CentraleSupélec

Statistics and Learning

Lecturers (alphabetic order):

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

† Course coordinator

Lecture 7/9

Classification: logistic regression. Some models for supervised learning

Course objectives

- Classification using logistic regression
- Performance metrics for classifiers
- Prediction with decision trees
- Prediction with neural networks

Lecture outline

1 – Some general notions about classification

2 – Logistic regression [classification]

3 – Decision trees [regression + classification]

4 – Neural networks [regression + classification]

5 – Standard exercices (with solutions)

6 – Appendices

Lecture outline

1 – Some general notions about classification
- 1.1 – Introduction
- 1.2 – Loss functions and associated optimal classifiers
- 1.3 – Performance metrics

2 – Logistic regression [classification]

3 – Decision trees [regression + classification]

4 – Neural networks [regression + classification]

5 – Standard exercices (with solutions)

6 – Appendices

Lecture outline

1 – Some general notions about classification
1.1 – Introduction
1.2 – Loss functions and associated optimal classifiers
1.3 – Performance metrics

2 – Logistic regression [classification]

3 – Decision trees [regression + classification]

4 – Neural networks [regression + classification]

5 – Standard exercices (with solutions)

6 – Appendices

Mathematical framework and objectives

Notations

- $(X_1, Y_1), \ldots, (X_n, Y_n) \overset{\text{iid}}{\sim} \mathrm{P}^{X,Y}$
- $\mathrm{P}^{X,Y}$: unknown distribution on $\mathcal{X} \times \mathcal{Y}$
- $\mathcal{X} \subset \mathbb{R}^p, \quad \mathcal{Y} = \{0, 1, \ldots, K-1\}$
- unless stated otherwise: $K = 2$ (binary classification)

Objectives

Construct a (good) prediction function $h : x \mapsto \{0, 1\}$.

Synonyms: classification function, or "classifier".

Objectives of this section

- introduction to the logistic regression method
- definition of relevant risk measures for classification

Mathematical framework and objectives

Notations

- $(X_1, Y_1), \ldots, (X_n, Y_n) \overset{\text{iid}}{\sim} \mathrm{P}^{X,Y}$
- $\mathrm{P}^{X,Y}$: unknown distribution on $\mathcal{X} \times \mathcal{Y}$
- $\mathcal{X} \subset \mathbb{R}^p, \quad \mathcal{Y} = \{0, 1, \ldots, K-1\}$
- unless stated otherwise: $K = 2$ (binary classification)

Objectives

Construct a (good) prediction function $h : x \mapsto \{0, 1\}$.

Synonyms: classification function, or "classifier".

Objectives of this section

- introduction to the logistic regression method
- definition of relevant risk measures for classification

Mathematical framework and objectives

Notations

- $(X_1, Y_1), \ldots, (X_n, Y_n) \overset{\text{iid}}{\sim} \mathrm{P}^{X,Y}$
- $\mathrm{P}^{X,Y}$: unknown distribution on $\mathcal{X} \times \mathcal{Y}$
- $\mathcal{X} \subset \mathbb{R}^p, \quad \mathcal{Y} = \{0, 1, \ldots, K-1\}$
- unless stated otherwise: $K = 2$ (binary classification)

Objectives

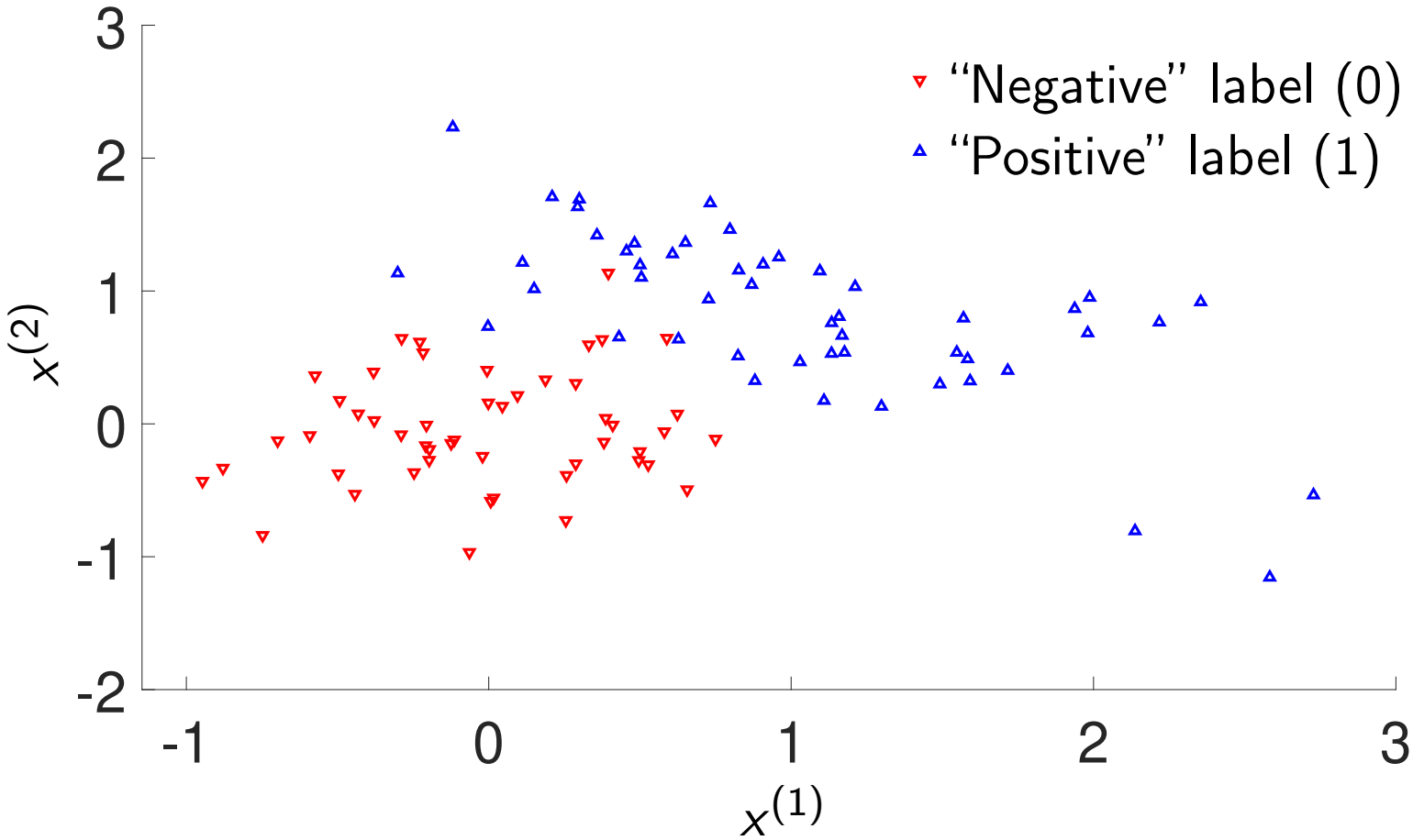
Construct a (good) prediction function $h : x \mapsto \{0, 1\}$.

Synonyms: classification function, or "classifier".

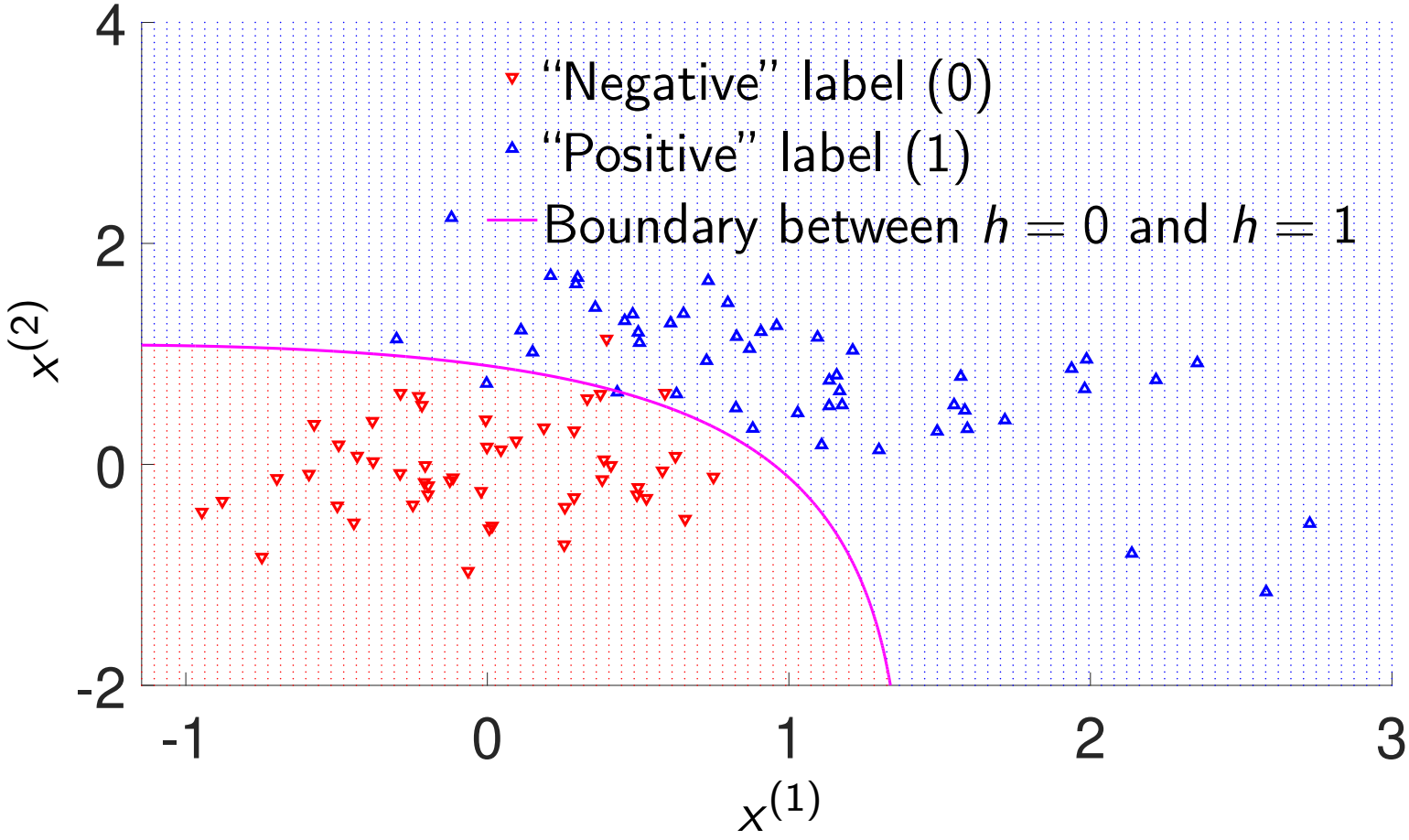
Objectives of this section

- introduction to the logistic regression method
- definition of relevant risk measures for classification

Example with two explanatory variables ($p = 2$)



A taste of things to come: a possible classifier



Lecture outline

1 – Some general notions about classification
- 1.1 – Introduction
- 1.2 – Loss functions and associated optimal classifiers
- 1.3 – Performance metrics

2 – Logistic regression [classification]

3 – Decision trees [regression + classification]

4 – Neural networks [regression + classification]

5 – Standard exercices (with solutions)

6 – Appendices

Reminder: loss function and risk

Definition: risk (generalization error)

Given a loss function $L : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_+$ and a prediction function $h : \mathcal{X} \to \mathcal{Y}$, the risk, or generalization error, is defined as :

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

where the expectation is with respect to (X, Y).

⚠ This risk depends on the unknown distribution:

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

Reminder: loss function and risk

Definition: risk (generalization error)

Given a loss function $L : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_+$ and a prediction function $h : \mathcal{X} \to \mathcal{Y}$, the risk, or generalization error, is defined as :

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

where the expectation is with respect to (X, Y).

⚠ This risk depends on the unknown distribution:

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

Soft and hard classifiers

Consider a problem with two classes: $\mathcal{Y} = \{0, 1\}$.

Definition: hard classifier

A hard classifier is a measurable function $h : \mathcal{X} \to \mathcal{Y}$ from the instance space to the label space.

Remark: special case of a prediction function, cf. Lecture 6.

Definition: soft classifier

A soft classifier is a measurable function $h : \mathcal{X} \to [0, 1]$ from the instance space to $[0, 1]$.

Given a soft classifier $h : \mathcal{X} \to [0, 1]$, we can construct a family of hard classifiers, of the form $x \mapsto \mathbb{1}_{h(x) \geq \delta}$, for $\delta \in [0, 1]$.

Soft and hard classifiers

Consider a problem with two classes: $\mathcal{Y} = \{0, 1\}$.

Definition: hard classifier

A hard classifier is a measurable function $h : \mathcal{X} \to \mathcal{Y}$ from the instance space to the label space.

Remark: special case of a prediction function, cf. Lecture 6.

Definition: soft classifier

A soft classifier is a measurable function $h : \mathcal{X} \to [0, 1]$ from the instance space to $[0, 1]$.

Given a soft classifier $h : \mathcal{X} \to [0, 1]$, we can construct a family of hard classifiers, of the form $x \mapsto \mathbb{1}_{h(x) \geq \delta}$, for $\delta \in [0, 1]$.

Soft and hard classifiers

Consider a problem with two classes: $\mathcal{Y} = \{0, 1\}$.

Definition: hard classifier

A hard classifier is a measurable function $h : \mathcal{X} \to \mathcal{Y}$ from the instance space to the label space.

Remark: special case of a prediction function, cf. Lecture 6.

Definition: soft classifier

A soft classifier is a measurable function $h : \mathcal{X} \to [0, 1]$ from the instance space to $[0, 1]$.

Given a soft classifier $h : \mathcal{X} \to [0, 1]$, we can construct a family of hard classifiers, of the form $x \mapsto \mathbb{1}_{h(x) \geq \delta}$, for $\delta \in [0, 1]$.

Commonly used loss functions

Definition: 0/1 loss for hard classification

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

- $R(h) = \mathbb{P}(Y \neq h(X))$ is the probability of misclassification.

Definition: Logarithmic loss for soft classification

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

Commonly used loss functions

Definition: 0/1 loss for hard classification

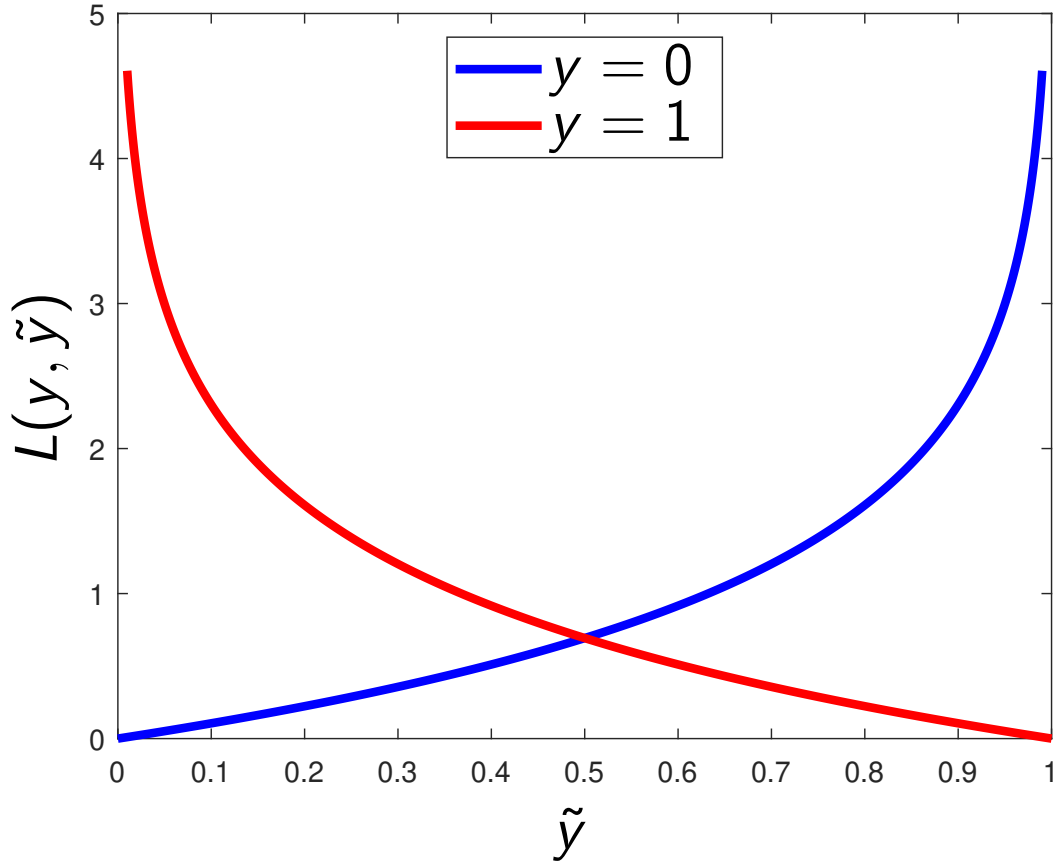
$$\begin{aligned} L : \mathcal{Y} \times \mathcal{Y} &\to \mathbb{R}_+ \\ (y, \tilde{y}) &\mapsto L(y, \tilde{y}) = \mathbb{1}_{y \neq \tilde{y}}. \end{aligned}$$

- $R(h) = \mathbb{P}(Y \neq h(X))$ is the probability of misclassification.

