites, permettant l'intégration d'un grand nombre d'informations provenant d'autres neurones. » ([https://fr.wikipedia.org/wiki/Neurone\\_multipolaire](https://fr.wikipedia.org/wiki/Neurone_multipolaire))

## Le neurone artificiel

### Définition : neurone (McCulloch and Pitts, 1943)<sup>†</sup>

En apprentissage statistique, un **neurone** à  $p$  variables (entrées) est une fonction, en général non-linéaire<sup>‡</sup>, de la forme

$$h(x) = \varphi(w x + b), \quad x \in \mathbb{R}^p,$$

où

- ▶  $\varphi$  est une fonction  $\mathbb{R} \rightarrow \mathbb{R}$  croissante ;
- ▶  $w \in \mathbb{R}^{1 \times p}$ , et  $b \in \mathbb{R}$ .

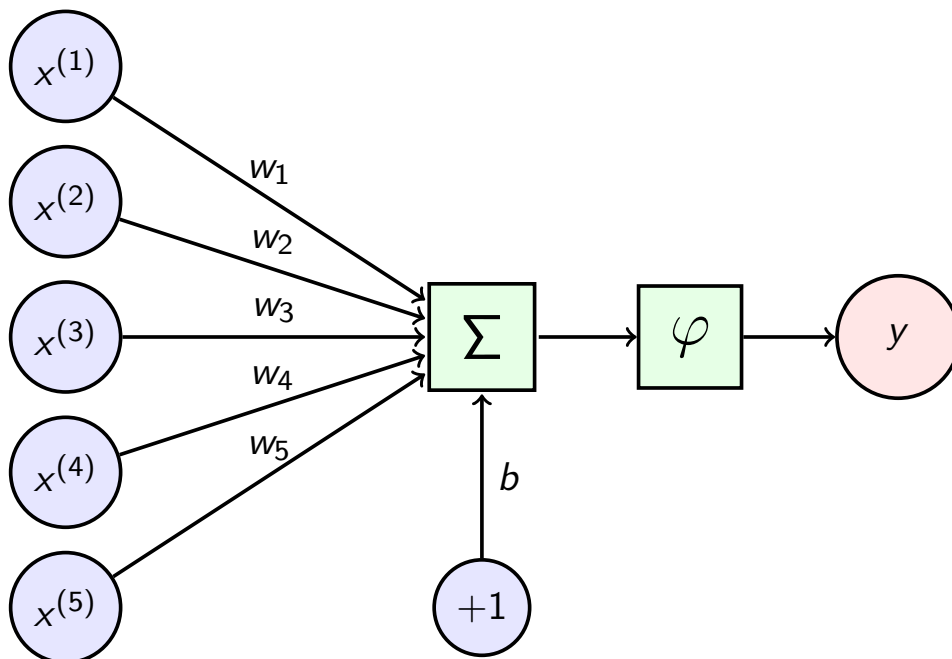
### Vocabulaire

- ▶  $\varphi$  : **fonction d'activation**,
- ▶  $w_1, \dots, w_p$  : **poids**,
- ▶  $b$  : **biais** (rien à voir avec le biais d'un estimateur).

<sup>†</sup> Le neurone de McCulloch & Pitts (1943) utilisait spécifiquement  $\varphi = \text{sgn}$  comme fonction d'activation.

<sup>‡</sup> On verra plus loin une situation où l'on fait intervenir un neurone linéaire ( $\varphi = \text{Id}$ ).

## Le neurone artificiel : illustration ( $p = 5$ )



## Fonctions d'activation

Fonctions d'activation discontinues (non recommandées<sup>†</sup>) :

- ▶ fonction de **Heaviside** :  $\varphi(v) = \mathbb{1}_{v \geq 0}$ , ou
- ▶ fonction **signe** :  $\varphi(v) = \text{sgn}(v) = \mathbb{1}_{v > 0} - \mathbb{1}_{v < 0}$ .

Fonctions en « forme de S », ou **sigmoïdes** :

- ▶ **logistique**<sup>‡</sup> :  $\varphi(v) = \frac{1}{1+e^{-v}} = \frac{1}{2} + \frac{1}{2} \tanh\left(\frac{v}{2}\right)$ , ou
- ▶ **tanh** :  $\varphi(v) = \tanh(v) = \frac{e^v - e^{-v}}{e^v + e^{-v}}$ .

Fonction **ReLU** (*Rectified Linear Unit*) :

- ▶  $\varphi(v) = \max(0, v)$ .

<sup>†</sup> Utilisée dans les modèles les plus anciens, notamment le perceptron de Rosenblatt (1957), mais abandonnées depuis pour cause de dérivée nulle pp, qui rend difficile la minimisation du risque.

