

# Introduction à l'apprentissage automatique

Les bases du *machine learning*

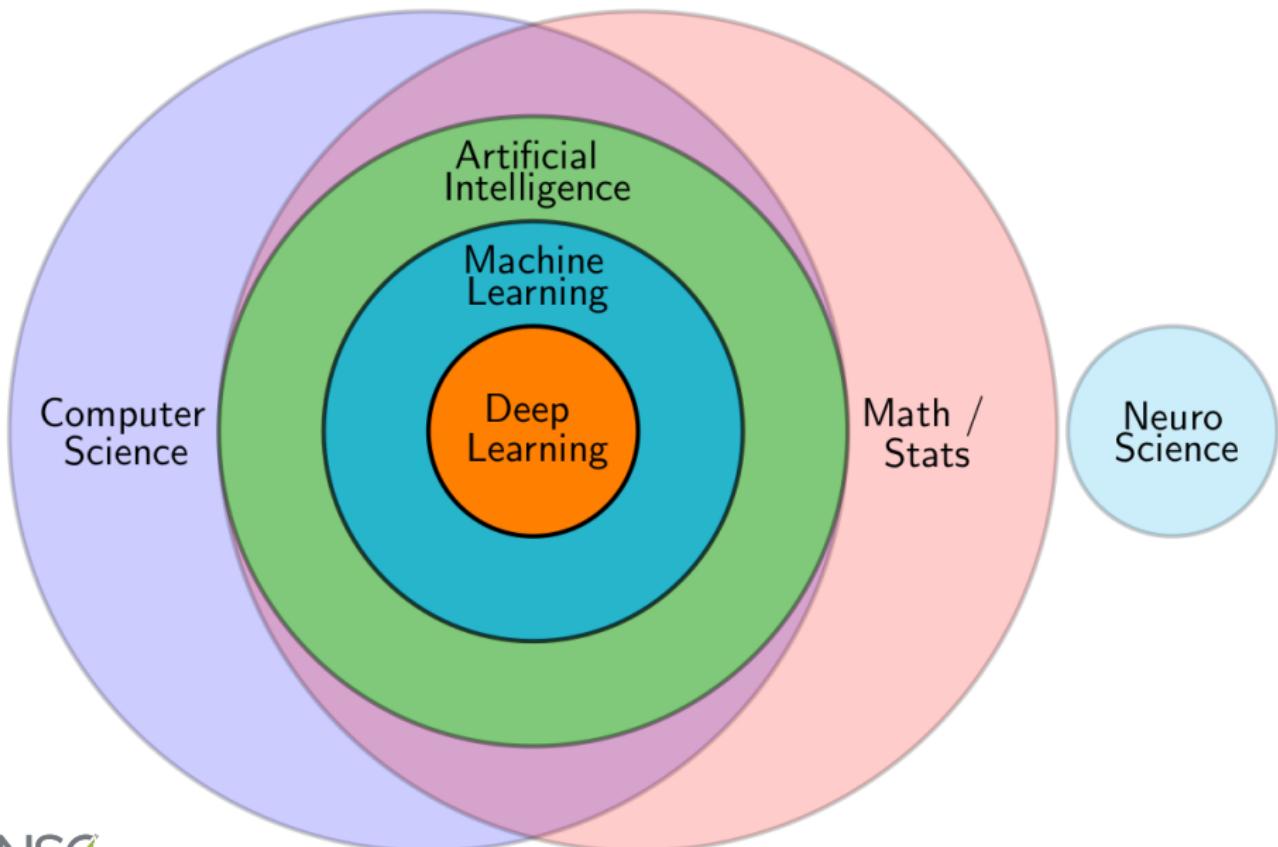
---

Nicolas Audebert [nicolas.audebert@ign.fr](mailto:nicolas.audebert@ign.fr)

École nationale des sciences géographiques

11 février 2025





# Historique de l'apprentissage automatique

**Objectif** : automatiser une prédiction à partir de valeurs observées, uniquement à partir d'exemples ( $\approx$  méta-heuristique)

Deux grandes familles d'approche :

- ▶ Symbolique : systèmes experts, arbres de décision, logique floue, ontologies
  - ▶ Formalisation de connaissances expertes encodées dans le système
  - ▶ Règles de logiques, par ex. déductives pour tirer des inférences
  - ▶ Interprétable et permet d'intégrer des connaissances humaines
  - ▶ Limité à des tâches pour lesquelles on peut encoder beaucoup d'information existante
- ▶ Connexionniste : réseaux de neurones, inférence bayésienne
  - ▶ Modélisation d'une distribution statistique entre entrées et sorties
  - ▶ Nécessite beaucoup d'exemples
  - ▶ Pas facilement interprétable
  - ▶ Mais flexible et adaptable à de nombreux problèmes

# Historique de l'apprentissage automatique

**Objectif** : automatiser une prédiction à partir de valeurs observées, uniquement à partir d'exemples ( $\approx$  méta-heuristique)

Deux grandes familles d'approche :

- ▶ Symbolique : systèmes experts, arbres de décision, logique floue, ontologies
  - ▶ Formalisation de connaissances expertes encodées dans le système
  - ▶ Règles de logiques, par ex. déductives pour tirer des inférences
  - ▶ Interprétable et permet d'intégrer des connaissances humaines
  - ▶ Limité à des tâches pour lesquelles on peut encoder beaucoup d'information existante
- ▶ **Connexionniste** : réseaux de neurones, inférence bayésienne
  - ▶ Modélisation d'une distribution statistique entre entrées et sorties
  - ▶ Nécessite beaucoup d'exemples
  - ▶ Pas facilement interprétable
  - ▶ Mais flexible et adaptable à de nombreux problèmes