Definition: Logarithmic loss for soft classification

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

Commonly used loss functions (cont'd)



Remark: for both loss functions,

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

Optimal classification functions

Proposition

$h : \mathcal{X} \to \mathcal{Y}$ is optimal for the 0/1 loss iff, P^X-ae,

- $h(x) = 1$ when $\mathrm{P}(Y = 1 \mid X = x) > \frac{1}{2}$,
- $h(x) = 0$ when $\mathrm{P}(Y = 1 \mid X = x) < \frac{1}{2}$.

with the notation $\mathrm{P}(A \mid X = x) = \mathbb{E}(\mathbb{1}_A \mid X = x)$.

proof: see PC 7

For instance, $x \mapsto \mathbb{1}_{\mathrm{P}(Y=1|X=x) \geq \frac{1}{2}}$ is optimal.

Remark: a more general formula can be proved for an asymmetric loss ($L(0,1) \neq L(1,0)$). See PHC's lecture notes.

Optimal classification functions (cont'd)

Proposition

$h : \mathcal{X} \to [0, 1]$ is optimal for the logarithmic loss iff, P^X-ae,

$$h(x) \;=\; \mathrm{P}\left(Y = 1 \mid X = x\right).$$

proof: exercise 1

Remark: since Y takes its values in $\{0, 1\}$, we have:

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

☞ soft classification + logarithmic loss $\approx$ regression.

(Hence the name of the method that we are about to study!)

Optimal classification functions (cont'd)

Proposition

$h : \mathcal{X} \to [0, 1]$ is optimal for the logarithmic loss iff, P^X-ae,

$$h(x) \;=\; \mathrm{P}\,(Y = 1 \mid X = x)\,.$$

proof: exercise 1

Remark: since Y takes its values in $\{0, 1\}$, we have:

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

☞ soft classification + logarithmic loss $\approx$ regression.

(Hence the name of the method that we are about to study!)

Lecture outline

1 – Some general notions about classification
- 1.1 – Introduction
- 1.2 – Loss functions and associated optimal classifiers
- 1.3 – Performance metrics

2 – Logistic regression [classification]

3 – Decision trees [regression + classification]

4 – Neural networks [regression + classification]

5 – Standard exercices (with solutions)

6 – Appendices

Confusion matrix & associated definitions

	Truth Negative (N)	Truth Positive (P)
Prediction Negative	True Negative (TN)	False Negative (FN)
Prediction Positive	False Positive (FP)	True Positive (TP)

True Positive Rate

$$TPR = \frac{TP}{P} = \frac{TP}{TP + FN}$$

(also called sensitivity)

True Negative Rate

$$TNR = \frac{TN}{N} = \frac{TN}{TN + FP}$$

(also called specificity)

Confusion matrix & associated definitions

	Truth Negative (N)	Truth Positive (P)
Prediction Negative	True Negative (TN)	False Negative (FN)
Prediction Positive	False Positive (FP)	True Positive (TP)

True Positive Rate

$$TPR = \frac{TP}{P} = \frac{TP}{TP + FN}$$

(also called sensitivity)

True Negative Rate

$$TNR = \frac{TN}{N} = \frac{TN}{TN + FP}$$

(also called specificity)

Confusion matrix & associated definitions (cont'd)

Alternative terminology, from the field of signal processing:

- $1 - TPR$ is the miss rate (false negative rate)
- $1 - TNR$ is the false alarm rate (false positive rate)

Applications to soft classifiers

- Reminder: to any given soft classifier h, we can associate a family of hard classifiers

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

- The value of δ impacts the TNR/TPR trade-off
  - when $\delta \nearrow$, $TNR \nearrow$, and $TPR \searrow$

Confusion matrix & associated definitions (cont'd)

Alternative terminology, from the field of signal processing:

- $1 - TPR$ is the miss rate (false negative rate)
- $1 - TNR$ is the false alarm rate (false positive rate)

Applications to soft classifiers

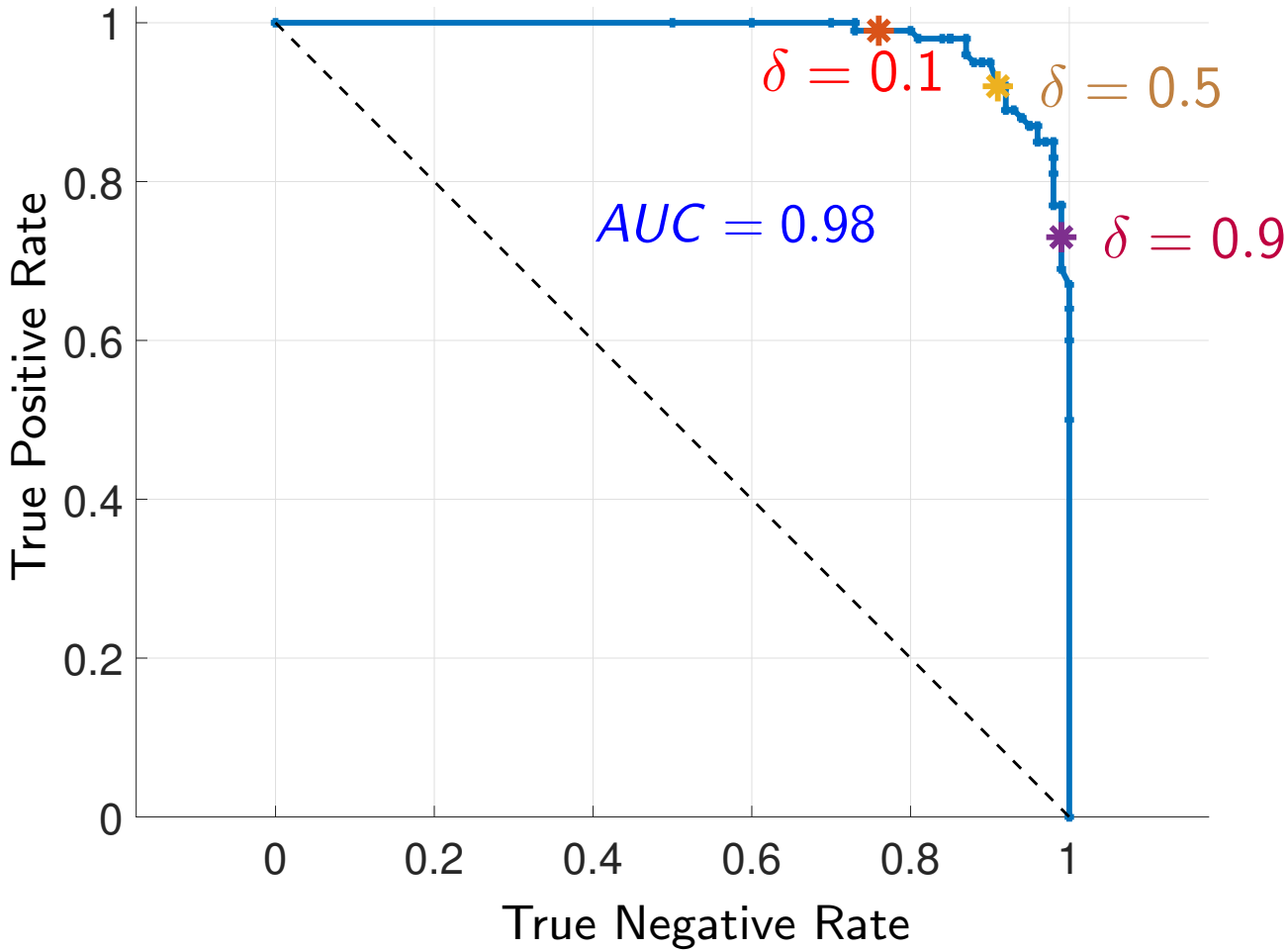
- Reminder: to any given soft classifier h, we can associate a family of hard classifiers

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

- The value of δ impacts the TNR/TPR trade-off
  - when $\delta \nearrow$, $TNR \nearrow$, and $TPR \searrow$

ROC curve (Receiver Operating Characteristic)

- a tool for decision support (choice of δ)
- a tool useful for classifier comparison
- associated definition: *AUC* = Area Under the Curve



Lecture outline

1 – Some general notions about classification

2 – Logistic regression [classification]
- 2.1 – A linear model for soft classification
- 2.2 – Training: selecting the coefficients
- 2.3 – Back to the introductory example
- 2.4 – Extensions

3 – Decision trees [regression + classification]

4 – Neural networks [regression + classification]

5 – Standard exercices (with solutions)

6 – Appendices

Lecture outline

1 – Some general notions about classification

2 – Logistic regression [classification]
- 2.1 – A linear model for soft classification
- 2.2 – Training: selecting the coefficients
- 2.3 – Back to the introductory example
- 2.4 – Extensions

3 – Decision trees [regression + classification]

4 – Neural networks [regression + classification]

5 – Standard exercices (with solutions)

6 – Appendices

A linear model for soft classification

Consider a (binary) classification problem

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

Choose the logarithmic loss:

- the goal is to approximate the optimal soft classifier

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

Logistic regression consists in using classifiers of the form

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

with $\beta_0 \in \mathbb{R}$, $\beta \in \mathbb{R}^p$, and $s(t) = e^t/(1 + e^t)$ the logistic function.

A linear model for soft classification

Consider a (binary) classification problem

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

Choose the logarithmic loss:

- the goal is to approximate the optimal soft classifier

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

Logistic regression consists in using classifiers of the form

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

with $\beta_0 \in \mathbb{R}$, $\beta \in \mathbb{R}^p$, and $s(t) = e^t/(1 + e^t)$ the logistic function.

A linear model for soft classification

Consider a (binary) classification problem

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

Choose the logarithmic loss:

- the goal is to approximate the optimal soft classifier

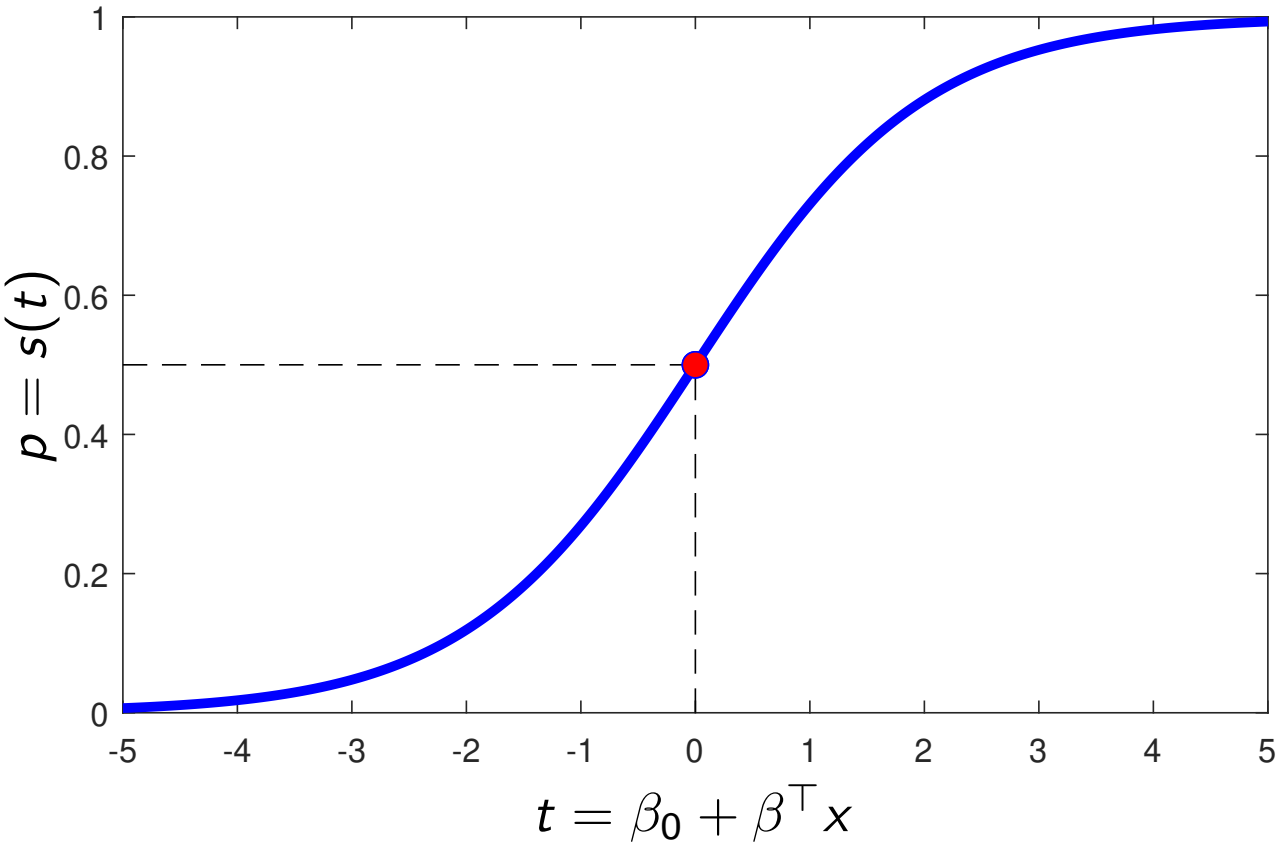
$$h^*(x) \;=\; \mathbb{P}(Y = 1 \mid X = x)\,.$$

Logistic regression consists in using classifiers of the form

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

with $\beta_0 \in \mathbb{R}$, $\beta \in \mathbb{R}^p$, and $s(t) = e^t/(1 + e^t)$ the logistic function.

The logistic function



➠ defines a correspondence: $\beta_0 + \beta^\top x \in \mathbb{R} \longleftrightarrow$ proba $p \in (0, 1)$

Also known as the sigmoid function.

A linear model for soft classification (cont'd)

Equivalently,

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

with

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

the logit function.

Properties

- The logistic function s is a strictly increasing, C^∞ bijection from $\mathbb{R}$ to $(0,1)$.
- The logit function is the inverse function: it is strictly increasing and C^∞ from $(0,1)$ to $\mathbb{R}$.

A linear model for soft classification (cont'd)

Equivalently,

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

with

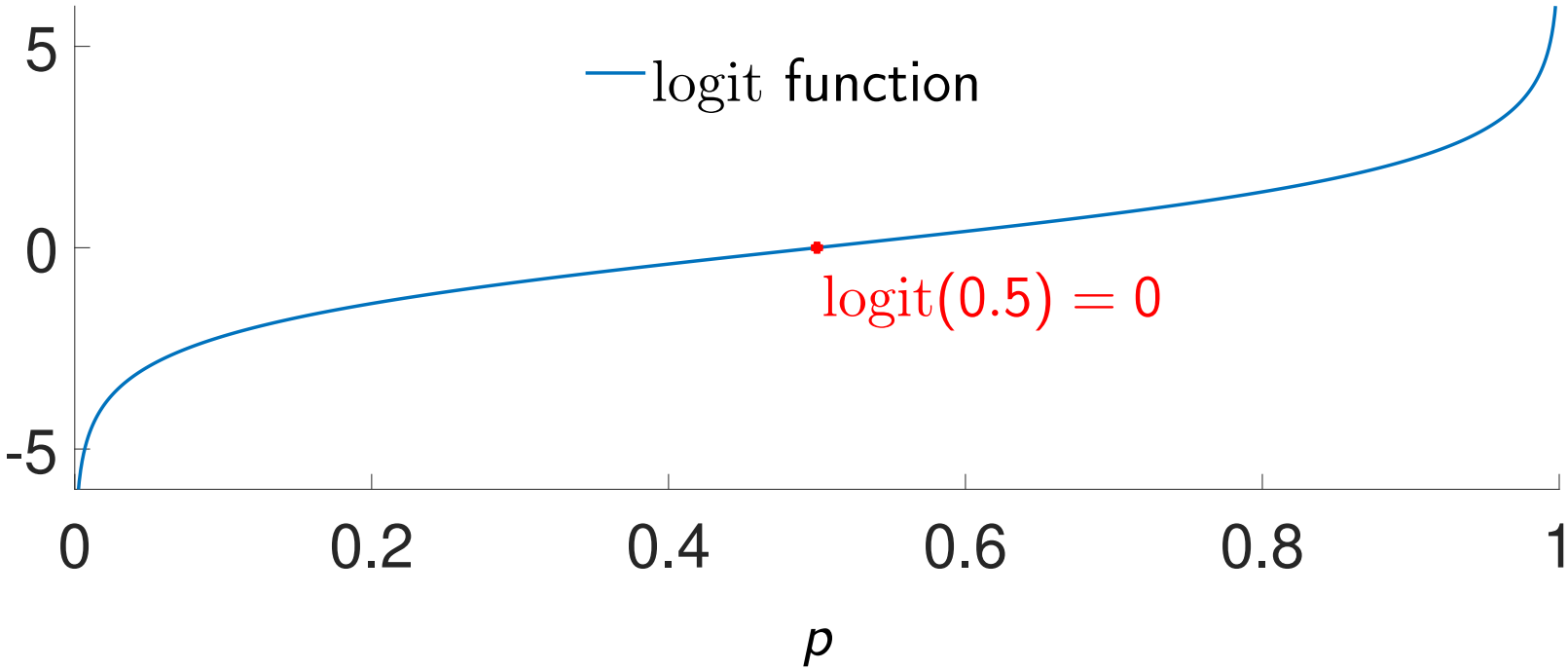
$$\begin{aligned} \text{logit}: \quad (0,1) &\to \mathbb{R} \\ p &\mapsto \ln\left(\frac{p}{1-p}\right) \end{aligned}$$

the logit function.

Properties

- The logistic function s is a strictly increasing, C^∞ bijection from $\mathbb{R}$ to $(0,1)$.
- The logit function is the inverse function: it is strictly increasing and C^∞ from $(0,1)$ to $\mathbb{R}$.