<sup>‡</sup> Le terme « sigmoïde » désigne parfois cette fonction particulière.

## Fonctions d'activation (suite)

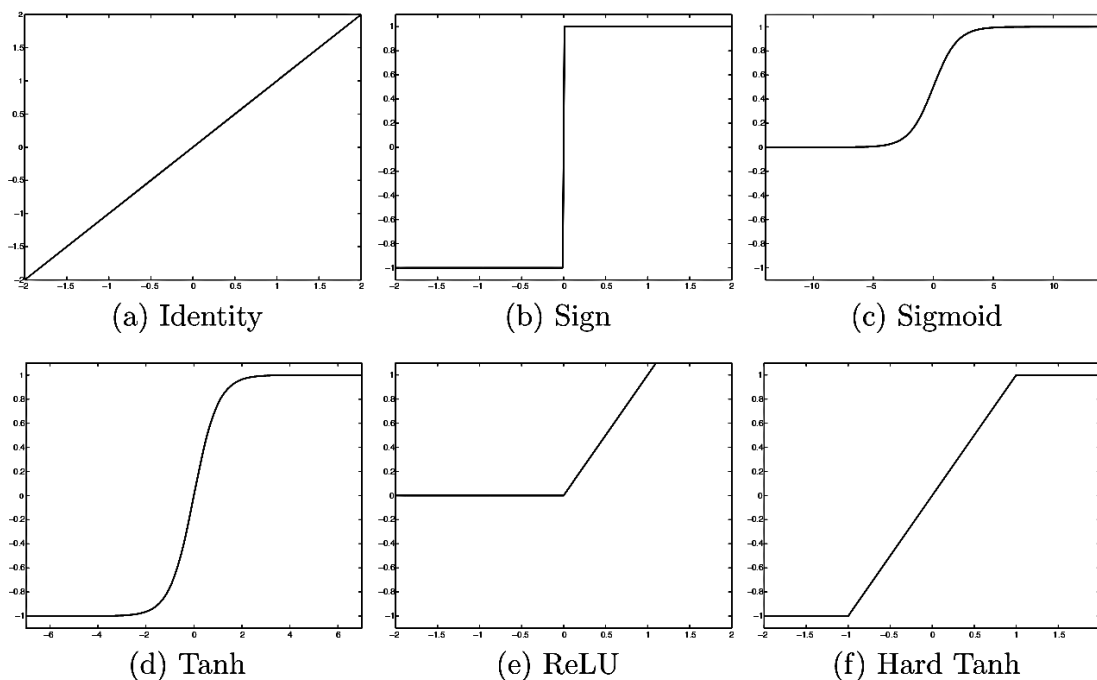


Image : C. C. Aggarwal (2018). *Neural networks and Deep Learning*, Springer.



## Remarque : lien avec la régression logistique

**Remarque.** Avec la **fonction d'activation logistique** (sigmoïde),

$$y = \varphi(v) = \frac{1}{1 + e^{-v}} \quad \Leftrightarrow \quad v = \ln \left( \frac{y}{1 - y} \right).$$

Puisque  $v = wx + b$ , on retrouve pour  $h(x)$  la forme du **prédicteur utilisé en régression logistique**.