# Quelques exemples d'application

- ▶ Perception visuelle :
  - ▶ Reconnaissance/détection d'objets
  - ▶ Cartographie automatisée
  - ▶ Reconstruction 3D
  - ▶ Reconnaissance de caractères
- ▶ Perception auditive :
  - ▶ Transcription automatisée
  - ▶ Séparation de sources
  - ▶ Synthèse vocale
- ▶ Analyse du langage :
  - ▶ Traduction de texte
  - ▶ Génération de texte
  - ▶ Analyse de sentiments
- ▶ Traitement du signal :
  - ▶ Prédictions météorologiques
  - ▶ Débruitage

# Apprentissage automatique à l'IGN

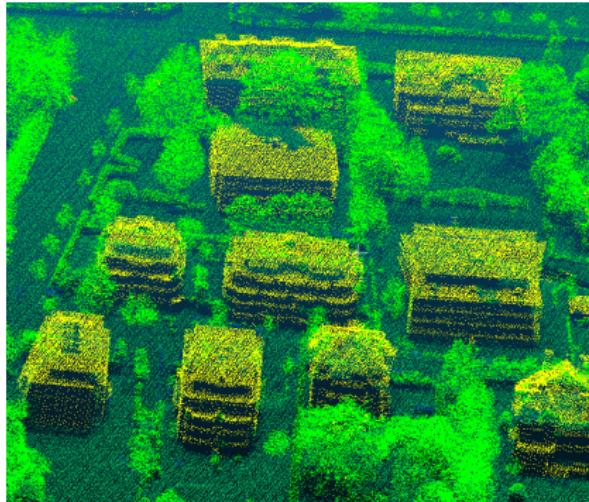
- ▶ Cartographie d'occupation des sols à partir d'images satellites
- ▶ Sémantisation de nuages de points massifs
- ▶ Numérisation/vectorisation de cartes historiques
- ▶ Reconnaissances des espèces d'arbres
- ▶ et bien d'autres



<https://geoservices.ign.fr/ressources-ia-de-couverture-du-sol>

# Apprentissage automatique à l'IGN

- ▶ Cartographie d'occupation des sols à partir d'images satellites
- ▶ Sémantisation de nuages de points massifs
- ▶ Numérisation/vectorisation de cartes historiques
- ▶ Reconnaissances des espèces d'arbres
- ▶ et bien d'autres



<https://github.com/IGNF/myria3d>

## Apprentissage automatique à l'IGN

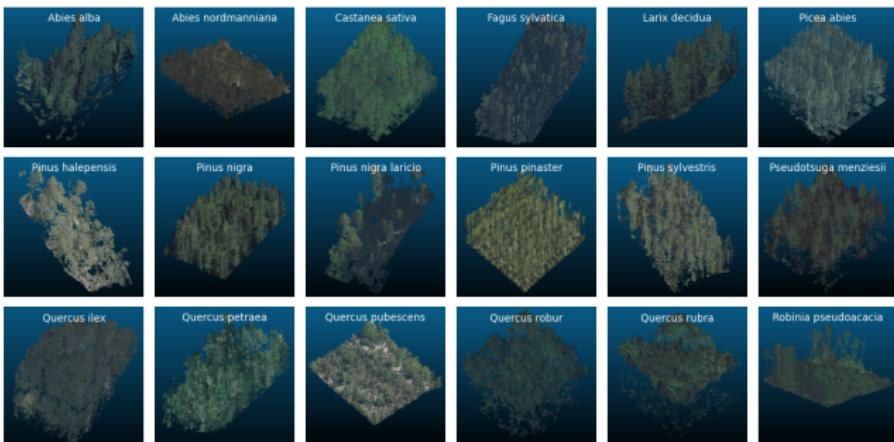
- ▶ Cartographie d'occupation des sols à partir d'images satellitaires
  - ▶ Sémantisation de nuages de points massifs
  - ▶ **Numérisation/vectorisation de cartes historiques**
  - ▶ Reconnaissances des espèces d'arbres
  - ▶ et bien d'autres



<https://icdar21-mapseg.github.io/>

# Apprentissage automatique à l'IGN

- ▶ Cartographie d'occupation des sols à partir d'images satellitaires
- ▶ Sémantisation de nuages de points massifs
- ▶ Numérisation/vectorisation de cartes historiques
- ▶ Reconnaissances des espèces d'arbres
- ▶ et bien d'autres



<https://huggingface.co/datasets/IGNF/PureForest>

- ▶ Cartographie d'occupation des sols à partir d'images satellitaires
- ▶ Sémantisation de nuages de points massifs
- ▶ Numérisation/vectorisation de cartes historiques
- ▶ Reconnaissances des espèces d'arbres
- ▶ et bien d'autres
  - ▶ Reconstruction 3D par photogrammétrie
  - ▶ Classification de parcelles agricoles
  - ▶ Transfert de styles pour la cartographie
  - ▶ Localisation de personnes perdues en montagne
  - ▶ ...

# **Qu'est-ce que l'apprentissage automatique ?**

---

## Apprentissage automatique

Sous-discipline de l'intelligence artificielle utilisant des approches algorithmiques qui sont capables d'améliorer leurs performances sur une tâche à partir d'exemples, sans être explicitement programmées.