The logit function



➠ defines a correspondence: proba $p \in (0,1) \longleftrightarrow \beta_0 + \beta^\top x \in \mathbb{R}$

From soft to hard classification

Given a soft classifier of the form

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

and a decision threshold $\delta \in [0,1]$, we set:

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

☞ h_δ separates the classes in $\mathcal{X}$ by an affine hyperplane:

$$h_\delta(x) = 1 \quad \Longleftrightarrow \quad \beta_0 + \beta^\top x \geq \mathrm{logit}(\delta)$$

For the 0/1 loss, the value $\delta = \frac{1}{2}$ is generally used.

From soft to hard classification

Given a soft classifier of the form

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

and a decision threshold $\delta \in [0,1]$, we set:

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

☞ h_δ separates the classes in $\mathcal{X}$ by an affine hyperplane:

$$h_\delta(x) = 1 \quad \Longleftrightarrow \quad \beta_0 + \beta^\top x \geq \mathrm{logit}(\delta)$$

For the 0/1 loss, the value $\delta = \frac{1}{2}$ is generally used.

Lecture outline

1 – Some general notions about classification

2 – Logistic regression [classification]
- 2.1 – A linear model for soft classification
- 2.2 – Training: selecting the coefficients
- 2.3 – Back to the introductory example
- 2.4 – Extensions

3 – Decision trees [regression + classification]

4 – Neural networks [regression + classification]

5 – Standard exercices (with solutions)

6 – Appendices

Minimization of the empirical risk

Simplification of notations: $x \to \begin{pmatrix} 1 \\ x \end{pmatrix}$ and $\beta \to \begin{pmatrix} \beta_0 \\ \beta \end{pmatrix}$

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

The parameter β is selected using empirical risk minimization.:

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

where L denotes the logarithmic loss.

Minimization of the empirical risk

Simplification of notations: $x \to \begin{pmatrix} 1 \\ x \end{pmatrix}$ and $\beta \to \begin{pmatrix} \beta_0 \\ \beta \end{pmatrix}$

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

The parameter β is selected using empirical risk minimization.:

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

where L denotes the logarithmic loss.

Minimization of the empirical risk (cont'd)

Equivalence between empirical risk minimization and MLE

$$\sum_{i=1}^{n} L(y_i, h(x_i)) \;=\; -\ln\Big(\underbrace{\prod_{i=1}^{n} (h(x_i))^{y_i} (1 - h(x_i))^{1-y_i}}_{} \Big)$$

Interpretation: $\hat{\beta}$ is the MLE of the parametric model

$$Y_i | X_i \overset{\text{iid}}{\sim} \mathrm{Ber}(h(X_i)), \qquad h(x) = \frac{\exp\left(\beta^\top x\right)}{1 + \exp\left(\beta^\top x\right)}, \qquad \beta \in \mathbb{R}^{p+1}.$$

Minimization of the empirical risk (cont'd)

Equivalence between empirical risk minimization and MLE

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

Interpretation: $\hat{\beta}$ is the MLE of the parametric model

$$Y_i | X_i \overset{\text{iid}}{\sim} \mathrm{Ber}(h(X_i)), \qquad h(x) = \frac{\exp\left(\beta^\top x\right)}{1 + \exp\left(\beta^\top x\right)}, \qquad \beta \in \mathbb{R}^{p+1}.$$

Log-likelihood

Log-likelihood (see PC)

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

Maximization of ℓ

Using a numerical optimization algorithm

➠ for instance, the Newton-Raphson algorithm

Log-likelihood

Log-likelihood (see PC)

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

Maximization of ℓ

Using a numerical optimization algorithm

➠ for instance, the Newton-Raphson algorithm

Log-likelihood

Log-likelihood (see PC)

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

Maximization of ℓ

Using a numerical optimization algorithm

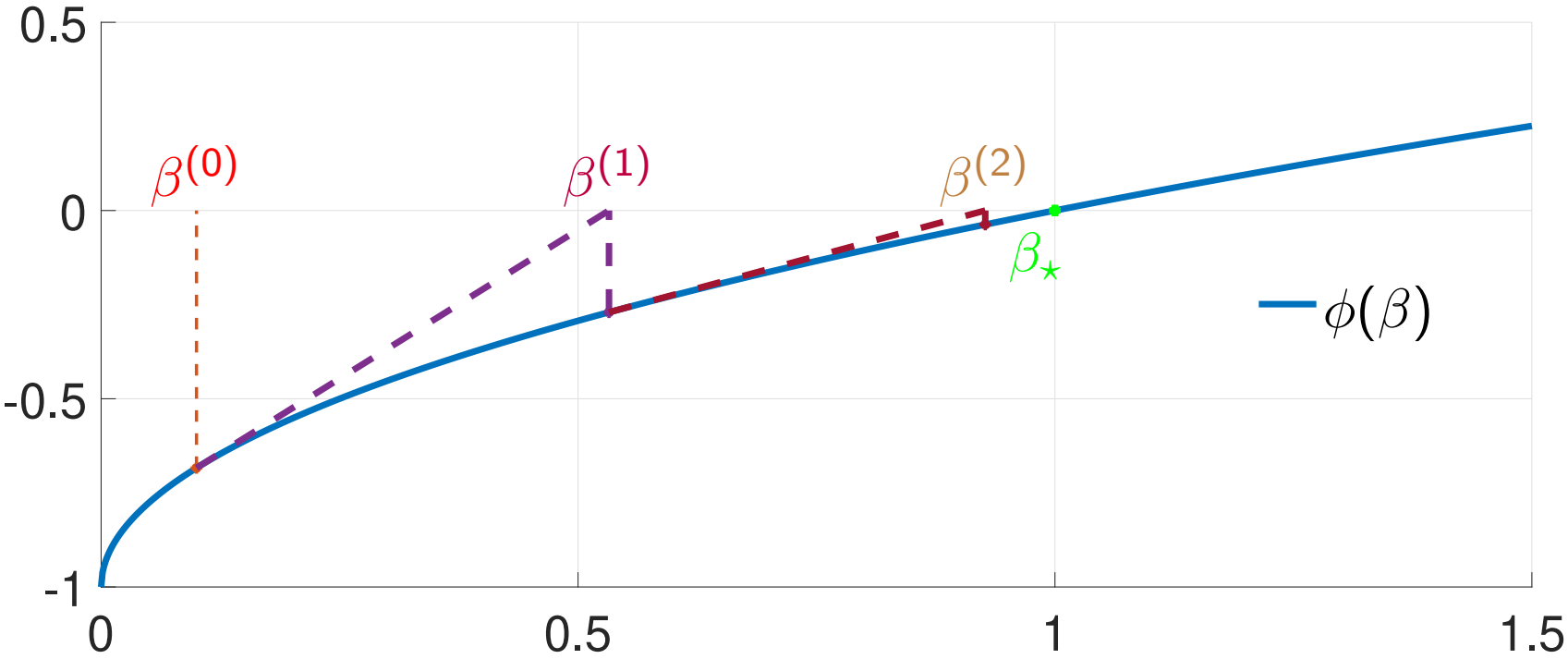
➠ for instance, the Newton-Raphson algorithm

Reminder: Newton-Raphson algorithm in one dimension

Let $\phi : \mathbb{R} \to \mathbb{R}$. We want β that satisfies $\phi(\beta) = 0$

Newton-Raphson algorithm is iterative:

- initialization: $\beta^{(0)}$
- iteration: $\beta^{(k+1)} = \beta^{(k)} - \dfrac{\phi\left(\beta^{(k)}\right)}{\phi'\left(\beta^{(k)}\right)}$



Maximization of ℓ using the Newton-Raphson method

Same algorithm but now in dimension $p+1$, with:

- $\phi \to \nabla_\beta \ell$
- $\phi' \to \nabla^2_\beta \ell$

The iteration follows:

$$\beta^{(k+1)} = \beta^{(k)} - \left[\nabla^2_\beta \ell\left(\beta^{(k)}\right)\right]^{-1} \nabla_\beta \ell\left(\beta^{(k)}\right)$$

Under the following conditions:

- $\nabla^2_\beta \ell\,(.)$ is Lipschitz continuous,
- $\nabla^2_\beta \ell\left(\beta^{(0)}\right)$ is invertible
- $h_0 = \left[\nabla^2_\beta \ell\left(\beta^{(0)}\right)\right]^{-1} \nabla_\beta \ell\left(\beta^{(0)}\right)$ small enough†,

the algorithm converges to a point $\beta^\star$ such that $\nabla_\beta \ell\,(\beta^\star) = 0$.

† cf. "Kantorovich theorem" on wikipedia for a more precise statement

Maximization of ℓ using the Newton-Raphson method

Same algorithm but now in dimension $p+1$, with:

- $\phi \to \nabla_\beta \ell$
- $\phi' \to \nabla^2_\beta \ell$

The iteration follows:

$$\beta^{(k+1)} = \beta^{(k)} - \left[\nabla^2_\beta \ell\left(\beta^{(k)}\right)\right]^{-1} \nabla_\beta \ell\left(\beta^{(k)}\right)$$

Under the following conditions:

- $\nabla^2_\beta \ell(.)$ is Lipschitz continuous,
- $\nabla^2_\beta \ell\left(\beta^{(0)}\right)$ is invertible
- $h_0 = \left[\nabla^2_\beta \ell\left(\beta^{(0)}\right)\right]^{-1} \nabla_\beta \ell\left(\beta^{(0)}\right)$ small enough†,

the algorithm converges to a point $\beta^\star$ such that $\nabla_\beta \ell(\beta^\star) = 0$.

† cf. "Kantorovich theorem" on wikipedia for a more precise statement

Maximization of ℓ using the Newton-Raphson method

Same algorithm but now in dimension $p+1$, with:

- $\phi \to \nabla_\beta \ell$
- $\phi' \to \nabla^2_\beta \ell$

The iteration follows:

$$\beta^{(k+1)} = \beta^{(k)} - \left[\nabla^2_\beta \ell\left(\beta^{(k)}\right)\right]^{-1} \nabla_\beta \ell\left(\beta^{(k)}\right)$$

Under the following conditions:

- $\nabla^2_\beta \ell\,(.)$ is Lipschitz continuous,
- $\nabla^2_\beta \ell\left(\beta^{(0)}\right)$ is invertible
- $h_0 = \left[\nabla^2_\beta \ell\left(\beta^{(0)}\right)\right]^{-1} \nabla_\beta \ell\left(\beta^{(0)}\right)$ small enough†,

the algorithm converges to a point $\beta^\star$ such that $\nabla_\beta \ell\,(\beta^\star) = 0$.

† cf. "Kantorovich theorem" on wikipedia for a more precise statement

Lecture outline

1 – Some general notions about classification

2 – Logistic regression [classification]
- 2.1 – A linear model for soft classification
- 2.2 – Training: selecting the coefficients
- 2.3 – Back to the introductory example
- 2.4 – Extensions

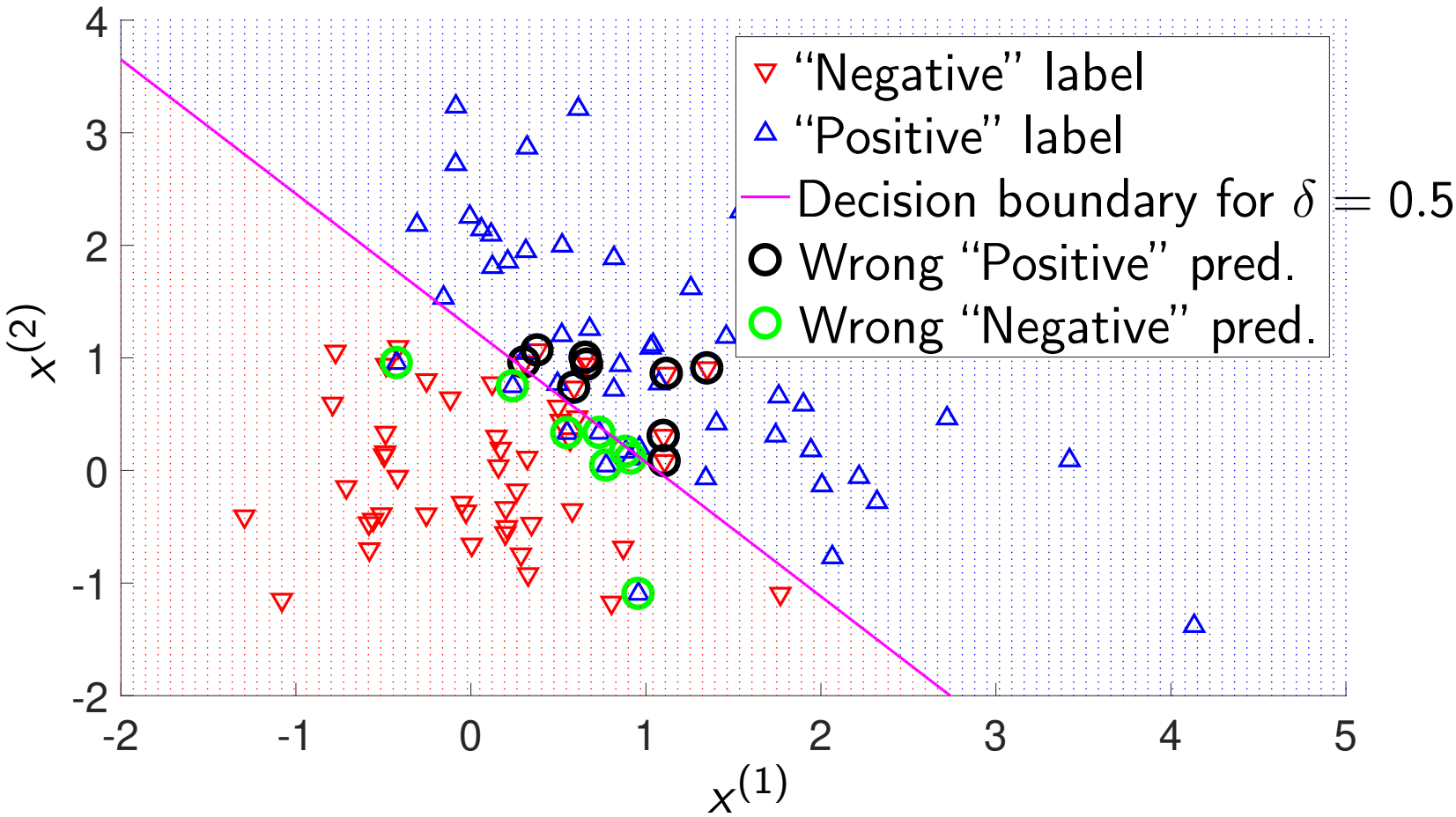
3 – Decision trees [regression + classification]

4 – Neural networks [regression + classification]

5 – Standard exercices (with solutions)

6 – Appendices

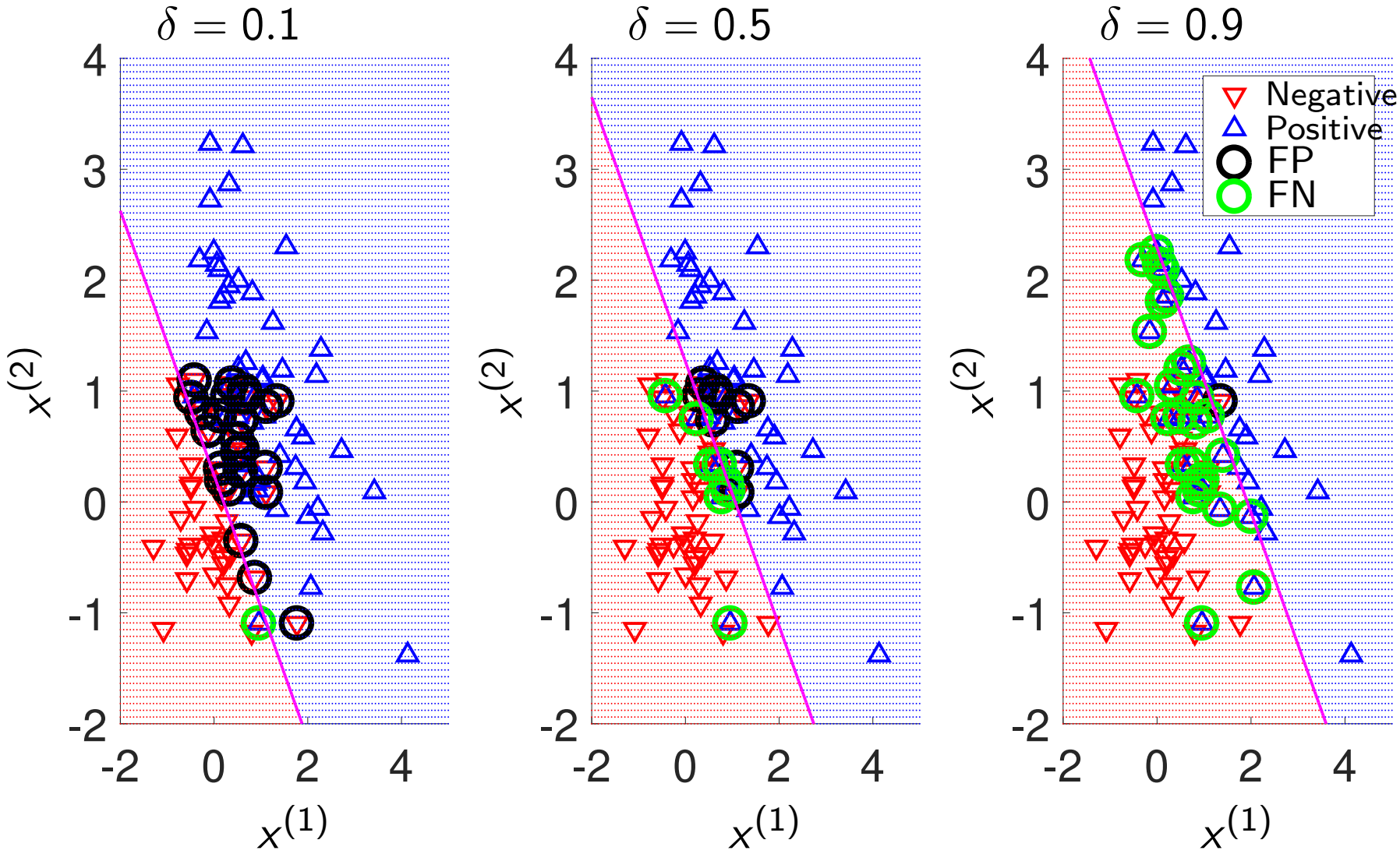
LR performed on the example with 2 explanatory variables