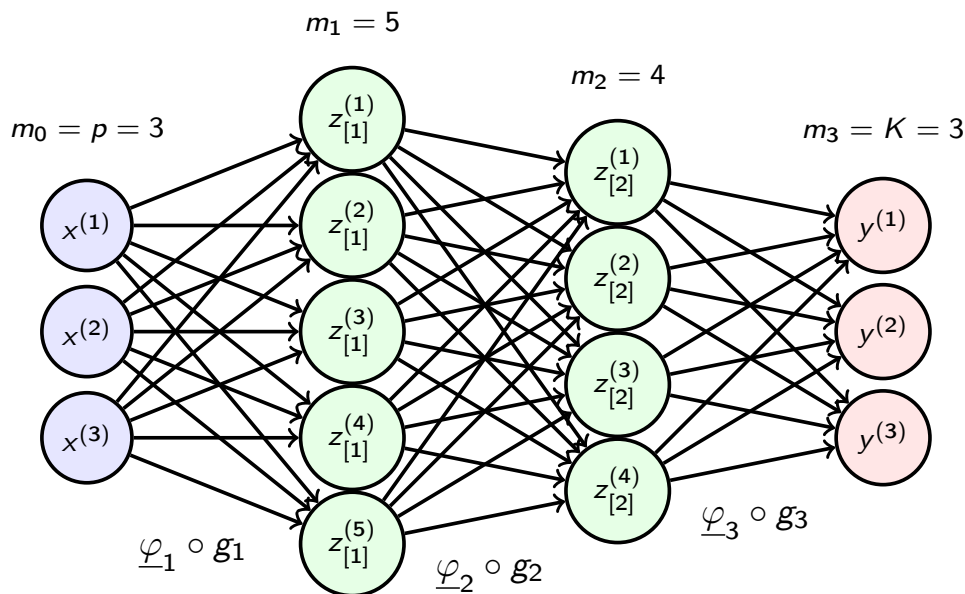
50/81

## Plan du cours

- 1 – Généralités sur la classification
- 2 – Régression logistique [classification]
- 3 – Arbres de décision [régression + classification]
- 4 – Réseaux de neurones [régression + classification]
  - 4.1 – Neurones
  - 4.2 – Le perceptron multi-couches
  - 4.3 – Exemple
  - 4.4 – Autres architectures
- 5 – Exercices types
- 6 – Annexes

## Le perceptron multi-couches : représentation

Exemple de perceptron multi-couches avec  $p = 3$  entrées,  $K = 3$  sorties, et deux couches cachées de tailles  $m_1 = 5$  et  $m_2 = 4$ .



Vocabulaire : réseau complètement connecté, à propagation avant  
(*fully connected, feed-forward neural network*)

51/81

## Le perceptron multi-couches : définition

Soient  $p, K$  des entiers non nuls.

### Définition : perceptron multi-couches<sup>†</sup> (MLP)

On appelle **perceptron multi-couche** à  $M + 1$  couches,  $p$  variables (entrées) et  $K$  réponses (sorties), une fonction  $\mathbb{R}^p \rightarrow \mathbb{R}^K$  de la forme

$$h = \left( \underline{\varphi}_M \circ g_M \right) \circ \cdots \circ \left( \underline{\varphi}_k \circ g_k \right) \circ \cdots \circ \left( \underline{\varphi}_1 \circ g_1 \right),$$

où<sup>‡</sup>

- ▶  $g_k : \mathbb{R}^{m_{k-1}} \rightarrow \mathbb{R}^{m_k}$  est **affine**,
- ▶  $\underline{\varphi}_k : \mathbb{R}^{m_k} \rightarrow \mathbb{R}^{m_k}$  représente l'action coordonnée par coordonnée d'une fonction  $\varphi_k : \mathbb{R} \rightarrow \mathbb{R}$  **croissante**.
- ▶  $m_0, m_1, \dots, m_M$  : entiers non nuls,  $m_0 = p$ ,  $m_M = K$ .

<sup>†</sup> Le perceptron original de Rosenblatt (1957) ne comportait pas de couche cachée ( $M = 1$ ). Il utilisait la fonction d'activation  $h(x) = \text{sgn}(x)$  comme McCulloch et Pitts (1943), et des poids  $w_j \in \{-1, +1, -\infty\}$ .

<sup>‡</sup> à une exception près que l'on verra plus loin (couche « softmax »)

52/81

## Le perceptron multi-couches : définition (suite)

**Vocabulaire :** **couches** de variables

- ▶  $z_{[0]} = x$  : couche d'entrée,
- ▶  $z_{[k]} = (\varphi_k \circ g_k)(z_{[k-1]})$ ,  $1 \leq k < M$  : **couches cachées**,
- ▶  $z_{[M]} = y = (\varphi_M \circ g_M)(z_{[M-1]})$  : couche de sortie.

**Remarque.** Notons

$$g_k(z_{[k-1]}) = W_k z_{[k-1]} + b_k.$$

Alors, pour tout  $j \in \{1, \dots, m_k\}$  on reconnaît un **neurone** :

$$z_{[k]}^{(j)} = \varphi_k \left( w_{k,j} z_{[k-1]} + b_k^{(j)} \right),$$

où  $w_{k,j} = e_j^\top W_k$  est la  $j$ -ème ligne de  $W_k$ .

▮ Vocabulaire : poids, biais, fonction d'activation.

53/81

## Couche de sortie : fonction d'activation

La couche de sortie doit être **adaptée au problème considéré**...

**Régression.**  $\mathcal{Y} \subset \mathbb{R}$ , ou plus généralement  $\mathbb{R}^K$ .

- ▶ Perceptron à  $K$  sorties.
- ▶ Fonction d'activation :  $\varphi_M = \text{Id}$ .
- ▶ La dernière transformation  $(\varphi_M \circ g_M)$  est donc **affine**.