Programmation classique : données + algorithme = résultat

Apprentissage automatique : données + résultat = algorithme

En pratique, on appelle l'algorithme obtenu un **modèle** (en général, décisionnel).

## Apprentissage automatique

Sous-discipline de l'intelligence artificielle utilisant des approches algorithmiques qui capables d'améliorer leurs performances sur une tâche à partir d'exemples, sans être explicitement programmées.

**Programmation classique** : données + algorithme = résultat

**Apprentissage automatique** : données + résultat = algorithme

En pratique, on appelle l'algorithme obtenu un **modèle** (en général, décisionnel).

## Apprentissage automatique

Sous-discipline de l'intelligence artificielle utilisant des approches algorithmiques qui sont capables d'améliorer leurs performances sur une tâche à partir d'exemples, sans être explicitement programmées.

**Programmation classique** : données + algorithme = résultat

**Apprentissage automatique** : données + résultat = algorithme

En pratique, on appelle l'algorithme obtenu un **modèle** (en général, décisionnel).

- ▶ Observations décrites par les valeurs prises par un ensemble de variables
- Objectif : prédire, pour chaque donnée, la valeur d'une variable (**expliquée** ou « dépendante » ou « de sortie ») à partir des valeurs des autres variables (**explicatives** ou « d'entrée »)

## Exemples

1. Une image représente un visage ou non ?
2. Les symptômes correspondent à la maladie A ou B ou C ou aucune ?
3. Quel sera le débit de la Loire à Tours dans 48h ?
4. Quelle est l'entité nommée dans « La Maison Blanche a démenti ces informations. » ?
5. Quelle est la région d'une image correspondant aux routes ?

- ▶ Observations décrites par les valeurs prises par un ensemble de variables
- Objectif : prédire, pour chaque donnée, la valeur d'une variable (**expliquée** ou « dépendante » ou « de sortie ») à partir des valeurs des autres variables (**explicatives** ou « d'entrée »)

## Exemples

1. Une image représente un visage ou non ?
2. Les symptômes correspondent à la maladie A ou B ou C ou aucune ?
3. Quel sera le débit de la Loire à Tours dans 48h ?
4. Quelle est l'entité nommée dans « La Maison Blanche a démenti ces informations. » ?
5. Quelle est la région d'une image correspondant aux routes ?

- ▶ Observations décrites par les valeurs prises par un ensemble de variables
- Objectif : prédire, pour chaque donnée, la valeur d'une variable (**expliquée** ou « dépendante » ou « de sortie ») à partir des valeurs des autres variables (**explicatives** ou « d'entrée »)

## Exemples

1. Une image représente un visage ou non ?
2. Les symptômes correspondent à la maladie A ou B ou C ou aucune ?
3. Quel sera le débit de la Loire à Tours dans 48h ?
4. Quelle est l'entité nommée dans « La Maison Blanche a démenti ces informations. » ?
5. Quelle est la région d'une image correspondant aux routes ?

- ▶ Observations décrites par les valeurs prises par un ensemble de variables
- Objectif : prédire, pour chaque donnée, la valeur d'une variable (**expliquée** ou « dépendante » ou « de sortie ») à partir des valeurs des autres variables (**explicatives** ou « d'entrée »)

## Exemples

1. Une image représente un visage ou non ?
2. Les symptômes correspondent à la maladie A ou B ou C ou aucune ?
3. Quel sera le débit de la Loire à Tours dans 48h ?
4. Quelle est l'entité nommée dans « La Maison Blanche a démenti ces informations. » ?
5. Quelle est la région d'une image correspondant aux routes ?

- ▶ Observations décrites par les valeurs prises par un ensemble de variables
- Objectif : prédire, pour chaque donnée, la valeur d'une variable (**expliquée** ou « dépendante » ou « de sortie ») à partir des valeurs des autres variables (**explicatives** ou « d'entrée »)

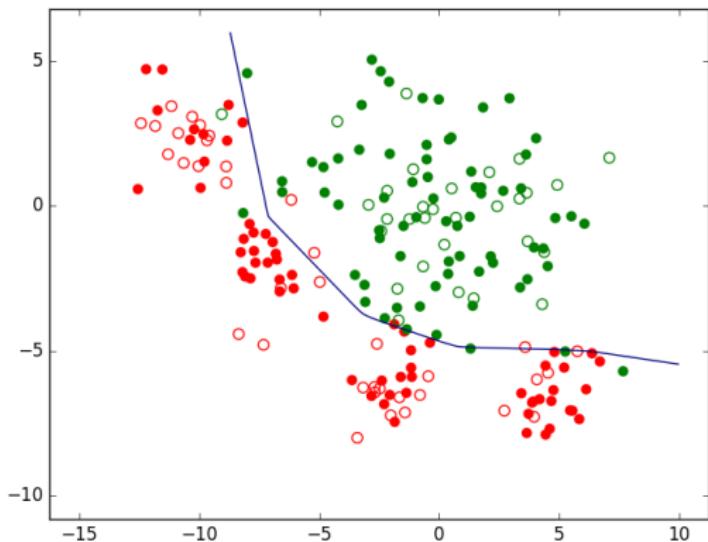
## Exemples

1. Une image représente un visage ou non ?
2. Les symptômes correspondent à la maladie A ou B ou C ou aucune ?
3. Quel sera le débit de la Loire à Tours dans 48h ?
4. Quelle est l'entité nommée dans « La Maison Blanche a démenti ces informations. » ?
5. Quelle est la région d'une image correspondant aux routes ?