Prediction errors:

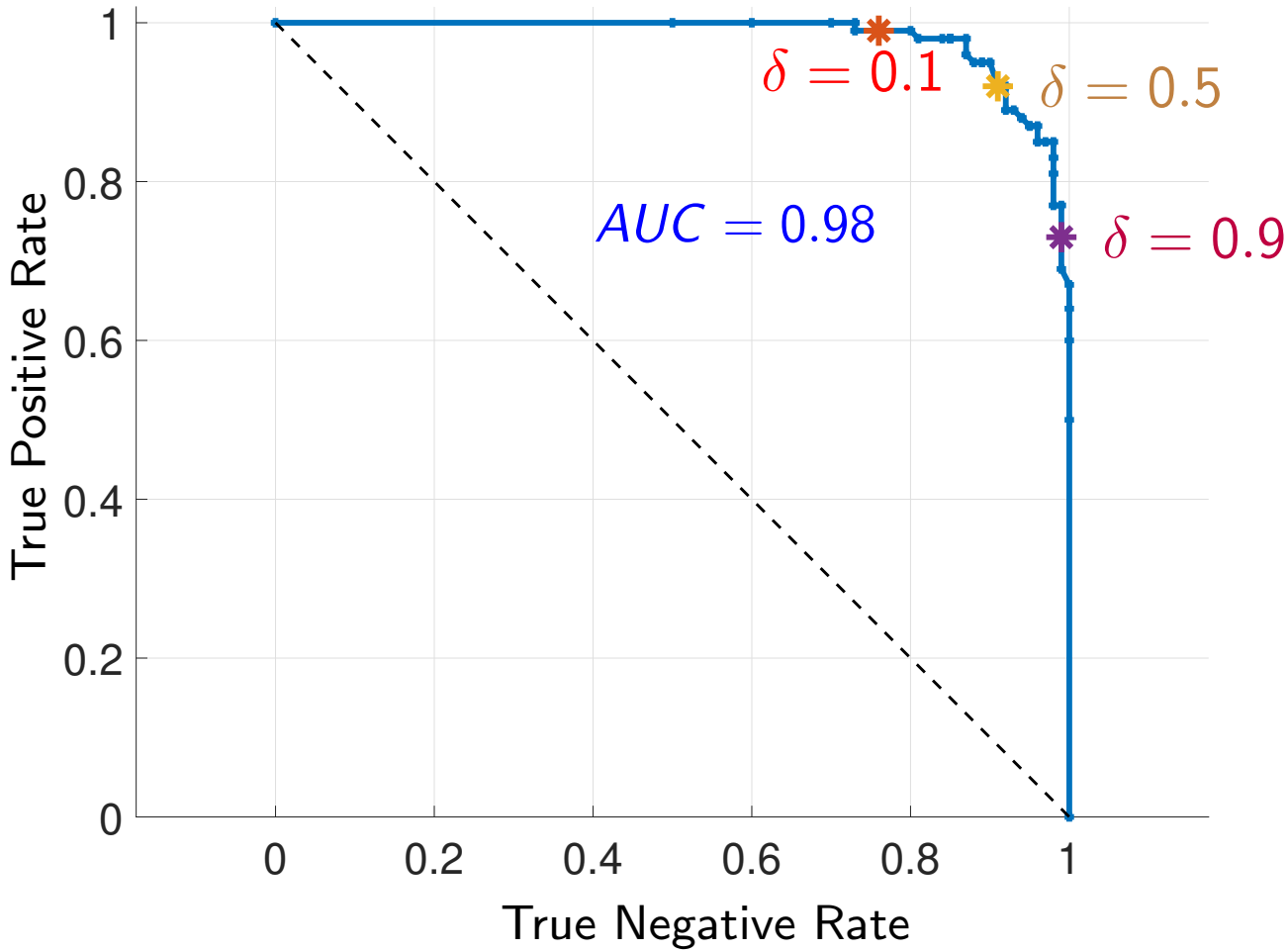
- "Negative" examples predicted as "Positive"
- "Positive" examples predicted as "Negative"

Influence of δ



ROC curve (Receiver Operating Characteristic)

- a tool for decision support (choice of δ)
- a tool useful for classifier comparison
- associated definition: *AUC* = Area Under the Curve



Lecture outline

1 – Some general notions about classification

2 – Logistic regression [classification]
- 2.1 – A linear model for soft classification
- 2.2 – Training: selecting the coefficients
- 2.3 – Back to the introductory example
- 2.4 – Extensions

3 – Decision trees [regression + classification]

4 – Neural networks [regression + classification]

5 – Standard exercices (with solutions)

6 – Appendices

Extension: large number of variables

How to handle the case where p is large

The log-likelihood is penalized:

- L_1 : $\hat{\beta} = \arg\max_\beta \left(\ell(\beta) - \lambda\|\beta\|^2\right)$
- L_2 : $\hat{\beta} = \arg\max_\beta \left(\ell(\beta) - \lambda\|\beta\|_1\right)$

➠ see Lecture 8

p is "large" if $p \gg n$, or even simply $p \approx n$

Extension: more than two classes

Multiclass classification

Let $\{0, 1, \ldots, K-1\}$ be the set of labels (classes), $K \geq 3$.

One class is chosen as the reference class and $K-1$ binary logistic regressions are performed (here class "0" was chosen):

$$\begin{cases} \ln\left(\frac{\mathrm{P}(Y=1|X=x)}{\mathrm{P}(Y=0|X=x)}\right) & = & \beta_{1,0} + \beta_1^\top x \\ \vdots & & \\ \ln\left(\frac{\mathrm{P}(Y=K-1|X=x)}{\mathrm{P}(Y=0|X=x)}\right) & = & \beta_{K-1,0} + \beta_{K-1}^\top x \end{cases}$$

Lecture outline

1 – Some general notions about classification

2 – Logistic regression [classification]

3 – Decision trees [regression + classification]
- 3.1 – Two introductory examples
- 3.2 – Recursive partitioning
- 3.3 – Prediction function

4 – Neural networks [regression + classification]

5 – Standard exercices (with solutions)

6 – Appendices

Lecture outline

1 – Some general notions about classification

2 – Logistic regression [classification]

3 – Decision trees [regression + classification]
- 3.1 – Two introductory examples
- 3.2 – Recursive partitioning
- 3.3 – Prediction function

4 – Neural networks [regression + classification]

5 – Standard exercices (with solutions)

6 – Appendices

Binary classification: spam detection

Data collected over 4601 e-mails

- explanatory variables: relative freq. of 57 of the most used words
- variable to be explained: label "Spam" or "Email"
  - ➠ categorical variable (binary in this example)

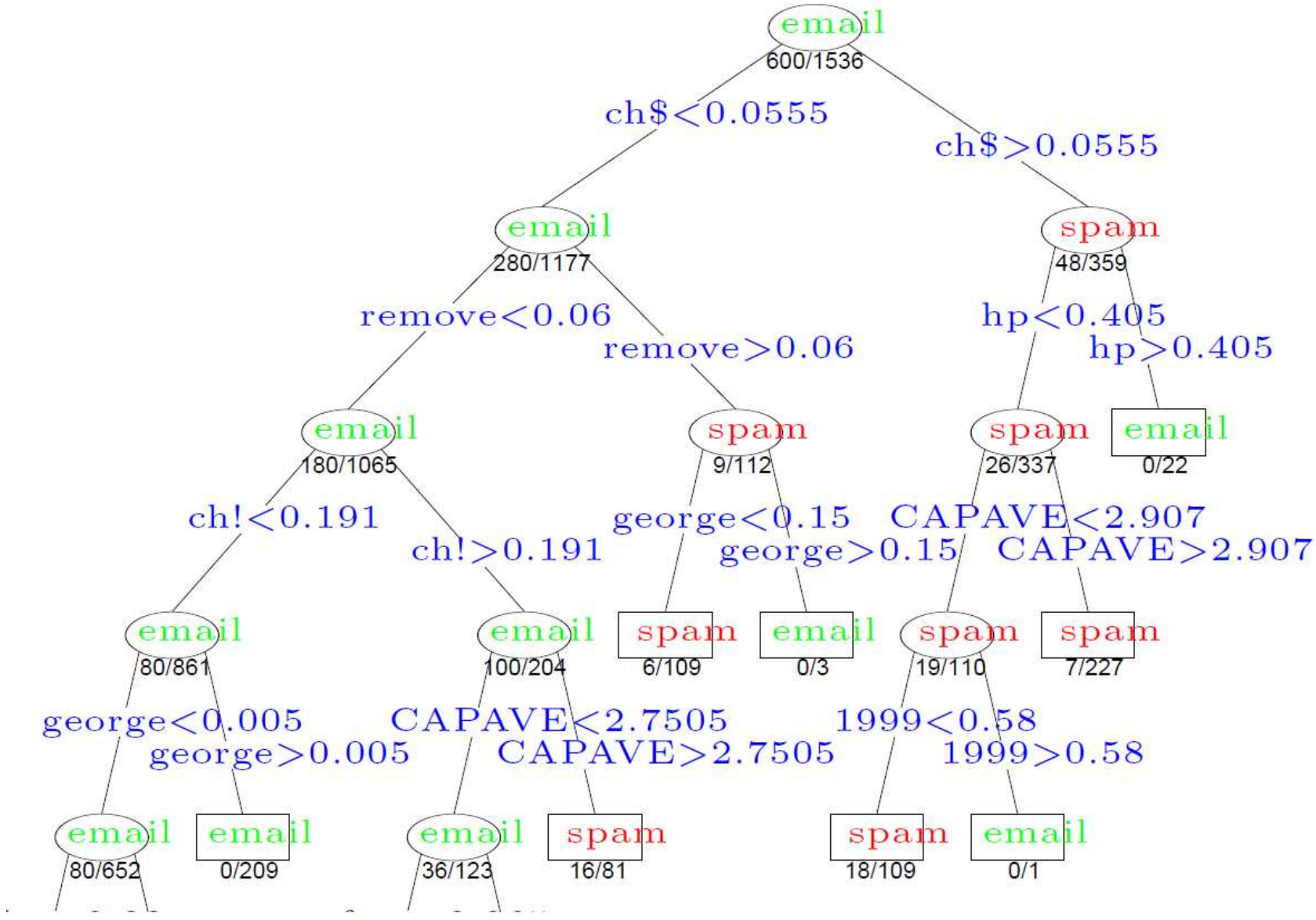
TABLE 1.1. *Average percentage of words or characters in an email message equal to the indicated word or character. We have chosen the words and characters showing the largest difference between* `spam` *and* `email`.

	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%.

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

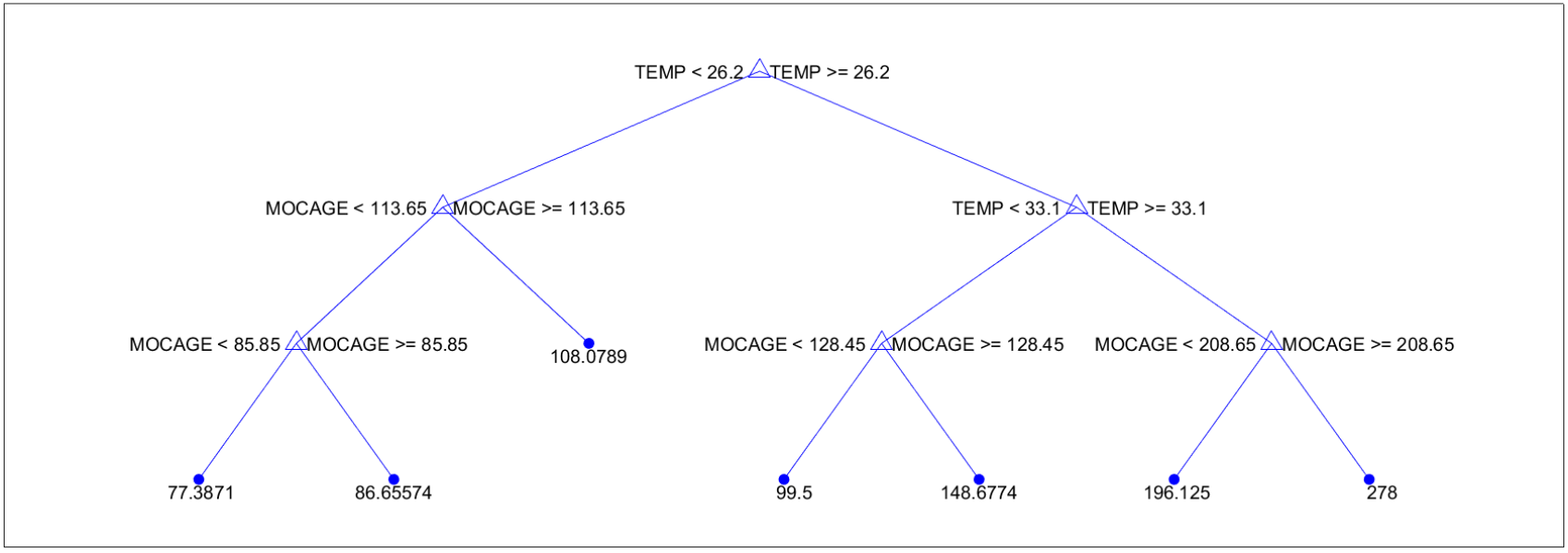
Source: The Elements of Statistical Learning, Springer (for next slide also)


email
600/1536
ch$<0.0555
ch$>0.0555
email
280/1177
spam
48/359
remove<0.06
remove>0.06
hp<0.405
hp>0.405
email
180/1065
spam
9/112
spam
26/337
email
0/22
ch!<0.191
ch!>0.191
george<0.15
george>0.15
CAPAVE<2.907
CAPAVE>2.907
email
80/861
email
100/204
spam
6/109
email
0/3
spam
19/110
spam
7/227
george<0.005
george>0.005
CAPAVE<2.7505
CAPAVE>2.7505
1999<0.58
1999>0.58
email
80/652
email
0/209
email
36/123
spam
16/81
spam
18/109
email
0/1

Regression tree: "Ozone" example

Simplified example (for the sake of visualization)

- predict variable 03 (quantitative variable)
- from variables MOCAGE and TEMP



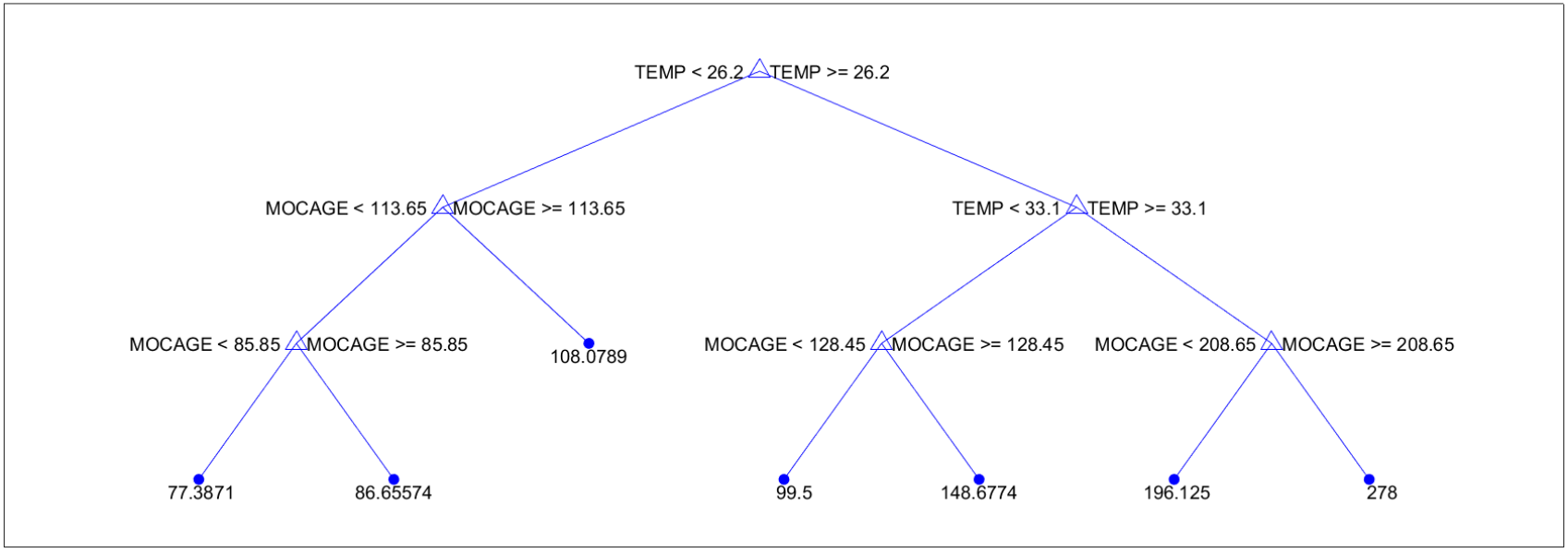
Vocabulary. When the variable to be explained is