**Classification.**  $K$  classes,  $\mathcal{Y} = [0, 1]^K$  (classification « douce »).

- ▶ Perceptron à  $K$  sorties, avec  $m_{M-1} = m_M = K$ .
- ▶ Exception à la définition ▮ la couche « **softmax** » :

$$z_{[M]}^{(j)} = \frac{\exp \left( z_{[M-1]}^{(j)} \right)}{\sum_{j'=1}^K \exp \left( z_{[M-1]}^{(j')} \right)}, \quad \sum_{j=1}^K z_{[M]}^{(j)} = 1.$$

Remarque : pour la classification *binaire* on peut se contenter d'une seule sortie ( $K = 1$  au lieu de  $K = 2$ ) en plaçant sur la dernière couche une fonction d'activation *logistique*.

54/81

## Apprentissage : fonctions de perte et régularisation

Les fonctions de pertes les plus couramment utilisées<sup>†</sup> sont

- ▶ **régression** : la **perte quadratique**
  - ▶  $L(y, \tilde{y}) = (y - \tilde{y})^2$  si une seule sortie,
  - ▶  $L(y, \tilde{y}) = \|y - \tilde{y}\|^2$  si  $K > 1$ .
- ▶ **classification (douce)** : la **perte logarithmique**
  - ▶ Pour tout  $j \in \{1, \dots, K\}$ , on a  $y^{(j)} \in \{0, 1\}$  et  $\tilde{y}^{(j)} \in [0, 1]$ .
  - ▶  $L(y, \tilde{y}) = -\sum_{j=1}^K y^{(j)} \ln(\tilde{y}^{(j)})$ .

Nb paramètres élevé  $\Rightarrow$  **régulariser** pour éviter le **sur-apprentissage**

- ▶ **pénalisation**, par exemple  $L^1$  (LASSO) ou  $L^2$  (ridge) ;
- ▶ autres (hors programme) : early stopping, drop out. . .

<sup>†</sup> par exemple [https://scikit-learn.org/stable/modules/neural\\_networks\\_supervised.html](https://scikit-learn.org/stable/modules/neural_networks_supervised.html)

## Apprentissage : optimisation numérique

On doit **minimiser le risque empirique** (éventuellement pénalisé)

$$\hat{\mathcal{R}}_n(\theta) = \frac{1}{n} \sum_{i=1}^n L(Y_i, h_{\theta}(X_i)),$$

où  $\theta$  désigne le vecteur des paramètres du modèle (poids, biais).

⇒ Recours à des **méthodes numériques**.

Ces méthodes utilisent le **gradient du critère**. Deux remarques :

- ▶ alléger les calculs qd  $n$  est grand : « **mini-batches** » aléatoires
  - ⇒ méthode du **gradient stochastique** (hors programme) ;
- ▶ calcul récursif du **gradient d'une fonction composée**
  - ⇒ méthode de la **rétro-propagation** (hors programme).

# Plan du cours

- 1 – Généralités sur la classification
- 2 – Régression logistique [classification]
- 3 – Arbres de décision [régression + classification]
- 4 – Réseaux de neurones [régression + classification]
  - 4.1 – Neurones
  - 4.2 – Le perceptron multi-couches
  - 4.3 – Exemple
  - 4.4 – Autres architectures
- 5 – Exercices types
- 6 – Annexes

## Exemple : MNIST



70 000 images<sup>†</sup> de  $28 \times 28$  pixels (256 niveau de gris)

Problème : classification multi-classes (10 classes) ;  
apprentissage : 60 000 images / test : 10 000 images

Source : <https://www.openml.org/search?type=data&id=554>

## Exemple : MNIST

▮▮▮▮ ➡ voir notebook Jupyter / Python / Scikit-Learn

58/81

## Plan du cours

- 1 – Généralités sur la classification
- 2 – Régression logistique [classification]
- 3 – Arbres de décision [régression + classification]
- 4 – Réseaux de neurones [régression + classification]**
  - 4.1 – Neurones
  - 4.2 – Le perceptron multi-couches
  - 4.3 – Exemple
  - 4.4 – Autres architectures
- 5 – Exercices types
- 6 – Annexes

# Réseaux de neurones convolutionnels (CNNs)

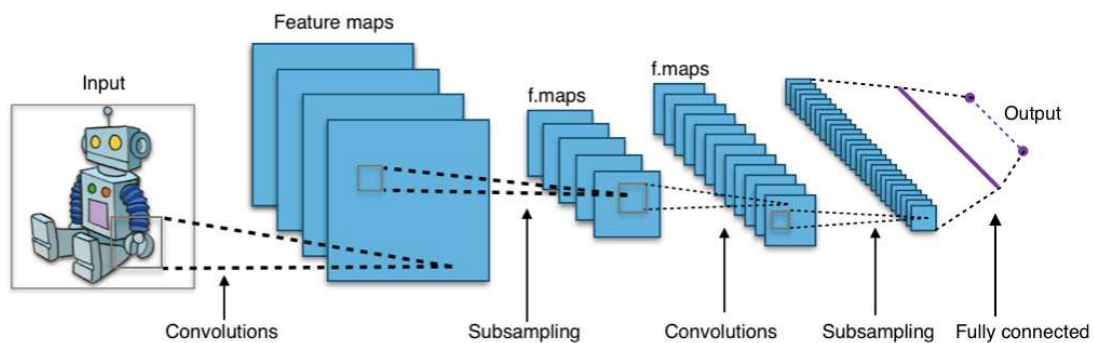


Schéma de principe d'un réseau convolutionnel typique

Image : Aphex34, <https://commons.wikimedia.org>, CC BY-SA 4.0

59/81

## Conclusion du cours et transition vers la séquence suivante

Nous avons vu et développerons en TD 7 :