1. Classification : la variable expliquée est une variable nominale, chaque observation possède une modalité (appelée en général **classe**)
  - quel est le chiffre représenté par cette image ?
2. Régression : la variable expliquée est une variable quantitative (domaine  $\subset \mathbb{R}$ )
  - combien vaudra le CAC 40 dans une semaine ?
3. Prédiction structurée : la variable expliquée prend des valeurs dans un domaine de données **structurées** (les relations entre parties comptent)
  - quelle est la forme de la protéine compte-tenu des molécules qui la composent ?

# Qu'est-ce qu'un modèle ?

- ▶ Modèle = règle de décision
- ▶ Exemple : frontière de discrimination pour une classification à 2 classes



## 1. Construction analytique à partir des connaissances du phénomène

► Exemples :

- ▶ Temps de vol  $t \leftarrow$  distance  $d$  et vitesse  $v$ ,  $t = v \cdot d$
- ▶ Concentration de produit de réaction  $\leftarrow$  concentration de réactif et température

► Néglige souvent l'impact de variables non contrôlables !

## 2. À partir de données : ensemble d'observations pour lesquelles les valeurs des variables explicatives et des variables expliquées sont en général connues

→ Apprentissage supervisé : à partir d'observations pour lesquelles les valeurs des variables explicatives et de la variable expliquée sont connues

## 1. Construction analytique à partir des connaissances du phénomène

### ► Exemples :

- ▶ Temps de vol  $t \leftarrow$  distance  $d$  et vitesse  $v$ ,  $t = v \cdot d$
- ▶ Concentration de produit de réaction  $\leftarrow$  concentration de réactif et température

▶ Néglige souvent l'impact de variables non contrôlables !

## 2. À partir de données : ensemble d'observations pour lesquelles les valeurs des variables explicatives et des variables expliquées sont en général connues

→ Apprentissage supervisé : à partir d'observations pour lesquelles les valeurs des variables explicatives et de la variable expliquée sont connues

## 1. Construction analytique à partir des connaissances du phénomène

- ▶ Exemples :

- ▶ Temps de vol  $t \leftarrow$  distance  $d$  et vitesse  $v$ ,  $t = v \cdot d$
- ▶ Concentration de produit de réaction  $\leftarrow$  concentration de réactif et température

- ▶ Néglige souvent l'impact de variables non contrôlables !

## 2. À partir de données : ensemble d'observations pour lesquelles les valeurs des variables explicatives et des variables expliquées sont en général connues

→ Apprentissage supervisé : à partir d'observations pour lesquelles les valeurs des variables explicatives et de la variable expliquée sont connues

## 1. Construction analytique à partir des connaissances du phénomène

- ▶ Exemples :

- ▶ Temps de vol  $t \leftarrow$  distance  $d$  et vitesse  $v$ ,  $t = v \cdot d$
- ▶ Concentration de produit de réaction  $\leftarrow$  concentration de réactif et température

- ▶ Néglige souvent l'impact de variables non contrôlables !

## 2. À partir de données : ensemble d'observations pour lesquelles les valeurs des variables explicatives et des variables expliquées sont en général connues

→ Apprentissage supervisé : à partir d'observations pour lesquelles les valeurs des variables explicatives et de la variable expliquée sont connues

## 1. Construction analytique à partir des connaissances du phénomène

- ▶ Exemples :
  - ▶ Temps de vol  $t \leftarrow$  distance  $d$  et vitesse  $v$ ,  $t = v \cdot d$
  - ▶ Concentration de produit de réaction  $\leftarrow$  concentration de réactif et température
- ▶ Néglige souvent l'impact de variables non contrôlables !

## 2. À partir de données : ensemble d'observations pour lesquelles les valeurs des variables explicatives et des variables expliquées sont en général connues

→ Apprentissage supervisé : à partir d'observations pour lesquelles les valeurs des variables explicatives et de la variable expliquée sont connues

## 1. Construction analytique à partir des connaissances du phénomène

- ▶ Exemples :

- ▶ Temps de vol  $t \leftarrow$  distance  $d$  et vitesse  $v$ ,  $t = v \cdot d$
- ▶ Concentration de produit de réaction  $\leftarrow$  concentration de réactif et température

- ▶ Néglige souvent l'impact de variables non contrôlables !

## 2. À partir de données : ensemble d'observations pour lesquelles les valeurs des variables explicatives et des variables expliquées sont en général connues

→ Apprentissage supervisé : à partir d'observations pour lesquelles les valeurs des variables explicatives et de la variable expliquée sont connues

## 1. Construction analytique à partir des connaissances du phénomène

- ▶ Exemples :

- ▶ Temps de vol  $t \leftarrow$  distance  $d$  et vitesse  $v$ ,  $t = v \cdot d$
- ▶ Concentration de produit de réaction  $\leftarrow$  concentration de réactif et température

- ▶ Néglige souvent l'impact de variables non contrôlables !

## 2. À partir de données : ensemble d'observations pour lesquelles les valeurs des variables explicatives et des variables expliquées sont en général connues

→ Apprentissage supervisé : à partir d'observations pour lesquelles les valeurs des variables explicatives et de la variable expliquée sont connues

# Dans quels cas ai-je besoin de l'apprentissage automatique ?

## Condition 1 : j'ai un problème décisionnel

Je cherche à prédire une variable  $y$  à partir d'un ensemble de variables  $x$ .

## Condition 2 : il n'y a pas de solution analytique

Je ne peux pas construire manuellement une fonction  $f: x \rightarrow y$ .