- quantitative → regression tree
- categorical → classification tree

Regression tree: "Ozone" example

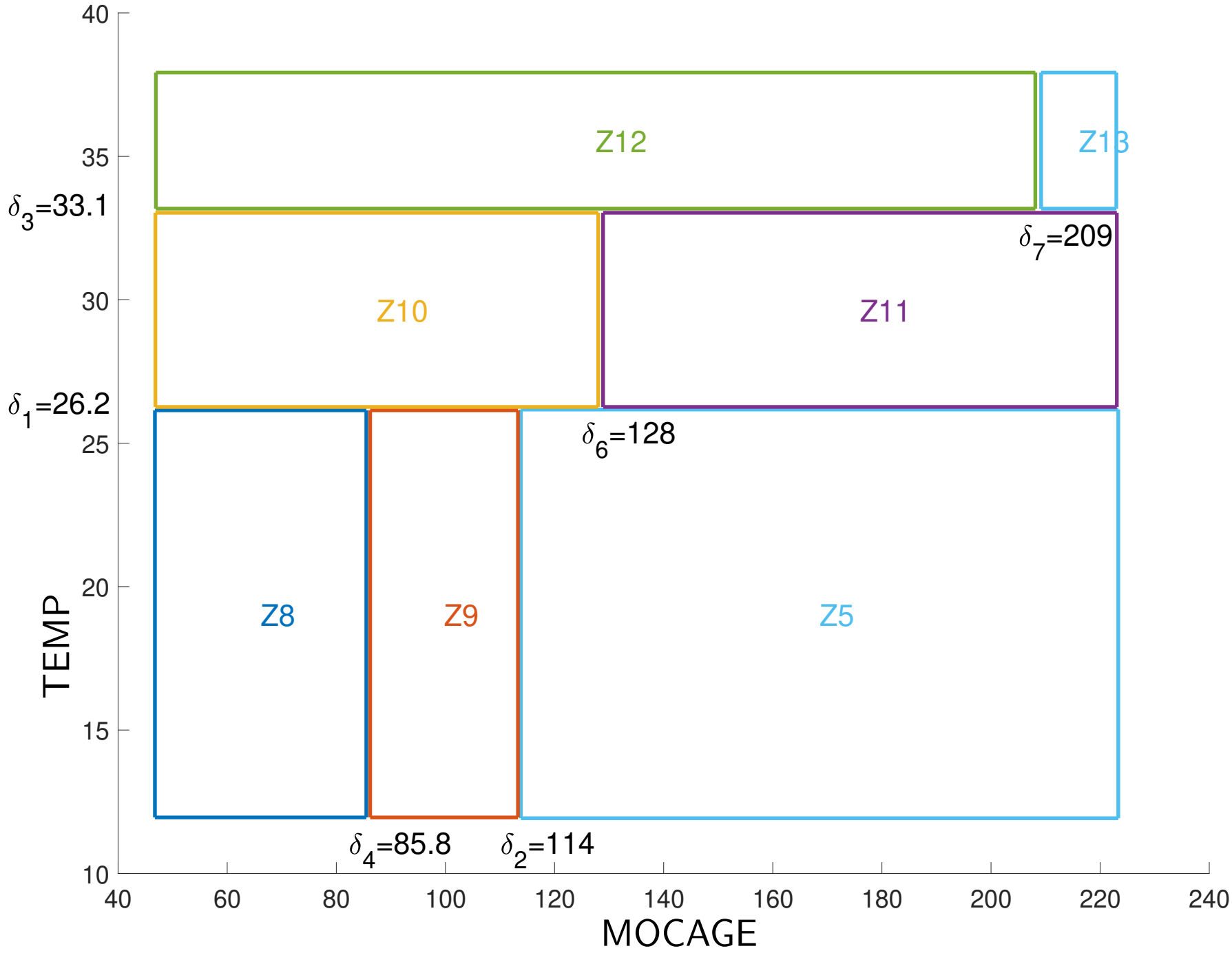
Simplified example (for the sake of visualization)

- predict variable 03 (quantitative variable)
- from variables MOCAGE and TEMP

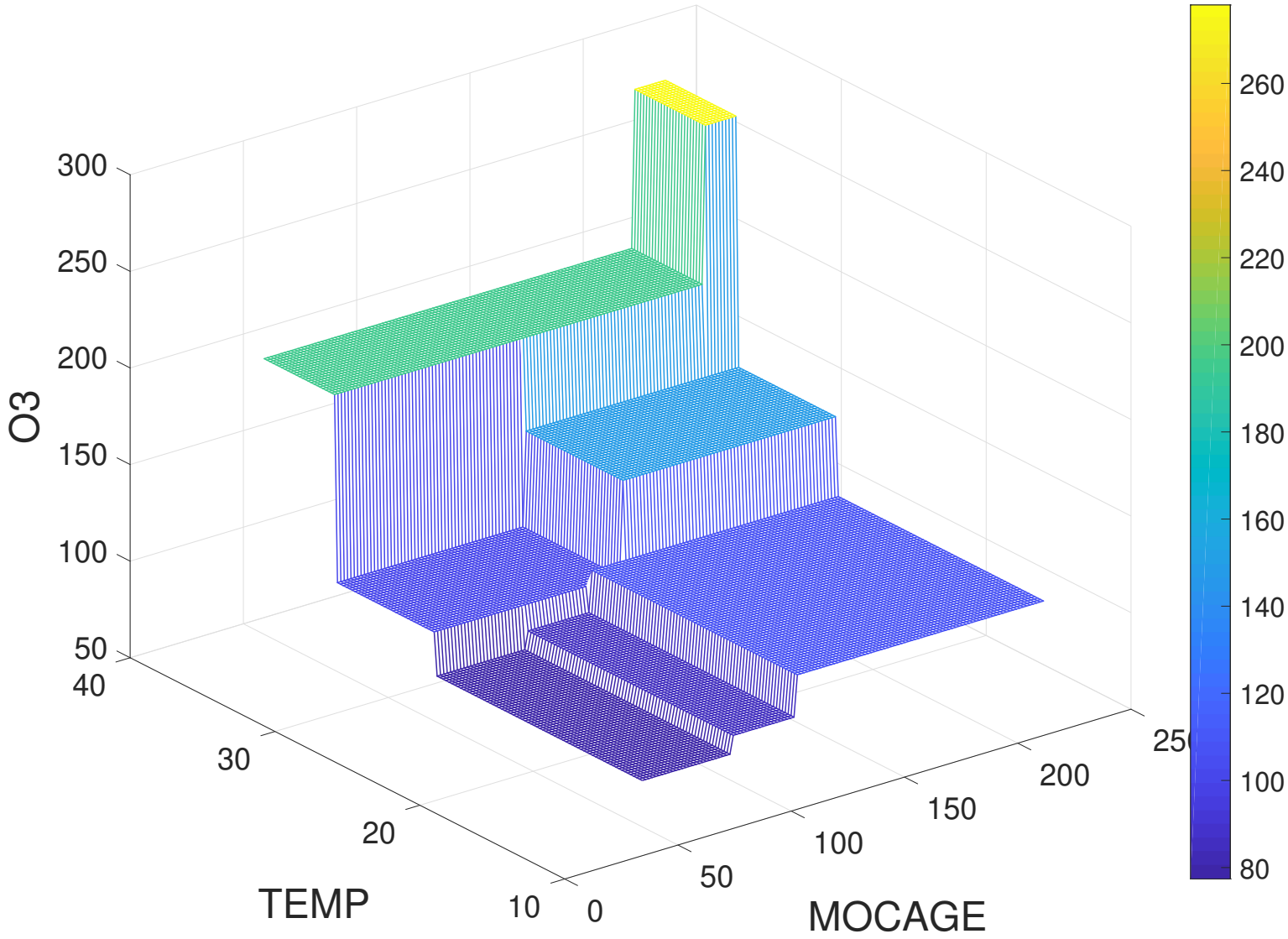


Vocabulary. When the variable to be explained is

- quantitative $\rightarrow$ regression tree
- categorical $\rightarrow$ classification tree


Z12
Z13
Z10
Z11
Z8
Z9
Z5
δ_3=33.1
δ_1=26.2
δ_7=209
δ_6=128
δ_4=85.8
δ_2=114
TEMP
MOCAGE
40
35
30
25
20
15
10
40
60
80
100
120
140
160
180
200
220
240

Regression tree: "Ozone" example



Lecture outline

1 – Some general notions about classification

2 – Logistic regression [classification]

3 – Decision trees [regression + classification]
- 3.1 – Two introductory examples
- 3.2 – Recursive partitioning
- 3.3 – Prediction function

4 – Neural networks [regression + classification]

5 – Standard exercices (with solutions)

6 – Appendices

Recursive partitioning: general principle

Objectives

Construct a partition of $\mathcal{X}$ from the data $(\underline{X}, \underline{Y})$.

Principle: iterative construction of a sequ. $(\mathcal{P}_m)_{m \geq 1}$ of partitions,

- $\mathcal{P}_m = \left\{ Z_1^{(m)}, \ldots, Z_m^{(m)} \right\}$, where partition $\mathcal{P}_m$ contains m subsets.

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

$\mathcal{P}_m \to \mathcal{P}_{m+1}$: split a subset $Z_{k_m}^{(m)}$ along one of the variables:

- $\tilde{Z}_1 = Z_{k_m}^{(m)} \cap \left\{ x \text{ such that } x^{(j_m)} \leq \delta_m \right\}$
- $\tilde{Z}_2 = Z_{k_m}^{(m)} \cap \left\{ x \text{ such that } x^{(j_m)} > \delta_m \right\}$

(the index j_m and the threshold δ_m still have to be specified)

Recursive partitioning: general principle

Objectives

Construct a partition of $\mathcal{X}$ from the data $(\underline{X}, \underline{Y})$.

Principle: iterative construction of a sequ. $(\mathcal{P}_m)_{m \geq 1}$ of partitions,

- $\mathcal{P}_m = \left\{ Z_1^{(m)}, \ldots, Z_m^{(m)} \right\}$, where partition $\mathcal{P}_m$ contains m subsets.

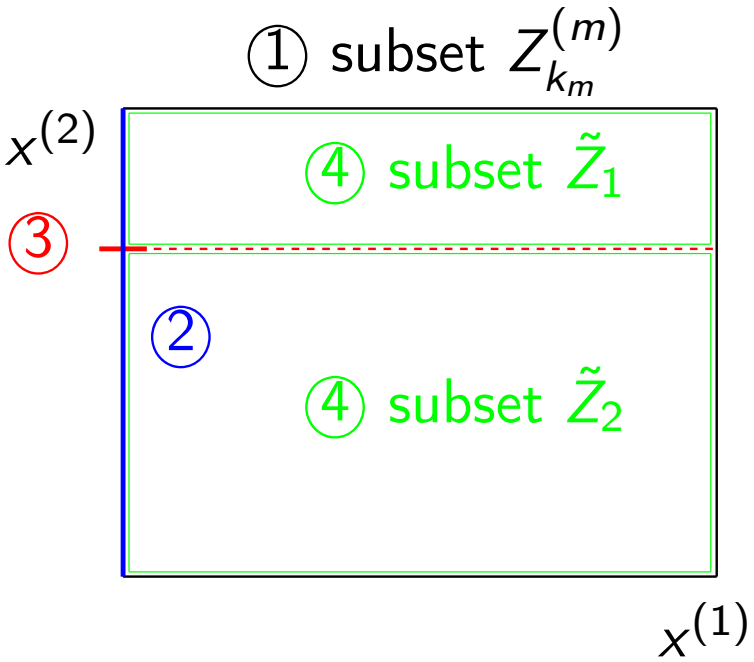
Initialization: $\mathcal{P}_1 = \{\mathcal{X}\}$.

$\mathcal{P}_m \to \mathcal{P}_{m+1}$: split a subset $Z_{k_m}^{(m)}$ along one of the variables:

- $\tilde{Z}_1 = Z_{k_m}^{(m)} \cap \left\{ x \text{ such that } x^{(j_m)} \leq \delta_m \right\}$
- $\tilde{Z}_2 = Z_{k_m}^{(m)} \cap \left\{ x \text{ such that } x^{(j_m)} > \delta_m \right\}$

(the index j_m and the threshold δ_m still have to be specified)

An example with $p = 2$



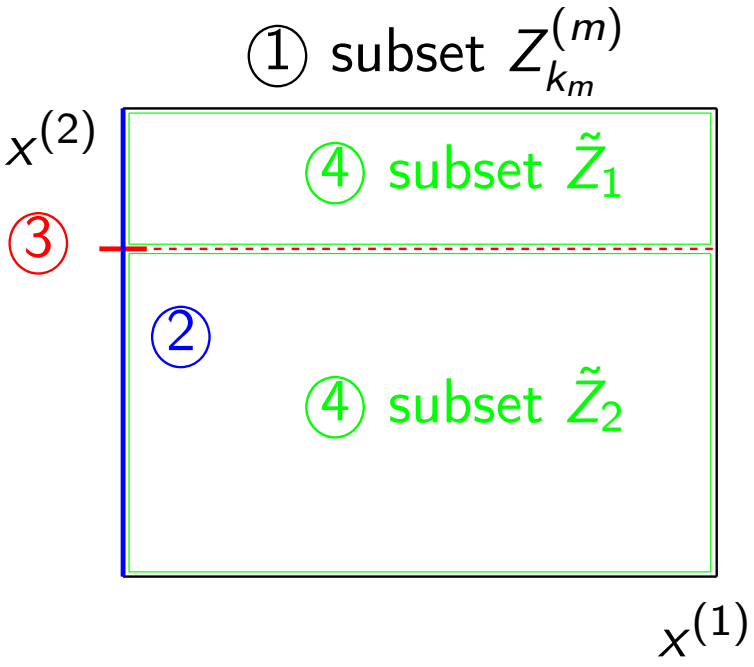
Iteration $\mathcal{P}_m \to \mathcal{P}_{m+1}$:

- ① subset $Z_{k_m}^{(m)} \in \mathcal{P}_m$
- ② variable $x^{(j_m)}$ (here $j_m = 2$)
- ③ threshold δ_m
- ④ construction of $\tilde{Z}_1$ and $\tilde{Z}_2$

After splitting $Z_{k_m}^{(m)}$, we get:

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

An example with $p = 2$



Iteration $\mathcal{P}_m \to \mathcal{P}_{m+1}$:

- ① subset $Z_{k_m}^{(m)} \in \mathcal{P}_m$
- ② variable $x^{(j_m)}$ (here $j_m = 2$)
- ③ threshold δ_m
- ④ construction of $\tilde{Z}_1$ and $\tilde{Z}_2$

After splitting $Z_{k_m}^{(m)}$, we get:

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

Choice of k_m, j_m and δ_m

Let $D(Z)$ be a measure of the heterogeneity of a subset Z.

Example (for a quantitative label y)

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

where $\bar{y}_Z$ is the empirical mean computed over Z.

k_m, j_m and δ_m are jointly chosen in such a way that

$D\left(Z_{k_m}^{(m)}\right) - D(\tilde{Z}_1) - D(\tilde{Z}_2)$ is as large as possible

➠ largest reduction of heterogeneity

(Recall that $\tilde{Z}_1$ and $\tilde{Z}_2$ are the subsets obtained by splitting $Z_{k_m}^{(m)}$)

Choice of k_m, j_m and δ_m

Let $D(Z)$ be a measure of the heterogeneity of a subset Z.

Example (for a quantitative label y)

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

where $\bar{y}_Z$ is the empirical mean computed over Z.

k_m, j_m and δ_m are jointly chosen in such a way that

$D\left(Z_{k_m}^{(m)}\right) - D(\tilde{Z}_1) - D(\tilde{Z}_2)$ is as large as possible

➠ largest reduction of heterogeneity

(Recall that $\tilde{Z}_1$ and $\tilde{Z}_2$ are the subsets obtained by splitting $Z_{k_m}^{(m)}$)

Lecture outline

1 – Some general notions about classification

2 – Logistic regression [classification]

3 – Decision trees [regression + classification]
- 3.1 – Two introductory examples
- 3.2 – Recursive partitioning
- 3.3 – Prediction function

4 – Neural networks [regression + classification]

5 – Standard exercices (with solutions)

6 – Appendices

Piecewise constant prediction function

Decision trees define a piecewise constant prediction fonction on the elements of the partition:

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

Remark: for a given partition, this is a linear model with respect to the m variables $\mathbb{1}_{Z_k^{(m)}}(x)$.

Piecewise constant prediction function

Decision trees define a piecewise constant prediction fonction on the elements of the partition:

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

Remark: for a given partition, this is a linear model with respect to the m variables $\mathbb{1}_{Z_k^{(m)}}(x)$.

Estimation of the coefficients

Principle: to estimate $\beta^{(m)} = \left(\beta_1^{(m)}, \dots \beta_m^{(m)}\right)$,

- choose a loss function $L(y, h_\beta(x))$,
- then minimize the empirical risk.

Simplification:

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

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

Estimation of the coefficients

Principle: to estimate $\beta^{(m)} = \left(\beta_1^{(m)}, \dots \beta_m^{(m)}\right)$,

- choose a loss function $L(y, h_\beta(x))$,
- then minimize the empirical risk.

Simplification:

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

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

Estimation of the coefficients

Principle: to estimate $\beta^{(m)} = \left(\beta_1^{(m)}, \dots \beta_m^{(m)}\right)$,

- choose a loss function $L(y, h_\beta(x))$,
- then minimize the empirical risk.