- ▶ les mesures de performance pour la tâche de classification,
- ▶ la construction et l'utilisation de la régression logistique.
- ▶ les méthodes d'arbres et de réseaux de neurones.

Nous aborderons dans la séquence 8 :

- ▶ les limites des méthodes des moindres carrés,
- ▶ les méthodes de pénalisation pour y remédier,
- ▶ des techniques pour choisir une bonne pénalisation.

60/81

## Plan du cours

1 – Généralités sur la classification

2 – Régression logistique [classification]

3 – Arbres de décision [régression + classification]

4 – Réseaux de neurones [régression + classification]

5 – Exercices types

5.1 – Énoncés

5.2 – Corrigés

6 – Annexes

## Plan du cours

1 – Généralités sur la classification

2 – Régression logistique [classification]

3 – Arbres de décision [régression + classification]

4 – Réseaux de neurones [régression + classification]

5 – Exercices types

5.1 – Énoncés

5.2 – Corrigés

6 – Annexes



## Exercice 1 (Classifieur optimal pour la perte logarithmique)

corrigé

### Question

Montrer la proposition du [slide 12](#) :

La fonction de classification  $h : \mathcal{X} \rightarrow [0, 1]$  est optimale pour la **perte logarithmique** ssi,  $P^X$ -pp,

$$h(x) = P(Y = 1 \mid X = x).$$

61/81

## Exercice 2 (Régression logistique multi-classes)

corrigé

Lorsque le nombre de classes  $K$  est supérieur ou égal à 3, on parle de **Classification multi-classes**.

Soit  $\{0, 1, \dots, K - 1\}$  l'ensemble des labels (classes),  $K \geq 3$ .

La régression logistique binaire (2 classes) s'étend au cas multi-classes en :

- ▶ considérant une classe (ici « 0 ») comme référence,
- ▶ réalisant  $K - 1$  régressions logistiques binaires :

$$\begin{cases} \ln \left( \frac{P(Y=\mathbf{1}|X=x)}{P(Y=\mathbf{0}|X=x)} \right) &= \beta_{\mathbf{1},0} + \beta_{\mathbf{1}}^\top x \\ \vdots & \\ \ln \left( \frac{P(Y=\mathbf{K-1}|X=x)}{P(Y=\mathbf{0}|X=x)} \right) &= \beta_{\mathbf{K-1},0} + \beta_{\mathbf{K-1}}^\top x \end{cases}$$

62/81

Afin d'alléger les notations, on supposera que la matrice des variables explicatives contient un vecteur constant et on fera ainsi le changement :  $\beta_k \leftarrow (\beta_{k,0}, \beta_k)$

### Questions

- 1 Exprimer  $P(Y = k|X = x)$  en fonction des  $\beta_k$ ,
- 2 En déduire que le choix de la classe de référence n'a pas d'influence sur le modèle de régression,
- 3 Ecrire la log-vraisemblance

## Plan du cours

1 – Généralités sur la classification

2 – Régression logistique [classification]

3 – Arbres de décision [régression + classification]

4 – Réseaux de neurones [régression + classification]

5 – Exercices types

5.1 – Énoncés

5.2 – Corrigés

6 – Annexes

Le classifieur optimal  $h^*$  minimise  $\mathbb{E}(L(Y, h(X)))$ , l'espérance portant sur  $(X, Y)$ .