## Condition 3 : j'ai des données

Je dispose d'une collection d'exemples pour apprendre un modèle.

# Construction d'un modèle à partir des données

- ▶  $x_1, x_2, \dots, x_n$  des observations  $x \in \mathcal{X}$ 
  - ▶ Typiquement, les observations  $x \in \mathbb{R}^d$
- ▶  $y_1, y_2, \dots, y_n$  des vérités terrain,  $y \in \mathcal{Y}$
- ▶ une famille paramétrique de fonctions  $f_\theta \in \mathcal{F}$ , avec  $f: \mathcal{X} \rightarrow \mathcal{Y}$ 
  - ▶  $\theta$  représente les paramètres du modèle
- ▶ une fonction de perte  $\mathcal{L}(y, \hat{y})$

## Problème d'optimisation

Formellement, on cherche à modéliser la fonction  $f$  qui permet de passer d'une observation  $x$  à la variable expliquée  $y$  :

$$\min_{f_\theta \in \mathcal{F}} \sum_{i=1}^n \mathcal{L}(y_i, f_\theta(x_i))$$

Apprendre un modèle = résoudre ce problème d'optimisation

# Construction d'un modèle à partir des données

- ▶  $x_1, x_2, \dots, x_n$  des observations  $x \in \mathcal{X}$ 
  - ▶ Typiquement, les observations  $x \in \mathbb{R}^d$
- ▶  $y_1, y_2, \dots, y_n$  des vérités terrain,  $y \in \mathcal{Y}$
- ▶ une famille paramétrique de fonctions  $f_\theta \in \mathcal{F}$ , avec  $f: \mathcal{X} \rightarrow \mathcal{Y}$ 
  - ▶  $\theta$  représente les paramètres du modèle
- ▶ une fonction de perte  $\mathcal{L}(y, \hat{y})$

## Problème d'optimisation

Formellement, on cherche à modéliser la fonction  $f$  qui permet de passer d'une observation  $x$  à la variable expliquée  $y$  :

$$\min_{f_\theta \in \mathcal{F}} \sum_{i=1}^n \mathcal{L}(y_i, f_\theta(x_i))$$

Apprendre un modèle = résoudre ce problème d'optimisation

# Apprentissage et généralisation

- ▶ Le modèle permet de prendre des décisions pour de futures données
- ▶ Erreur du modèle sur ces futures données = erreur de **généralisation**

## Erreur d'entraînement/erreur de généralisation

En pratique, on veut minimiser l'erreur de généralisation mais on ne peut calculer que l'erreur d'entraînement !

- ▶ Erreur d'apprentissage facilement mesurable sur les données disponibles
- ▶ Données futures inconnues ⇒ erreur de généralisation ne peut pas être mesurée
- ▶ Hypothèse importante : la distribution des données d'apprentissage est représentative de celle des données futures !
- Minimiser l'erreur d'apprentissage = minimiser l'erreur de généralisation ?

# Apprentissage et généralisation

- ▶ Le modèle permet de prendre des décisions pour de futures données
- ▶ Erreur du modèle sur ces futures données = erreur de **généralisation**

## Erreur d'entraînement/erreur de généralisation

En pratique, on veut minimiser l'erreur de généralisation mais on ne peut calculer que l'erreur d'entraînement !

- ▶ Erreur d'apprentissage facilement mesurable sur les données disponibles
- ▶ Données futures inconnues ⇒ erreur de généralisation **ne peut pas être mesurée**
- ▶ **Hypothèse importante** : la distribution des données d'apprentissage est **représentative** de celle des données futures !
- Minimiser l'erreur d'apprentissage = minimiser l'erreur de généralisation ?

# Apprentissage et généralisation

- ▶ Le modèle permet de prendre des décisions pour de futures données
- ▶ Erreur du modèle sur ces futures données = erreur de **généralisation**

## Erreur d'entraînement/erreur de généralisation

En pratique, on veut minimiser l'erreur de généralisation mais on ne peut calculer que l'erreur d'entraînement !

- ▶ Erreur d'apprentissage facilement mesurable sur les données disponibles
- ▶ Données futures inconnues ⇒ erreur de généralisation **ne peut pas être mesurée**
- ▶ **Hypothèse importante** : la distribution des données d'apprentissage est **représentative** de celle des données futures !
  - Minimiser l'erreur d'apprentissage = minimiser l'erreur de généralisation ?

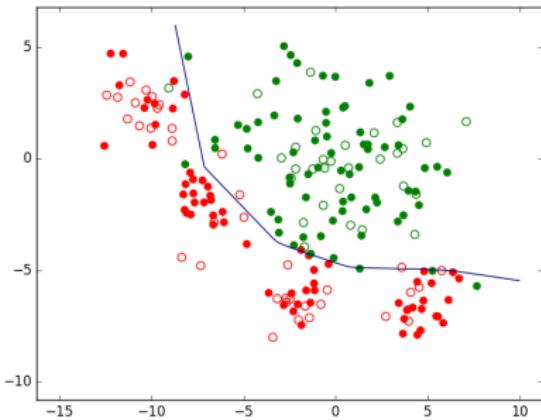
- ▶ Le modèle permet de prendre des décisions pour de futures données
- ▶ Erreur du modèle sur ces futures données = erreur de **généralisation**

## Erreur d'entraînement/erreur de généralisation

En pratique, on veut minimiser l'erreur de généralisation mais on ne peut calculer que l'erreur d'entraînement !