Simplification:

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

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

Two important special cases

Regression with the quadratic loss

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

Binary classification with the logarithmic loss

Soft classification:

$$\hat{\beta}_k^{(m)} = \text{argmin}_{\beta_k \in [0,1]} \sum_{i \in Z_k^{(m)}} (-y_i \ln(\beta_k) - (1 - y_i) \ln(1 - \beta_k))$$

$$= \frac{1}{\text{card}\left(Z_k^{(m)}\right)} \cdot \text{card}\left(i \in Z_k^{(m)} \text{ such that } y_i = 1\right)$$

Hard classification: threshold at $\delta = \frac{1}{2}$ (cf. logistic regression).

Two important special cases

Regression with the quadratic loss

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

Binary classification with the logarithmic loss

Soft classification:

$$\hat{\beta}_k^{(m)} = \text{argmin}_{\beta_k \in [0,1]} \sum_{i \in Z_k^{(m)}} (-y_i \ln(\beta_k) - (1 - y_i) \ln(1 - \beta_k))$$

$$= \frac{1}{\text{card}\left(Z_k^{(m)}\right)} \cdot \text{card}\left(i \in Z_k^{(m)} \text{ such that } y_i = 1\right)$$

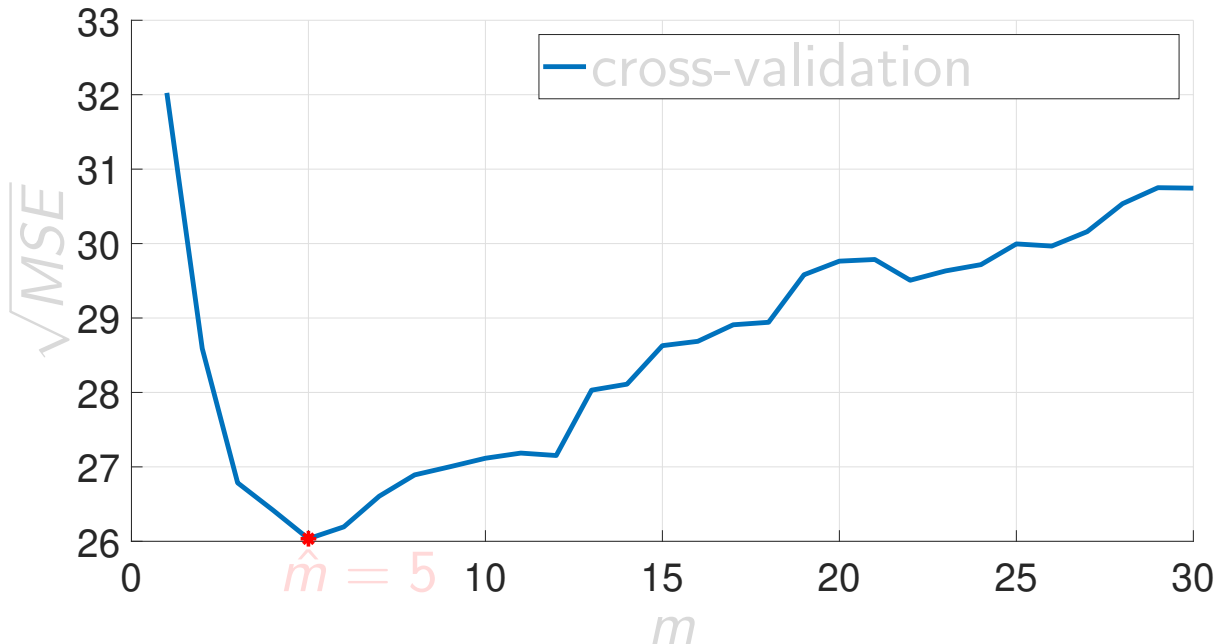
Hard classification: threshold at $\delta = \frac{1}{2}$ (cf. logistic regression).

Choosing the size m of the partition

- m can either be given beforehand ($\sim$ prior knowledge)
- or estimated by cross-validation.

"Ozone" example

- Regression of O3 with $p = 7$ explanatory variables
- m is chosen by leave-one-out cross-validation

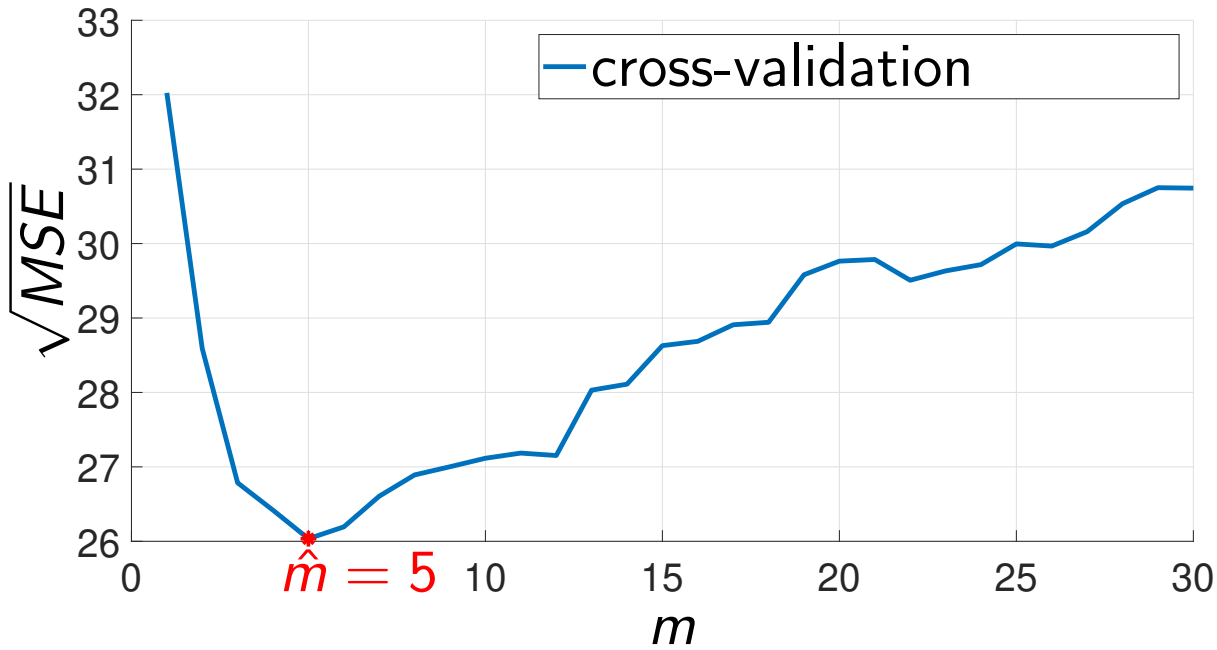


Choosing the size m of the partition

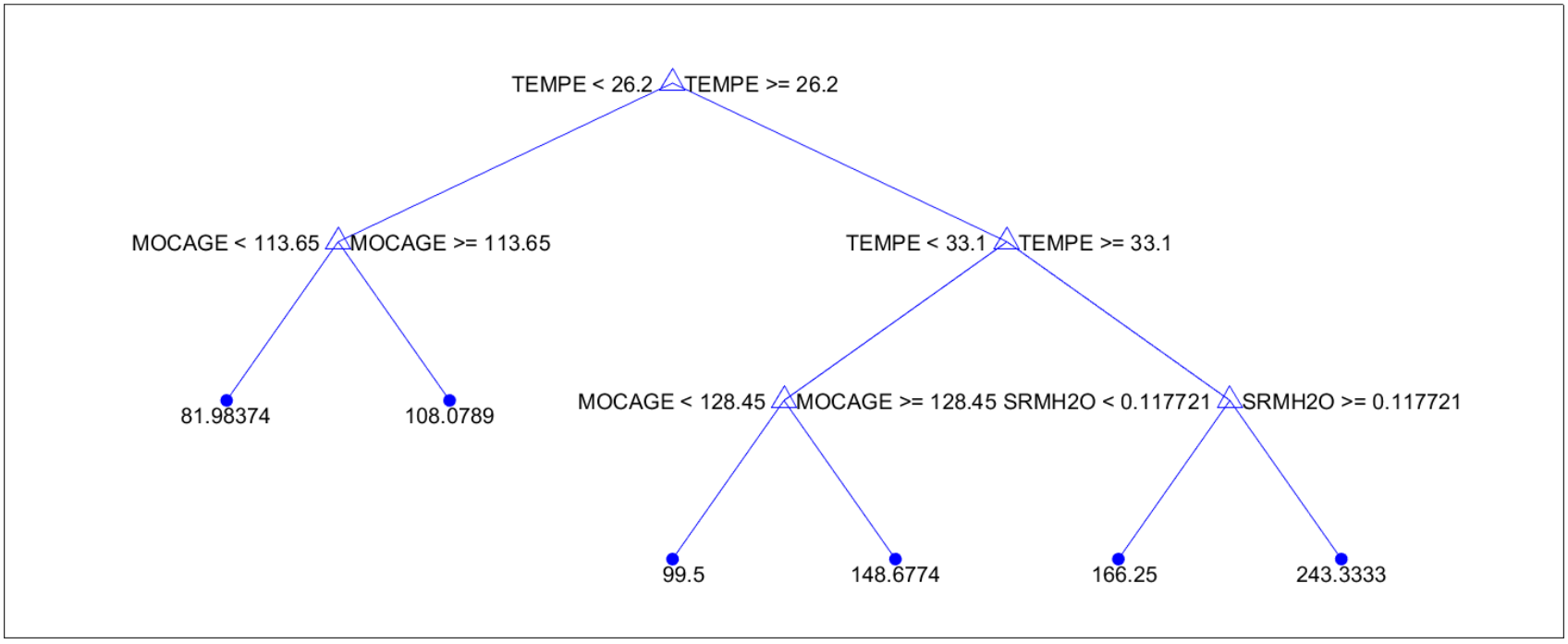
- m can either be given beforehand ($\sim$ prior knowledge)
- or estimated by cross-validation.

"Ozone" example

- Regression of O3 with $p = 7$ explanatory variables
- m is chosen by leave-one-out cross-validation



Regression tree: "Ozone" example




```
 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
```

More trees...

Disadvantages of decision trees

- high sensitivity to the sample $(\underline{x}, \underline{y})$
- piecewise constant prediction on each subset (by construct.)
  (not satisfactory if the optimal prediction function is smooth)

Extensions

- aggregation of decisions tree models
  - ➠ Random forests
- weighted sum of weak classifiers
  - ➠ Boosting (AdaBoost)

More trees...

Disadvantages of decision trees

- high sensitivity to the sample $(\underline{x}, \underline{y})$
- piecewise constant prediction on each subset (by construct.)
  (not satisfactory if the optimal prediction function is smooth)

Extensions

- aggregation of decisions tree models
  - ➠ Random forests
- weighted sum of weak classifiers
  - ➠ Boosting (AdaBoost)

Lecture outline

1 – Some general notions about classification

2 – Logistic regression [classification]

3 – Decision trees [regression + classification]

4 – Neural networks [regression + classification]
- 4.1 – Neurons
- 4.2 – Multi-layer perceptrons
- 4.3 – Example
- 4.4 – Other architectures

5 – Standard exercices (with solutions)

6 – Appendices

Lecture outline

1 – Some general notions about classification

2 – Logistic regression [classification]

3 – Decision trees [regression + classification]

4 – Neural networks [regression + classification]

4.1 – Neurons

4.2 – Multi-layer perceptrons

4.3 – Example

4.4 – Other architectures

5 – Standard exercices (with solutions)

6 – Appendices

The (multipolar) biological neuron: axons, dendrites...

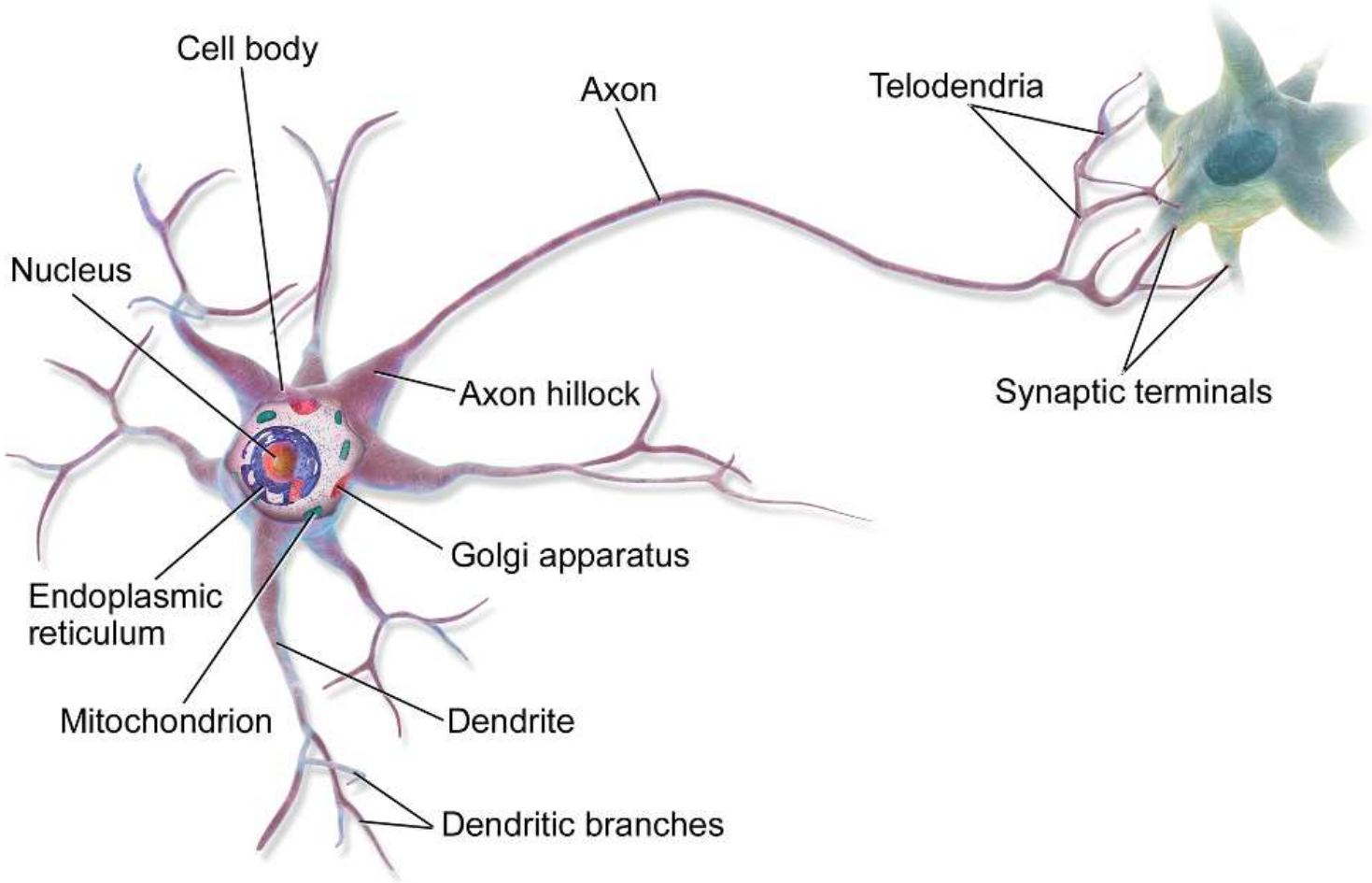


Image: Bruce Blaus, `https://commons.wikimedia.org`, CC BY 3.0

"A multipolar neuron is a type of neuron that possesses a single axon and many dendrites (and dendritic branches), allowing for the integration of a great deal of information from other neurons." (`https://fr.wikipedia.org/wiki/Neurone_multipolaire`)

The artificial neuron

Definition: neuron (McCulloch and Pitts, 1943)[†]

In statistical learning, a neuron with p variables (inputs) is a function, generally non-linear[†], of the form

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

where

- φ is an increasing $\mathbb{R} \to \mathbb{R}$ function;
- $w \in \mathbb{R}^{1\times p}$, and $b \in \mathbb{R}$.

Vocabulary

- φ: activation function,
- $w_1, \ldots, w_p$: weights,
- b: bias (nothing to do with the bias of an estimator).

[†] The original neuron of McCulloch & Pitts (1943) specifically used $\varphi = \text{sgn}$ as an activation function.

[‡] We will see later a situation where a linear neuron ($\varphi = \text{Id}$) is used.

The artificial neuron

Definition: neuron (McCulloch and Pitts, 1943)[†]

In statistical learning, a neuron with p variables (inputs) is a function, generally non-linear[†], of the form

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

where

- φ is an increasing $\mathbb{R} \to \mathbb{R}$ function;
- $w \in \mathbb{R}^{1\times p}$, and $b \in \mathbb{R}$.