Par conditionnement, il vient :

$$\mathbb{E}_{(X,Y)}(L(Y, h(X))) = \mathbb{E}_X(\mathbb{E}_{Y|X}(L(Y, h(X))|X))$$

Ainsi :

$$h^* = \operatorname{argmin}_h \mathbb{E}(L(Y, h(X)))$$

$$\Downarrow$$

$$h^*(x) = \operatorname{argmin}_{t \in \mathcal{Y}} \underbrace{\mathbb{E}(L(Y, t) \mid X = x)}_{\mathcal{J}(t)} \quad P^X\text{-pp.}$$

Avec la perte logarithmique :

$$\begin{aligned} \mathcal{J}(t) &= \mathbb{E}_{Y|X}(L(Y, t)|X = x) \\ &= \mathbb{P}(Y = 1|X = x)L(1, t) + \mathbb{P}(Y = 0|X = x)L(0, t) \\ &= \mathbb{P}(Y = 1|X = x)(L(1, t) - L(0, t)) + L(0, t) \\ &= \mathbb{P}(Y = 1|X = x)(-\ln(t) + \ln(1 - t)) - \ln(1 - t) \end{aligned}$$

La minimisation de  $\mathcal{J}(t)$  s'obtient en étudiant le signe de la dérivée de  $\mathcal{J}(t)$  :

$$\begin{aligned}\mathcal{J}'(t) &= \mathbb{P}(Y = 1|X = x) \left( -\frac{1}{t} - \frac{1}{1-t} \right) + \frac{1}{1-t} \\ &= \mathbb{P}(Y = 1|X = x) \left( -\frac{1}{t(1-t)} \right) + \frac{1}{1-t} \\ &= \frac{1}{1-t} \left( 1 - \frac{\mathbb{P}(Y = 1|X = x)}{t} \right)\end{aligned}$$

Pour  $\mathbb{P}(Y = 1|X = x) \in ]0, 1[$ ,  $\mathcal{J}'(t)$  est donc :

- ▶ strictement négative pour  $t \in ]0, \mathbb{P}(Y = 1|X = x)[$ ,
- ▶ nulle en  $t = \mathbb{P}(Y = 1|X = x)$ ,
- ▶ strictement positive pour  $t \in ]\mathbb{P}(Y = 1|X = x), 1[$ ,

**Bilan :**  $t = \mathbb{P}(Y = 1|X = x)$  est l'unique minimiseur sur  $[0, 1]$  de la fonction  $\mathcal{J}(t)$ .

Lorsque

- ▶  $\mathbb{P}(Y = 1|X = x) = 0$ ,  $\mathcal{J}(t)$  est minimale en  $t = 0$ ,
- ▶  $\mathbb{P}(Y = 1|X = x) = 1$ ,  $\mathcal{J}(t)$  est minimale en  $t = 1$ .

On en conclut que :

- ①  $h^* : x \mapsto \mathbb{P}(Y = 1|X = x)$  est optimale,
- ②  $h^*(x)$  est l'unique minimiseur de la fonction  $\mathcal{J} : t \mapsto \mathbb{E}_{Y|X} (L(Y, t)|X = x)$ .

Réciproquement : supposons  $h$  optimale.

Soit la fonction :

$$g(x) = \mathbb{E}_{Y/X}(L, Y, h(X))/X = x) - \mathbb{E}_{Y/X}(L, Y, h^*(X))/X = x)$$

On a :

- ▶  $\mathbb{E}_X(g(X)) = \int_{\mathcal{X}} g(x) dP^X = 0$  (sinon  $h$  ne serait pas optimale),
- ▶  $\forall x, g(x) \geq 0$  par optimalité de  $h^*$ .

Ainsi  $g = 0$   $P^X$  - pp, c'est-à-dire :

$$\mathbb{P}(X \in F) = 0 \text{ avec } F = \{x \in \mathcal{X} \text{ t.q. } g(x) > 0\}$$

$h^*(x)$  étant l'unique minimiseur de la fonction

$\mathcal{J} : t \mapsto \mathbb{E}_{Y|X}(L(Y, t)|X = x)$ , on a :

$$F = \{x \in \mathcal{X} \text{ t.q. } h(x) \neq h^*(x)\}$$

On en conclut que  $h = h^*$   $P^X$  - p.p., d'où l'équivalence cherchée.

❶ Pour  $k \in \{1, \dots, K-1\}$  :

$$P(Y = k|X = x) = \exp(\beta_k^\top x) P(Y = 0|X = x).$$

Utilisant  $\sum_{k=0}^{K-1} P(Y = k|X = x) = 1$ , il vient :

$$\begin{cases} P(Y = k|X = x) &= \frac{\exp(\beta_k^\top x)}{1 + \sum_{k' \neq 0} \exp(\beta_{k'}^\top x)}, \quad k \neq 0 \\ P(Y = 0|X = x) &= \frac{1}{1 + \sum_{k' \neq 0} \exp(\beta_{k'}^\top x)} \end{cases} \quad (1)$$

❷ Soit  $\tilde{k} \neq 0$ . Supposons que l'on ait pris la classe  $\tilde{k}$  comme référence et soit pour  $k \neq \tilde{k}$  le vecteur  $\tilde{\beta}_k$  du modèle :

$$\ln \left( \frac{P(Y = k|X = x)}{P(Y = \tilde{k}|X = x)} \right) = \tilde{\beta}_k^\top x, \quad k \neq \tilde{k}$$