- ▶ Erreur d'apprentissage facilement mesurable sur les données disponibles
- ▶ Données futures inconnues ⇒ erreur de généralisation **ne peut pas être mesurée**
- ▶ **Hypothèse importante** : la distribution des données d'apprentissage est **représentative** de celle des données futures !
- Minimiser l'erreur d'apprentissage = minimiser l'erreur de généralisation ?

# Classification

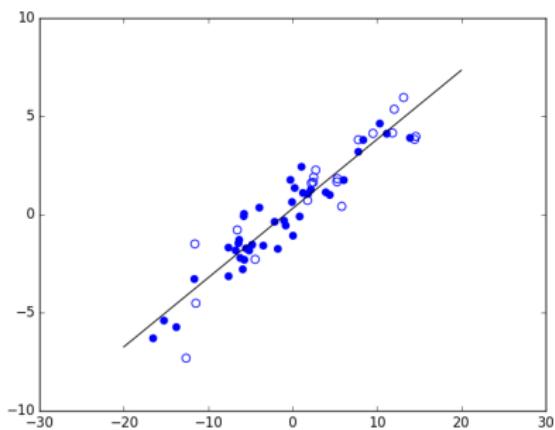


- ▶ Modèle : règle de classement, par ex. frontière de discrimination (trait bleu foncé)
- ▶ Exemple : observations  $(x_1, x_2)$  à deux dimensions (abscisse/ordonnée)
- ▶  $y \in \llbracket 1, k \rrbracket$  pour  $k$  classes

Fonction de coût :

$$\mathcal{L}_{01}(y_i, f_\theta(x_i)) = \begin{cases} 1 & \text{si } y_i \neq f_\theta(x_i) \\ 0 & \text{sinon} \end{cases}$$

⇒ métrique associée = taux de bonne classification (nombre d'erreurs)



- ▶ Modèle : règle de prédiction (trait noir dans la figure)
  - ▶ Par ex.  $y = Ax + b$  pour modèle linéaire
- ▶ Exemple : observations  $x$  à 1 dimension (abscisse), variable à prédire  $y$  en ordonnée
- ▶  $y \in \mathbb{R}$  ou  $\mathbb{R}^p$

Fonction de coût :

$$\mathcal{L}_{\text{MSE}}(y_i, f_\theta(x_i)) = \|y_i - f_\theta(x_i)\|$$

⇒ erreur quadratique moyenne

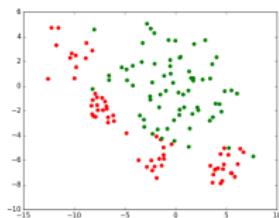
- ▶ Espace d'entrée (variables explicatives) :  $\mathcal{X}$  (par ex.  $\mathbb{R}^p$ )
- ▶ Espace de sortie (variable expliquée) :  $\mathcal{Y}$  (par ex.  $\{-1; 1\}, \mathbb{R}$ )
- ▶ Données à modéliser décrites par variables aléatoires  $(X, Y) \in \mathcal{X} \times \mathcal{Y}$  suivant la distribution inconnue  $P$
- ▶ Exemples

Classement :

$$\mathcal{X} \subset \mathbb{R}^2$$

$$\mathcal{Y} =$$

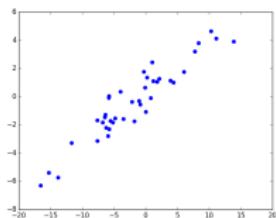
$$\{c_1, c_2\}$$



Régression :

$$\mathcal{X} \subset \mathbb{R}$$

$$\mathcal{Y} \subset \mathbb{R}$$



# Modélisation à partir de données : un cadre plus précis (2)

## Jeu de données

Observations avec information de supervision :  $\mathcal{D}_N = \{(\mathbf{x}_i, y_i)\}_{1 \leq i \leq N}$

- ▶ **Hypothèses** : ces observations sont des tirages identiquement distribués suivant  $P$
- ▶ Sauf cas particuliers les tirages indépendants
  - ▶ Attention, ce n'est pas toujours vrai (séries temporelles, par exemple)

## Objectif

Trouver, dans une famille  $\mathcal{F}$ , une fonction  $f: \mathcal{X} \rightarrow \mathcal{Y}$  qui prédit  $y$  à partir de  $x$

- ▶ Avec l'erreur de généralisation  $R(f) = \mathbb{E}_P[\mathcal{L}(X, Y, f)]$  la plus faible
- ▶  $\mathcal{L}()$  est la fonction de perte (ou d'erreur ou de coût)
- ▶  $\mathbb{E}_P$  est l'espérance par rapport à la distribution inconnue  $P$

## Jeu de données

Observations avec information de supervision :  $\mathcal{D}_N = \{(\mathbf{x}_i, y_i)\}_{1 \leq i \leq N}$

- ▶ **Hypothèses** : ces observations sont des tirages identiquement distribués suivant  $P$
- ▶ Sauf cas particuliers les tirages indépendants
  - ▶ Attention, ce n'est pas toujours vrai (séries temporelles, par exemple)