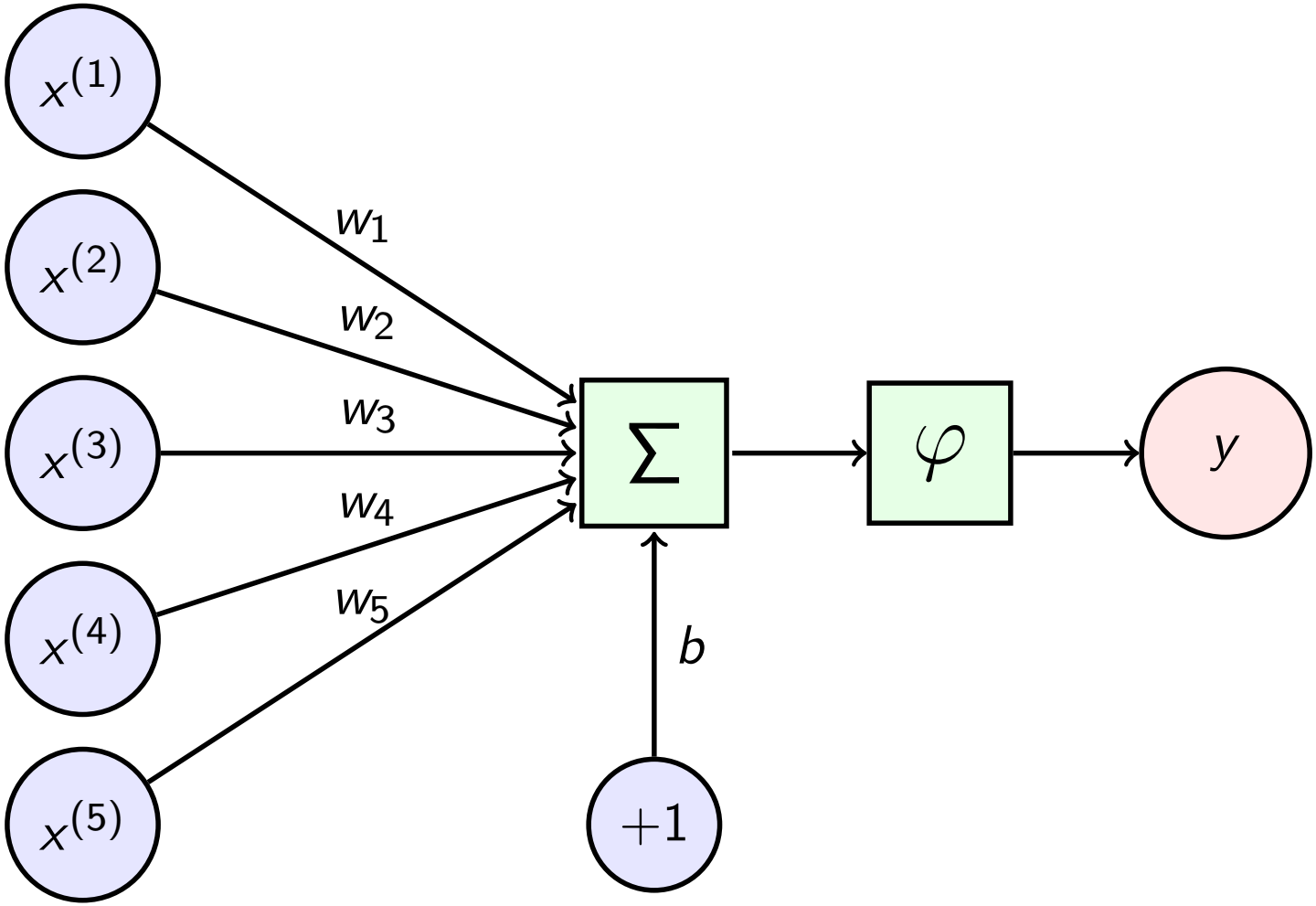
Vocabulary

- φ: activation function,
- $w_1, \ldots, w_p$: weights,
- b: bias (nothing to do with the bias of an estimator).

[†] The original neuron of McCulloch & Pitts (1943) specifically used $\varphi = \text{sgn}$ as an activation function.

[‡] We will see later a situation where a linear neuron ($\varphi = \text{Id}$) is used.

The artificial neuron: illustration ($p = 5$)



Activation functions

Discontinuous activation functions (not recommended[†]):

- Heaviside function: $\varphi(v) = \mathbb{1}_{v \geq 0}$, or
- sign function: $\varphi(v) = \operatorname{sgn}(v) = \mathbb{1}_{v>0} - \mathbb{1}_{v<0}$.

"S-shaped" functions, a.k.a. sigmoids:

- logistic[‡] : $\varphi(v) = \frac{1}{1+e^{-v}} = \frac{1}{2} + \frac{1}{2}\tanh\left(\frac{v}{2}\right)$, or
- tanh : $\varphi(v) = \tanh(v) = \frac{e^{v}-e^{-v}}{e^{v}+e^{-v}}$.

The ReLU (*Rectified Linear Unit*) function:

- $\varphi(v) = \max(0, v)$.

[†] Used in the oldest models, most notably the Rosenblatt's perceptron (1957), but abandonned since then because of their almost-everywhere zero gradient, which creates problems for optimization procedures.

[‡] The word "sigmoid" sometimes refers to this particular function.

Activation functions

Discontinuous activation functions (not recommended[†]):

- Heaviside function: $\varphi(v) = \mathbb{1}_{v \geq 0}$, or
- sign function: $\varphi(v) = \operatorname{sgn}(v) = \mathbb{1}_{v>0} - \mathbb{1}_{v<0}$.

"S-shaped" functions, a.k.a. sigmoids:

- logistic[‡] : $\varphi(v) = \frac{1}{1+e^{-v}} = \frac{1}{2} + \frac{1}{2}\tanh\left(\frac{v}{2}\right)$, or
- tanh : $\varphi(v) = \tanh(v) = \frac{e^{v}-e^{-v}}{e^{v}+e^{-v}}$.

The ReLU (*Rectified Linear Unit*) function:

- $\varphi(v) = \max(0, v)$.

[†] Used in the oldest models, most notably the Rosenblatt's perceptron (1957), but abandonned since then because of their almost-everywhere zero gradient, which creates problems for optimization procedures.

[‡] The word "sigmoid" sometimes refers to this particular function.

Activation functions (cont'd)

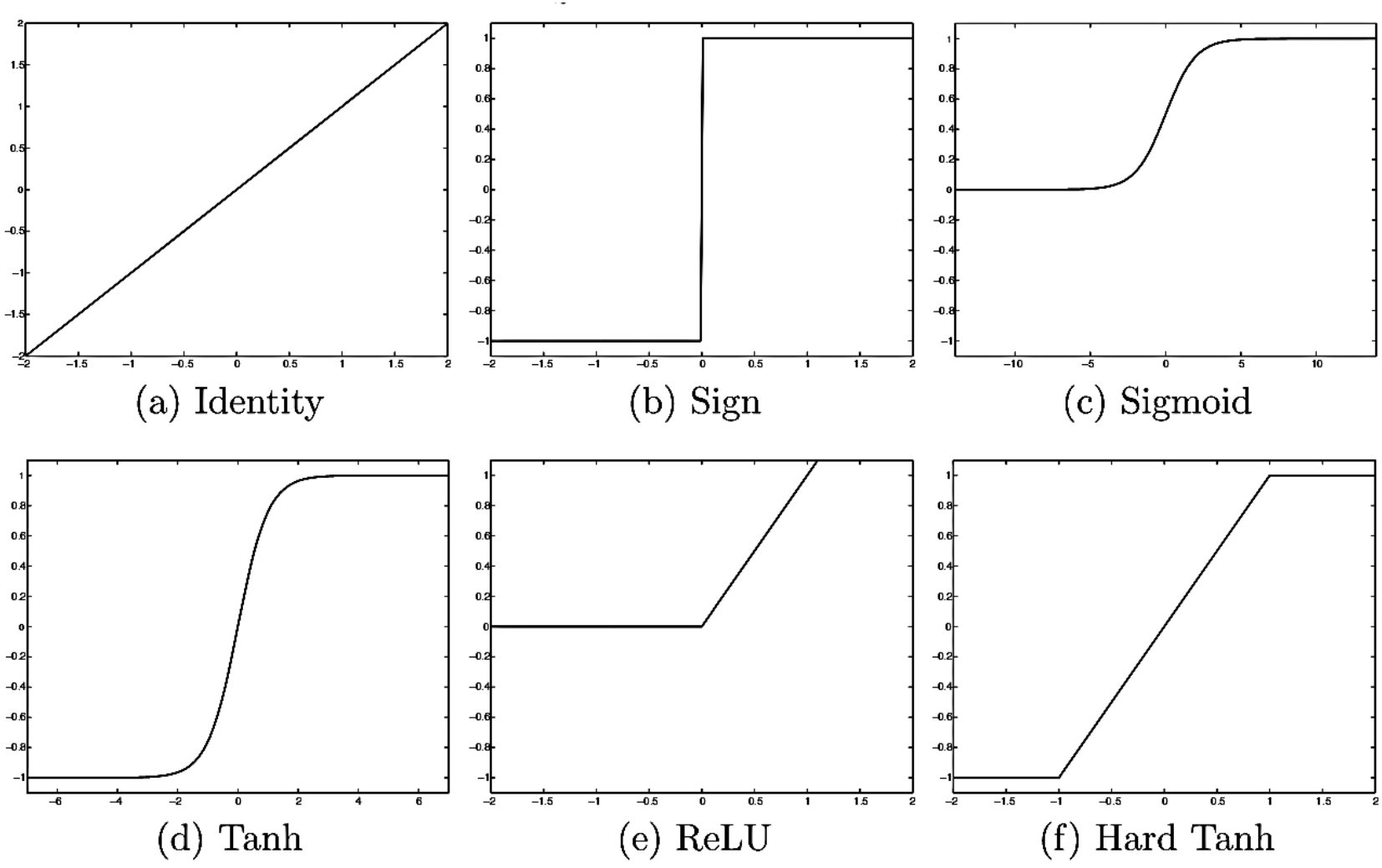


Image: C. C. Aggarwal (2018). *Neural networks and Deep Learning*, Springer.

Remark: relation with logistic regression

Remark. With the logistic activation function (sigmoid),

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

Since $v = wx + b$, we recover for $h(x)$ the form of the logistic regression predictor.

Lecture outline

1 – Some general notions about classification

2 – Logistic regression [classification]

3 – Decision trees [regression + classification]

4 – Neural networks [regression + classification]

- 4.1 – Neurons
- 4.2 – Multi-layer perceptrons
- 4.3 – Example
- 4.4 – Other architectures

5 – Standard exercices (with solutions)

6 – Appendices

Multi-layer perceptron: definition

Let p, K be non-zero integers.

Definition: multi-layer perceptron† (MLP)

We call multi-layer perceptron with $M+1$ layers, p variables (input) and K responses (output), any function $\mathbb{R}^p \to \mathbb{R}^K$ of the form

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

where‡

- $g_k : \mathbb{R}^{m_{k-1}} \to \mathbb{R}^{m_k}$ is affine,
- $\underline{\varphi}_k : \mathbb{R}^{m_k} \to \mathbb{R}^{m_k}$ represents the action coordinate by coordinate of an increasing function $\varphi_k : \mathbb{R} \to \mathbb{R}$.
- $m_0, m_1, \ldots, m_M$: non-zero integers, $m_0 = p$, $m_M = K$.

† Rosenblatt's original perceptron (1957) did not include hidden layers ($M = 1$). It was using the activation function $h(x) = \text{sgn}(x)$ as McCulloch and Pitts (1943), and weights $w_j \in \{-1, +1, -\infty\}$.

‡ there will be one exception this rule later ("softmax" layer)

Multi-layer perceptron: definition (cont'd)

Vocabulary: layers of variables

- $z_{[0]} = x$: input layer,
- $z_{[k]} = \big(\underline{\varphi}_k \circ g_k\big)\big(z_{[k-1]}\big)$, $1 \leq k < M$: hidden layers,
- $z_{[M]} = y = \big(\underline{\varphi}_M \circ g_M\big)\big(z_{[M-1]}\big)$: output layer.

Remark. Let us write

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

Then, for all $j \in \{1, \ldots, m_k\}$ we recognize a neuron:

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

where $w_{k,j} = e_j^\top W_k$ is the j-th row of W_k.

➠ Vocabulary: weights, bias, activation function.

Multi-layer perceptron: definition (cont'd)

Vocabulary: layers of variables

- $z_{[0]} = x$: input layer,
- $z_{[k]} = \big(\underline{\varphi}_k \circ g_k\big)\big(z_{[k-1]}\big)$, $1 \leq k < M$: hidden layers,
- $z_{[M]} = y = \big(\underline{\varphi}_M \circ g_M\big)\big(z_{[M-1]}\big)$: output layer.

Remark. Let us write

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

Then, for all $j \in \{1, \ldots, m_k\}$ we recognize a neuron:

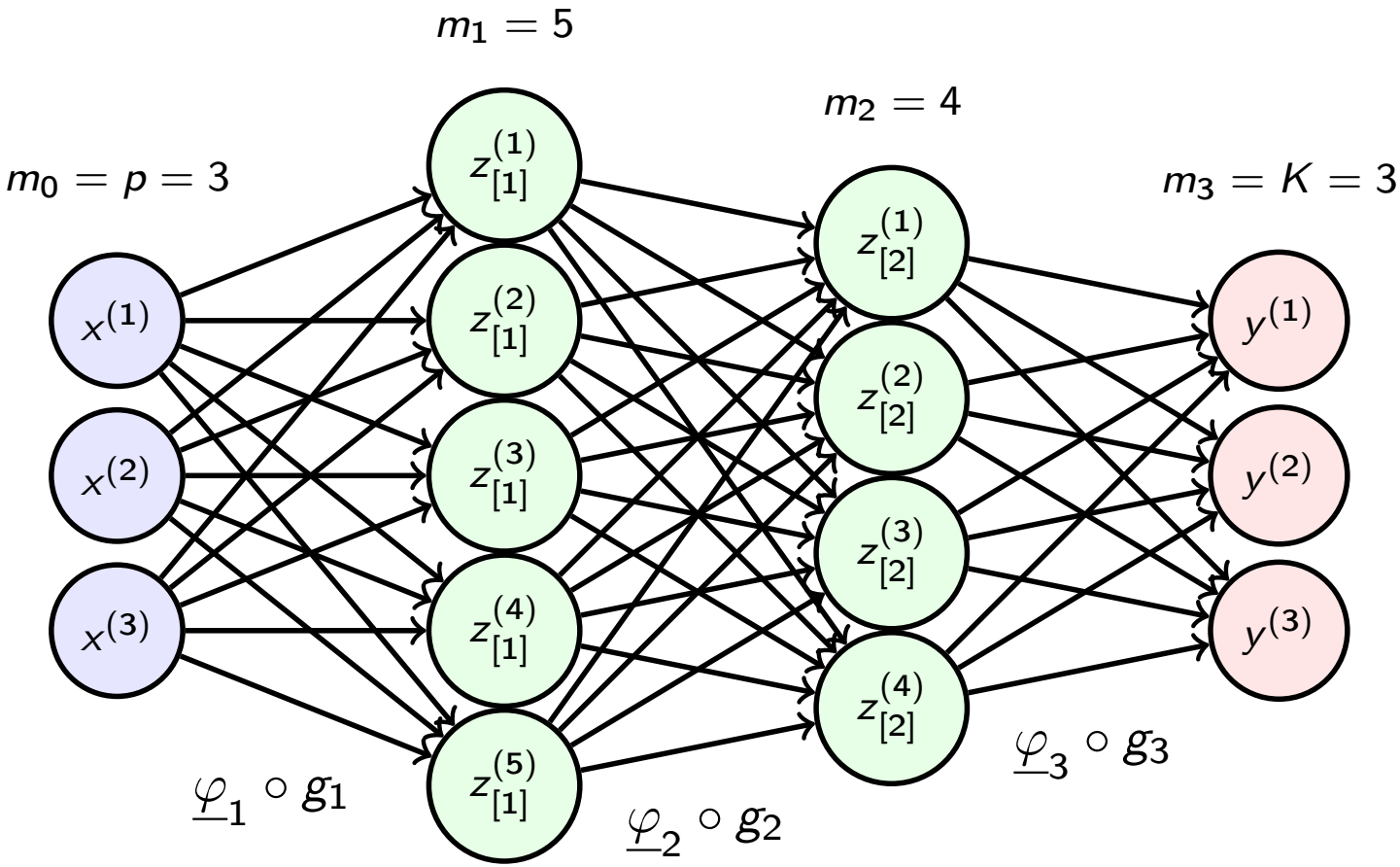
$$z_{[k]}^{(j)} \;=\; \varphi_k\left(w_{k,j}\, z_{[k-1]} + b_k^{(j)}\right),$$

where $w_{k,j} = e_j^\top W_k$ is the j-th row of W_k.

➠ Vocabulary: weights, bias, activation function.

Multi-layer perceptron: illustration

Example of a multi-layer perceptron with $p = 3$ inputs, $K = 3$ outputs, and two hidden layers of sizes $m_1 = 5$ and $m_2 = 4$.



Vocabulary: fully connected, feed-forward neural network

Output layer: activation function

The output layer must be adapted to the problem at hand...

Regression. $\mathcal{Y} \subset \mathbb{R}$, or more generally $\mathbb{R}^K$.

- Perceptron with K outputs
- Activation function: $\varphi_M = \mathrm{Id}$.
- Thus the last transformation $(\varphi_M \circ g_M)$ is linear (affine).

Classification. K classes, $\mathcal{Y} = [0,1]^K$ ("soft" classification).

- Perceptron with K outputs, with $m_{M-1} = m_M = K$.
- Exception to the definition ➠ the "softmax" layer:

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

Remark: alternatively, for *binary* classification, we can use a single output ($K = 1$ instead of $K = 2$) with the *logistic* function used as the activation function on the last layer.

Output layer: activation function

The output layer must be adapted to the problem at hand...

Regression. $\mathcal{Y} \subset \mathbb{R}$, or more generally $\mathbb{R}^K$.

- Perceptron with K outputs
- Activation function: $\varphi_M = \mathrm{Id}$.
- Thus the last transformation $(\varphi_M \circ g_M)$ is linear (affine).

Classification. K classes, $\mathcal{Y} = [0,1]^K$ ("soft" classification).

- Perceptron with K outputs, with $m_{M-1} = m_M = K$.
- Exception to the definition ➠ the "softmax" layer:

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

Remark: alternatively, for *binary* classification, we can use a single output ($K = 1$ instead of $K = 2$) with the *logistic* function used as the activation function on the last layer.