Comme

$$\begin{aligned} \ln \left( \frac{P(Y=k|X=x)}{P(Y=\tilde{k}|X=x)} \right) &= \ln \left( \frac{P(Y=k|X=x) P(Y=0|X=x)}{P(Y=0|X=x) P(Y=\tilde{k}|X=x)} \right) \\ &= \beta_k^\top x - \beta_{\tilde{k}}^\top x, \end{aligned}$$

il vient :

$$\begin{cases} \tilde{\beta}_k &= \beta_k - \beta_{\tilde{k}}, \quad k \neq 0, \quad k \neq \tilde{k} \\ \tilde{\beta}_0 &= -\beta_{\tilde{k}} \end{cases} \quad (2)$$

Utilisant le résultat de la question ❶, on a, pour  $k \neq \tilde{k}$  :

$$\begin{cases} P(Y = k|X = x) = \frac{\exp(\tilde{\beta}_k^\top x)}{1 + \sum_{k' \neq \tilde{k}} \exp(\tilde{\beta}_{k'}^\top x)}, & k \neq \tilde{k} \\ P(Y = 0|X = x) = \frac{1}{1 + \sum_{k' \neq \tilde{k}} \exp(\tilde{\beta}_{k'}^\top x)} \end{cases}$$

En substituant dans ces équations les  $\tilde{\beta}_k$  à l'aide des relations (2), il vient les équations (1).

**Bilan.** Changer de référence revient à paramétrer autrement le modèle (sans changer de modèle).

❸ Afin de faire le lien avec la régression logistique binaire, on représente l'observation  $y_i \in \{0, \dots, K-1\}$  par le vecteur  $z_i \in \{0, 1\}^K$  :

$$z_{i,k} = \begin{cases} 1 & \text{si } y_i = k \\ 0 & \text{sinon} \end{cases}$$

On note également  $\beta$  l'ensemble des vecteurs  $\beta_1, \dots, \beta_{K-1}$ .

On écrit la vraisemblance associée à l'exemple  $(x_i, y_i)$  :

$$\begin{aligned}
 P_{\beta}^{Y_i|X_i}(y_i|x_i) &= \prod_{k=0}^{K-1} P_{\beta}^{Y_i|X_i}(k|x_i)^{z_{i,k}} \\
 &= P_{\beta}^{Y_i|X_i}(0|x_i)^{1-z_{i,1}-\dots-z_{i,K-1}} \left( \prod_{k=1}^{K-1} P_{\beta}^{Y_i|X_i}(k|x_i)^{z_{i,k}} \right) \\
 &= P_{\beta}^{Y_i|X_i}(0|x_i) \prod_{k=1}^{K-1} \left( \frac{P_{\beta}^{Y_i|X_i}(k|x_i)}{P_{\beta}^{Y_i|X_i}(0|x_i)} \right)^{z_{i,k}}
 \end{aligned}$$

Utilisant :

- les  $(K - 1)$  modèles de régression :

$$\frac{P_{\beta}^{Y_i|X_i}(k|x_i)}{P_{\beta}^{Y_i|X_i}(0|x_i)} = \exp(\beta_k^{\top} x_i)^{z_{i,k}},$$

- la deuxième équation de (1) :

$$P(Y = 0|X = x) = \frac{1}{1 + \sum_{k'=1}^{K-1} \exp(\beta_{k'}^{\top} x)},$$

il vient la log-vraisemblance :

$$\ell(\beta) = \sum_{k=1}^{K-1} \sum_{i=1}^n z_{i,k} \beta_k^{\top} x_i - \ln \left( 1 + \sum_{k'=1}^{K-1} \exp(\beta_{k'}^{\top} x_i) \right)$$



# Plan du cours

- 1 – Généralités sur la classification
- 2 – Régression logistique [classification]
- 3 – Arbres de décision [régression + classification]
- 4 – Réseaux de neurones [régression + classification]
- 5 – Exercices types
- 6 – Annexes

## Arbre de décision : estimation des coefficients

### Régression avec la perte quadratique

Il s'agit de montrer que :

$$\hat{\beta}_k^{(m)} = \operatorname{argmin}_{\beta_k} \sum_{i \in Z_k^{(m)}} (y_i - \beta_k)^2 = \bar{y}_{Z_k^{(m)}}$$

On a

$$\sum_{i \in Z_k^{(m)}} (y_i - \beta_k)^2 = \operatorname{card}(Z_k^{(m)}) \left( \bar{y}_{Z_k^{(m)}} - \beta_k \right)^2 + \sum_{i \in Z_k^{(m)}} \left( y_i - \bar{y}_{Z_k^{(m)}} \right)^2,$$

donc

$$\sum_{i \in Z_k^{(m)}} (y_i - \beta_k)^2 \text{ est minimale en } \beta_k = \bar{y}_{Z_k^{(m)}}.$$