## Objectif

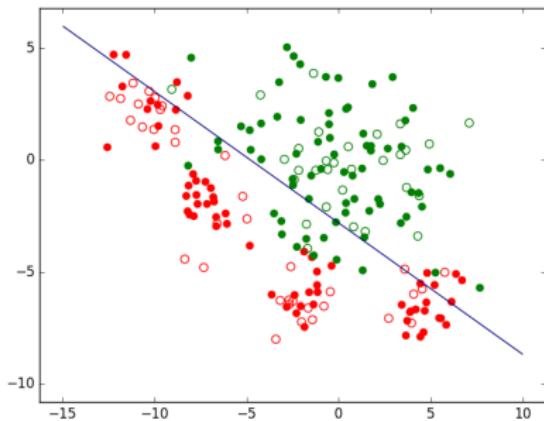
Trouver, dans une famille  $\mathcal{F}$ , une fonction  $f: \mathcal{X} \rightarrow \mathcal{Y}$  qui prédit  $y$  à partir de  $x$

- ▶ Avec l'erreur de généralisation  $R(f) = \mathbb{E}_P[\mathcal{L}(X, Y, f)]$  la plus faible
- ▶  $\mathcal{L}()$  est la fonction de perte (ou d'erreur ou de coût)
- ▶  $\mathbb{E}_P$  est l'espérance par rapport à la distribution inconnue  $P$

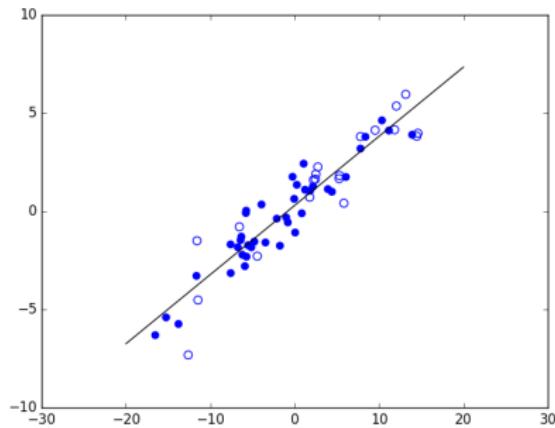
## Modèles linéaires

Prédiction = combinaison linéaire des variables

Classement :  $f(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + w_0$   
 $H(f(\mathbf{x})) \in \{-1, 1\}$

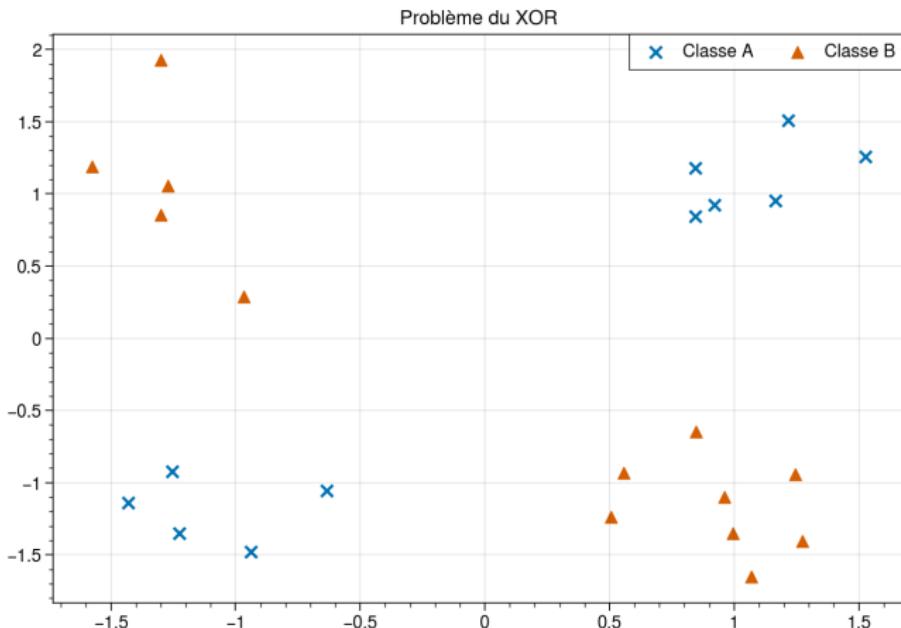


Régression :  $f(x) = w_1 x + w_0$



# Limites des modèles linéaires

Exemple simple de jeu de données non-linéairement séparable

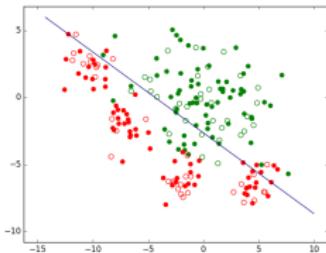


- ▶ Les modèles linéaires peuvent s'avérer insuffisants
  - ▶ Utile de commencer par un modèle linéaire, ne serait-ce que pour pouvoir comparer
- ▶ Modèles polynomiaux de degré borné : la capacité d'approximation (d'une frontière pour le classement, d'une dépendance pour la régression) augmente avec le degré
- ▶ Diverses familles de modèles non linéaires, par ex. réseaux de neurones d'architecture donnée, etc.

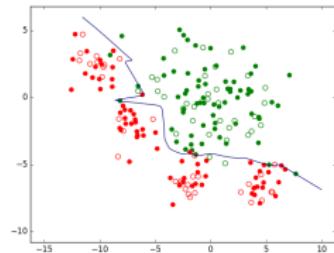
⇒ **Comment choisir ?**

⇒ **Pourquoi ne pas choisir systématiquement la famille de plus grande capacité ?**

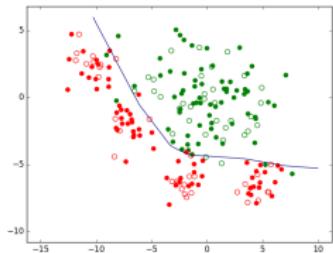
# Comment choisir la famille paramétrique ?