Output layer: activation function

The output layer must be adapted to the problem at hand...

Regression. $\mathcal{Y} \subset \mathbb{R}$, or more generally $\mathbb{R}^K$.

- Perceptron with K outputs
- Activation function: $\varphi_M = \mathrm{Id}$.
- Thus the last transformation $(\varphi_M \circ g_M)$ is linear (affine).

Classification. K classes, $\mathcal{Y} = [0,1]^K$ ("soft" classification).

- Perceptron with K outputs, with $m_{M-1} = m_M = K$.
- Exception to the definition ➠ the "softmax" layer:

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

Remark: alternatively, for *binary* classification, we can use a single output ($K = 1$ instead of $K = 2$) with the *logistic* function used as the activation function on the last layer.

Training: loss functions and regularization

The most commonly used loss functions[†] are

- regression: the quadratic loss
  - $L(y, \tilde{y}) = (y - \tilde{y})^2$ for the single-output case,
  - $L(y, \tilde{y}) = \|y - \tilde{y}\|^2$ if $K > 1$.
- (soft) classification: the logarithmic loss
  - For all $j \in \{1, \ldots, K\}$, we have $y^{(j)} \in \{0, 1\}$ and $\tilde{y}^{(j)} \in [0, 1]$.
  - $L(y, \tilde{y}) = -\sum_{j=1}^{K} y^{(j)} \ln\left(\tilde{y}^{(j)}\right)$.

Nb parameters is high $\Rightarrow$ regularize to avoid over-fitting

- penalization, for instance L^1 (LASSO) or L^2 (ridge);
- other (not covered): early stopping, drop out...

[†] for instance `https://scikit-learn.org/stable/modules/neural_networks_supervised.html`

Training: loss functions and regularization

The most commonly used loss functions[†] are

- regression: the quadratic loss
  - $L(y, \tilde{y}) = (y - \tilde{y})^2$ for the single-output case,
  - $L(y, \tilde{y}) = \|y - \tilde{y}\|^2$ if $K > 1$.
- (soft) classification: the logarithmic loss
  - For all $j \in \{1, \ldots, K\}$, we have $y^{(j)} \in \{0, 1\}$ and $\tilde{y}^{(j)} \in [0, 1]$.
  - $L(y, \tilde{y}) = -\sum_{j=1}^{K} y^{(j)} \ln\left(\tilde{y}^{(j)}\right)$.

Nb parameters is high $\Rightarrow$ regularize to avoid over-fitting

- penalization, for instance L^1 (LASSO) or L^2 (ridge);
- other (not covered): early stopping, drop out. . .

[†] for instance `https://scikit-learn.org/stable/modules/neural_networks_supervised.html`

Training: loss functions and regularization

The most commonly used loss functions[†] are

- regression: the quadratic loss
  - $L(y, \tilde{y}) = (y - \tilde{y})^2$ for the single-output case,
  - $L(y, \tilde{y}) = \|y - \tilde{y}\|^2$ if $K > 1$.
- (soft) classification: the logarithmic loss
  - For all $j \in \{1, \ldots, K\}$, we have $y^{(j)} \in \{0, 1\}$ and $\tilde{y}^{(j)} \in [0, 1]$.
  - $L(y, \tilde{y}) = -\sum_{j=1}^{K} y^{(j)} \ln\left(\tilde{y}^{(j)}\right)$.

Nb parameters is high $\Rightarrow$ regularize to avoid over-fitting

- penalization, for instance L^1 (LASSO) or L^2 (ridge);
- other (not covered): early stopping, drop out...

[†] for instance `https://scikit-learn.org/stable/modules/neural_networks_supervised.html`

Training: numerical optimization

We want to minimize the empirical risk (possibly penalized)

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

where θ denotes the parameters of the model (weights, biases).

➠ Numerical methods are used to this end.

These methods use the gradient of the criterion. Two remarks:

- computational burden when n is large: random "mini-batchs"
  - ➠ stochastic gradient method (not covered);
- recursive computation of the gradient of a composition of fcts
  - ➠ back-propagation method (not covered).

Training: numerical optimization

We want to minimize the empirical risk (possibly penalized)

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

where θ denotes the parameters of the model (weights, biases).

➠ Numerical methods are used to this end.

These methods use the gradient of the criterion. Two remarks:

- computational burden when n is large: random "mini-batchs"
  - ➠ stochastic gradient method (not covered);
- recursive computation of the gradient of a composition of fcts
  - ➠ back-propagation method (not covered).

Training: numerical optimization

We want to minimize the empirical risk (possibly penalized)

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

where θ denotes the parameters of the model (weights, biases).

➠ Numerical methods are used to this end.

These methods use the gradient of the criterion. Two remarks:

- computational burden when n is large: random "mini-batchs"
  - ➠ stochastic gradient method (not covered);
- recursive computation of the gradient of a composition of fcts
  - ➠ back-propagation method (not covered).

Lecture outline

1 – Some general notions about classification

2 – Logistic regression [classification]

3 – Decision trees [regression + classification]

4 – Neural networks [regression + classification]
- 4.1 – Neurons
- 4.2 – Multi-layer perceptrons
- 4.3 – Example
- 4.4 – Other architectures

5 – Standard exercices (with solutions)

6 – Appendices

Example: MNIST

70 000 images† of size 28 × 28 pixels (256 gray levels)

Problem: multi-class classification (10 classes);

training: 60 000 images / test: 10 000 images

Source: `http://yann.lecun.com/exdb/mnist/`

Example: MNIST

➠ see Jupyter / Python / Scikit-Learn notebook

Lecture outline

1 – Some general notions about classification

2 – Logistic regression [classification]

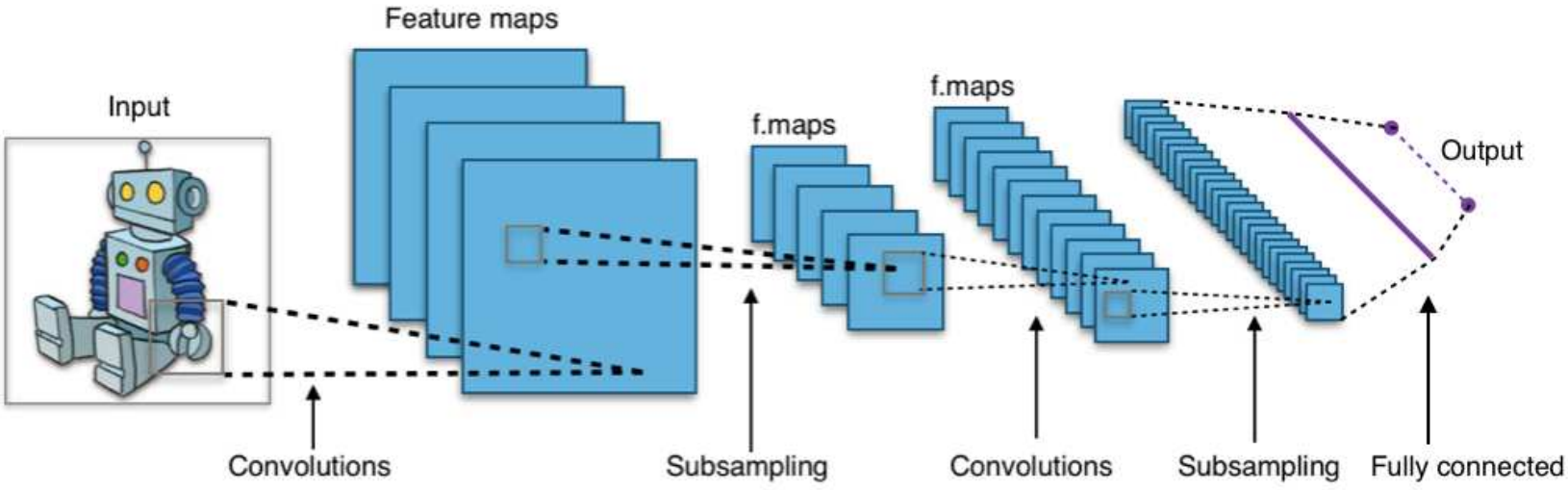
3 – Decision trees [regression + classification]

4 – Neural networks [regression + classification]
- 4.1 – Neurons
- 4.2 – Multi-layer perceptrons
- 4.3 – Example
- 4.4 – Other architectures

5 – Standard exercices (with solutions)

6 – Appendices

Convolutional neural networks (CNNs)



Schematic diagram of a typical CNN

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

Lecture outline

1 – Some general notions about classification

2 – Logistic regression [classification]

3 – Decision trees [regression + classification]

4 – Neural networks [regression + classification]

5 – Standard exercices (with solutions)
- 5.1 – Questions
- 5.2 – Solutions

6 – Appendices

Lecture outline

1 – Some general notions about classification

2 – Logistic regression [classification]

3 – Decision trees [regression + classification]

4 – Neural networks [regression + classification]

5 – Standard exercices (with solutions)

5.1 – Questions

5.2 – Solutions

6 – Appendices

Exercise 1 (Optimal classifier with the logarithmic loss)

solution

Question

Show the proposition stated on slide 12:

$h : \mathcal{X} \to [0, 1]$ is optimal for the logarithmic loss iff, P^X-ae,

$$h(x) \;=\; \mathrm{P}\left(Y = 1 \mid X = x\right).$$

Exercise 2 (Multi-class logistic regression)

solution

When the number of classes K is ≥ 3, the classification problem is called Multiclass.

Let $\{0, 1, \ldots, K-1\}$ be the set of labels (classes), $K \geq 3$.

Binary logistic regression can be extended to multi-class classification by

- selecting a reference class (here, "0")
- performing $K-1$ binary logistic regressions:

$$\begin{cases} \ln\left(\frac{\mathrm{P}(Y=1|X=x)}{\mathrm{P}(Y=0|X=x)}\right) & = & \beta_{1,0} + \beta_1^\top x \\ \vdots & & \\ \ln\left(\frac{\mathrm{P}(Y=K-1|X=x)}{\mathrm{P}(Y=0|X=x)}\right) & = & \beta_{K-1,0} + \beta_{K-1}^\top x \end{cases}$$

Exercise 2 (Multi-class logistic regression (cont'd))

solution

To simplify notation, we'll assume that the matrix of explanatory variables contains a constant vector and make the following change: $\beta_k \leftarrow (\beta_{k,0}, \beta_k)$

Questions

1. Give the expression of $\mathrm{P}(Y = k|X = x)$ from the vectors β_k,
2. Deduce from 1. that the choice of reference class is arbitrary (it has no influence on the regression model),
3. Express the log-likelihood

Lecture outline

1 – Some general notions about classification

2 – Logistic regression [classification]

3 – Decision trees [regression + classification]

4 – Neural networks [regression + classification]

5 – Standard exercices (with solutions)

5.1 – Questions

5.2 – Solutions

6 – Appendices

Solution of exercise 1

back to questions

The optimal classifier $h^\star$ minimizes $\mathbb{E}\left(L(Y, h(X))\right)$, where the expectation is taken over (X, Y).

By conditioning, we have:

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

Thus:

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

$$\Updownarrow$$

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

Solution of exercise 1

back to questions

With the logarithmic loss:

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

Solution of exercise 1

back to questions

The minimization of $\mathcal{J}(t)$ is achieved by examining the sign of the derivative of $\mathcal{J}(t)$:

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

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

- strictly negative for $t \in]0, \mathbb{P}(Y=1|X=x)[$,
- zero at $t = \mathbb{P}(Y=1|X=x)$,
- strictly positive for $t \in]\mathbb{P}(Y=1|X=x), 1[$,

Solution of exercise 1

back to questions

Conclusion: $t = \mathbb{P}(Y = 1|X = x)$ is the unique minimizer on $[0, 1]$ of the function $\mathcal{J}(t)$.

When

- $\mathbb{P}(Y = 1|X = x) = 0$, $\mathcal{J}(t)$ is minimal at $t = 0$,
- $\mathbb{P}(Y = 1|X = x) = 1$, $\mathcal{J}(t)$ est minimal at $t = 1$.

We conclude that:

1. $h^\star : x \mapsto \mathbb{P}(Y = 1|X = x)$ is optimal,
2. $h^\star(x)$ is the unique minimizer of the fonction $\mathcal{J} : t \mapsto \mathbb{E}_{Y|X}\left(L(Y, t)|X = x\right)$.

Solution of exercise 1

back to questions

In converse : suppose that h optimal.
Let the function:

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

We have:

- $\mathbb{E}_X(g(X)) = \int_{\mathcal{X}} g(x) d\mathrm{P}^X = 0$ (otherwise, h would not be optimal),
- $\forall x,\ g(x) \geq 0$ by the optimality of $h^\star$.

Therefore, $g = 0\ \mathrm{P}^X - a.e.$, that is:

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

Solution of exercise 1

back to questions

$h^\star(x)$ is the unique minimizer of the function $\mathcal{J} : t \mapsto \mathbb{E}_{Y|X}\left(L(Y,t)|X=x\right)$, we have:

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

We conclude that $h = h^\star$ $\mathrm{P}^X - a.e.$, establishing the desired equivalence.

Solution of exercise 2

back to questions

1. For all $k \in \{1, \ldots, K-1\}$:

$$\mathrm{P}(Y = k|X = x) = \exp\left(\beta_k^\top x\right) \mathrm{P}(Y = 0|X = x).$$

Following $\sum_{k=0}^{K-1} \mathrm{P}(Y = k|X = x) = 1$, we have:

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

Solution of exercise 2

back to questions

❷ Let $\tilde{k} \neq 0$. Suppose that we select the class $\tilde{k}$ as the reference and for $k \neq \tilde{k}$ the vector $\tilde{\beta}_k$ of the model is given by:

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

Since

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

we have:

$$\left\{\begin{array}{rcl}\tilde{\beta}_k &=& \beta_k - \beta_{\tilde{k}},\ k \neq 0,\ k \neq \tilde{k}\\ \tilde{\beta}_0 &=& -\beta_{\tilde{k}}\end{array}\right. \qquad (2)$$

Solution of exercise 2

back to questions

By using the results of Question ❶, we have that for $k \neq \tilde{k}$:

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

By substituting $\tilde{\beta}_k$ in these equations using the relations (2), we obtain the equations (1).

Summary: Changing the reference class alters the model's parameters (without changing the model itself).

Solution of exercise 2

back to questions

❸ To establish the connection with binary logistic regression, we represent the observation $y_i \in \{0, \ldots, K-1\}$ by the vector $z_i \in \{0,1\}^K$:

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

We also denote $\boldsymbol{\beta}$ the set of vectors $\beta_1, \ldots, \beta_{K-1}$.

Solution of exercise 2

back to questions

We express the likelihood associated with the example (x_i, y_i):

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

Solution of exercise 2

back to questions

Using :

- the $(K-1)$ regression models:

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

- the second equation of (1):

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

the log-likelihood is expressed as:

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

Lecture outline

1 – Some general notions about classification

2 – Logistic regression [classification]

3 – Decision trees [regression + classification]

4 – Neural networks [regression + classification]

5 – Standard exercices (with solutions)

6 – Appendices

Generalized linear models

Definition

The GLM contains all statistical models such that

- $Y|X$ follows a distribution from an exponential family:

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

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

Vocabulary. The function g is called the link function.[†]

Example. Bernoulli distributions form an exponential family.

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

[†] Let N denote the set of admissible value for η: g is often chosen to be a bijection from N to $\mathbb{R}$.

Generalized linear models

Definition

The GLM contains all statistical models such that

- $Y|X$ follows a distribution from an exponential family:

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

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

Vocabulary. The function g is called the link function.[†]

Example. Bernoulli distributions form an exponential family.

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

[†] Let N denote the set of admissible value for η: g is often chosen to be a bijection from N to $\mathbb{R}$.

Remark: generalized linear models (GLM)

The logistic regression model has the form

- $Y|X \sim \mathrm{Ber}\left(\mathbb{E}_\beta(Y|X)\right)$,
- $g\left(\mathbb{E}_\beta(Y|X)\right) = \beta_0 + \beta^\top X, \quad$ with $g = \mathrm{logit}$.

➠ special case of the generalized linear model (GLM)
(g is called link function)

➠ complement

Remark: we have already met another GLM model

- $Y|X \sim \mathcal{N}\left(\mathbb{E}_\beta(Y|X), \sigma^2\right)$
- $g\left(\mathbb{E}_\beta(Y|X)\right) = \beta_0 + \beta^\top X$ with $g = \mathrm{Id}$

Remark: generalized linear models (GLM)

The logistic regression model has the form

- $Y|X \sim \mathrm{Ber}\left(\mathbb{E}_\beta(Y|X)\right)$,
- $g\left(\mathbb{E}_\beta(Y|X)\right) = \beta_0 + \beta^\top X, \quad$ with $g = \mathrm{logit}$.

➠ special case of the generalized linear model (GLM)
(g is called link function)

➠ complement

Remark: we have already met another GLM model

- $Y|X \sim \mathcal{N}\left(\mathbb{E}_\beta(Y|X), \sigma^2\right)$
- $g\left(\mathbb{E}_\beta(Y|X)\right) = \beta_0 + \beta^\top X$ with $g = \mathrm{Id}$

Example: $Y_i|X_i \overset{\text{iid}}{\sim} \text{Poisson}(\theta_i)$, with $\ln \theta_i = \beta_0 + \beta_1 X_i$

Poisson distributions form an exponential family:

$$f(y) = \exp(-\theta)\,\frac{\theta^y}{y!}$$

$$= \frac{1}{y!}\exp(-\theta)\exp(\ln(\theta)y) \qquad \Rightarrow \eta = \ln(\theta)$$