## Arbre de décision : estimation des coefficients

### Classification binaire avec la perte logarithmique

Il s'agit de minimiser, pour  $\beta_k \in [0, 1]$ ,

$$\mathcal{J}(\beta_k) = \sum_{i \in Z_k^{(m)}} (-y_i \ln(\beta_k) - (1 - y_i) \ln(1 - \beta_k))$$

Soient :

- ▶  $n_k = \text{card} \left( Z_k^{(m)} \right)$ ,
- ▶  $n_k^{(1)} = \text{card} \left( i \in Z_k^{(m)} \text{ tel que } y_i = 1 \right)$ .

$$\mathcal{J}(\beta_k) = -n_k^{(1)} \ln(\beta_k) - (n_k - n_k^{(1)}) \ln(1 - \beta_k)$$

77/81

### Classification binaire avec la perte logarithmique (suite)


$$\nabla \mathcal{J}(\beta_k) = \frac{-n_k^{(1)}}{\beta_k} + \frac{n_k - n_k^{(1)}}{1 - \beta_k} \text{ s'annule en } \hat{\beta}_k = \frac{n_k^{(1)}}{n_k} \text{ avec}$$

- ▶  $\nabla \mathcal{J}(\beta_k) \leq 0$  pour  $\beta_k \leq \hat{\beta}_k$
- ▶  $\nabla \mathcal{J}(\beta_k) \geq 0$  pour  $\beta_k \geq \hat{\beta}_k$

⇒  $\mathcal{J}(\beta_k)$  est donc minimale en  $\hat{\beta}_k$



### Interprétation

Notez la « ressemblance » entre  $\hat{\beta}_k$  et la fonction de classification optimale  $h(x) = P(Y = 1 | X = x)$  obtenue  slide 12 :

$\hat{\beta}_k$  s'interprète comme la version empirique de  $h(x)$  pour  $x \in Z_k^{(m)}$ .

 retour au slide ??

78/81

# Modèle linéaire généralisé

## GLM = Generalized Linear Model

Il regroupe les modèles pour lesquels :

- ▶  $Y|X$  suit une loi issue d'une **famille exponentielle** :

$$f^{Y|X}(y|x) = C(\eta)h(y)\exp(\eta y) \quad \text{avec } \eta = \eta(x).$$

- ▶  $g(\mathbb{E}_\beta(Y|X=x)) = \beta_0 + \beta^\top x$ .

**Vocabulaire.** La fonction  $g$  s'appelle **fonction de lien**.<sup>†</sup>

**Exemple.** Les lois de Bernoulli forment une famille exponentielle.

$$\begin{aligned} f(y) &= \theta^y (1-\theta)^{1-y} \\ &= (1-\theta) \exp\left(\ln\left(\frac{\theta}{1-\theta}\right)y\right) \quad \Rightarrow \eta = \ln\left(\frac{\theta}{1-\theta}\right) \end{aligned}$$

<sup>†</sup> Si  $N$  désigne l'ensemble des valeurs admissibles du paramètre  $\eta$ , on choisit le plus souvent pour  $g$  une bijection de  $N$  dans  $\mathbb{R}$ .

## Remarque : modèle linéaires généralisés (GLM)

Forme du modèle considéré

- ▶  $Y|X \sim \text{Ber}(\mathbb{E}_\beta(Y|X))$ ,
- ▶  $g(\mathbb{E}_\beta(Y|X)) = \beta_0 + \beta^\top X$ , avec  $g = \text{logit}$ .

⇒ cas particulier du **modèle linéaire généralisé (GLM)**  
( $g$  s'appelle la fonction de lien)

⇒ complément

Remarque : nous avons déjà rencontré un autre exemple de GLM

- ▶  $Y|X \sim \mathcal{N}(\mathbb{E}_\beta(Y|X), \sigma^2)$
- ▶  $g(\mathbb{E}_\beta(Y|X)) = \beta_0 + \beta^\top X$  avec  $g = \text{Id}$

Exemple :  $Y_i|X_i \stackrel{\text{iid}}{\sim} \text{Poisson}(\theta_i)$ , avec  $\ln \theta_i = \beta_0 + \beta_1 X_i$

Les lois de Poisson forment une famille exponentielle :

$$\begin{aligned} f(y) &= \exp(-\theta) \frac{\theta^y}{y!} \\ &= \frac{1}{y!} \exp(-\theta) \exp(\ln(\theta)y) \quad \Rightarrow \eta = \ln(\theta) \end{aligned}$$