Err. app. 12%  
Err. test 14%



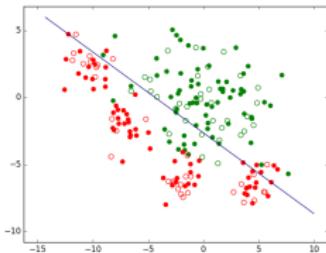
2,3%  
6%



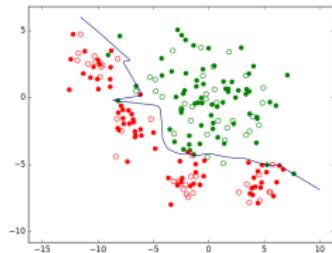
4,5%  
4,6%

- Risque de **sur-apprentissage** (*overfitting*) : erreur d'apprentissage très faible mais erreur de test comparativement élevée
- ⇒ Ce n'est pas avec la capacité la plus grande qu'on obtient la meilleure généralisation
  - Quel lien entre capacité et généralisation ?

# Comment choisir la famille paramétrique ?

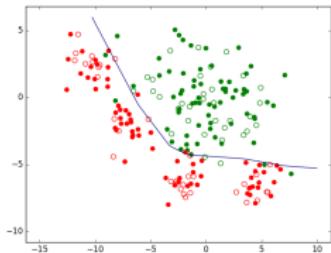


Err. app. 12%  
Err. test 14%



2,3%

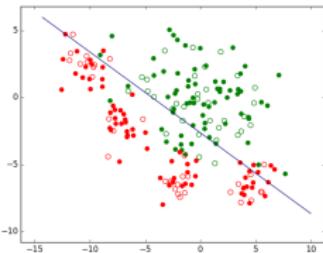
6%



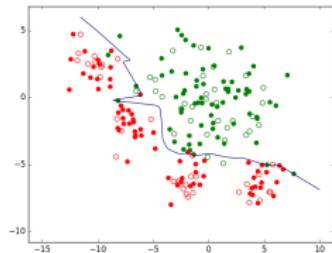
4,5%  
4,6%

- Risque de **sur-apprentissage** (*overfitting*) : erreur d'apprentissage très faible mais erreur de test comparativement élevée
- ⇒ Ce n'est pas avec la capacité la plus grande qu'on obtient la meilleure généralisation
  - Quel lien entre capacité et généralisation ?

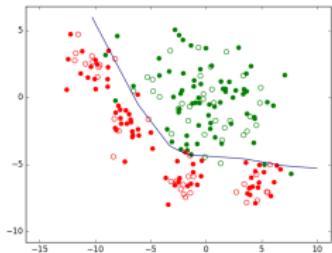
# Comment choisir la famille paramétrique ?



Err. app. 12%  
Err. test 14%



2,3%  
6%



4,5%  
4,6%

- Risque de **sur-apprentissage** (*overfitting*) : erreur d'apprentissage très faible mais erreur de test comparativement élevée
- ⇒ Ce n'est pas avec la capacité la plus grande qu'on obtient la meilleure généralisation
  - Quel lien entre capacité et généralisation ?

## Objectif

Trouver, dans une famille  $\mathcal{F}$  choisie, une fonction (un modèle)  $f: \mathcal{X} \rightarrow \mathcal{Y}$  qui prédit  $y$  à partir de  $x$  et présente l'erreur de généralisation

$R(f) = \mathbb{E}_P[\mathcal{L}(X, Y, f)]$  la plus faible

- ▶  $R(f)$  ne peut pas être évalué car on ne connaît pas la distribution des données  $P$
- ▶ Mais on peut mesurer l'erreur d'apprentissage empirique  
$$R_{\mathcal{D}_N}(f) = \frac{1}{N} \sum_{i=1}^N \mathcal{L}(x_i, y_i, f)$$
- ▶ Quels liens y-a-t-il entre ces deux erreurs ?

## Objectif

Trouver, dans une famille  $\mathcal{F}$  choisie, une fonction (un modèle)  $f: \mathcal{X} \rightarrow \mathcal{Y}$  qui prédit  $y$  à partir de  $x$  et présente l'erreur de généralisation

$R(f) = \mathbb{E}_P[\mathcal{L}(X, Y, f)]$  la plus faible

- ▶  $R(f)$  ne peut pas être évalué car on ne connaît pas la distribution des données  $P$
- ▶ Mais on peut mesurer l'erreur d'apprentissage empirique  
$$R_{\mathcal{D}_N}(f) = \frac{1}{N} \sum_{i=1}^N \mathcal{L}(\mathbf{x}_i, y_i, f)$$
- ▶ Quels liens y-a-t-il entre ces deux erreurs ?

## Objectif

Trouver, dans une famille  $\mathcal{F}$  choisie, une fonction (un modèle)  $f: \mathcal{X} \rightarrow \mathcal{Y}$  qui prédit  $y$  à partir de  $x$  et présente l'erreur de généralisation  
 $R(f) = \mathbb{E}_P[\mathcal{L}(X, Y, f)]$  la plus faible

- ▶  $R(f)$  ne peut pas être évalué car on ne connaît pas la distribution des données  $P$
- ▶ Mais on peut mesurer l'erreur d'apprentissage empirique  
$$R_{\mathcal{D}_N}(f) = \frac{1}{N} \sum_{i=1}^N \mathcal{L}(\mathbf{x}_i, y_i, f)$$
- ▶ Quels liens y-a-t-il entre ces deux erreurs ?

# Analyse des composantes de l'erreur de généralisation

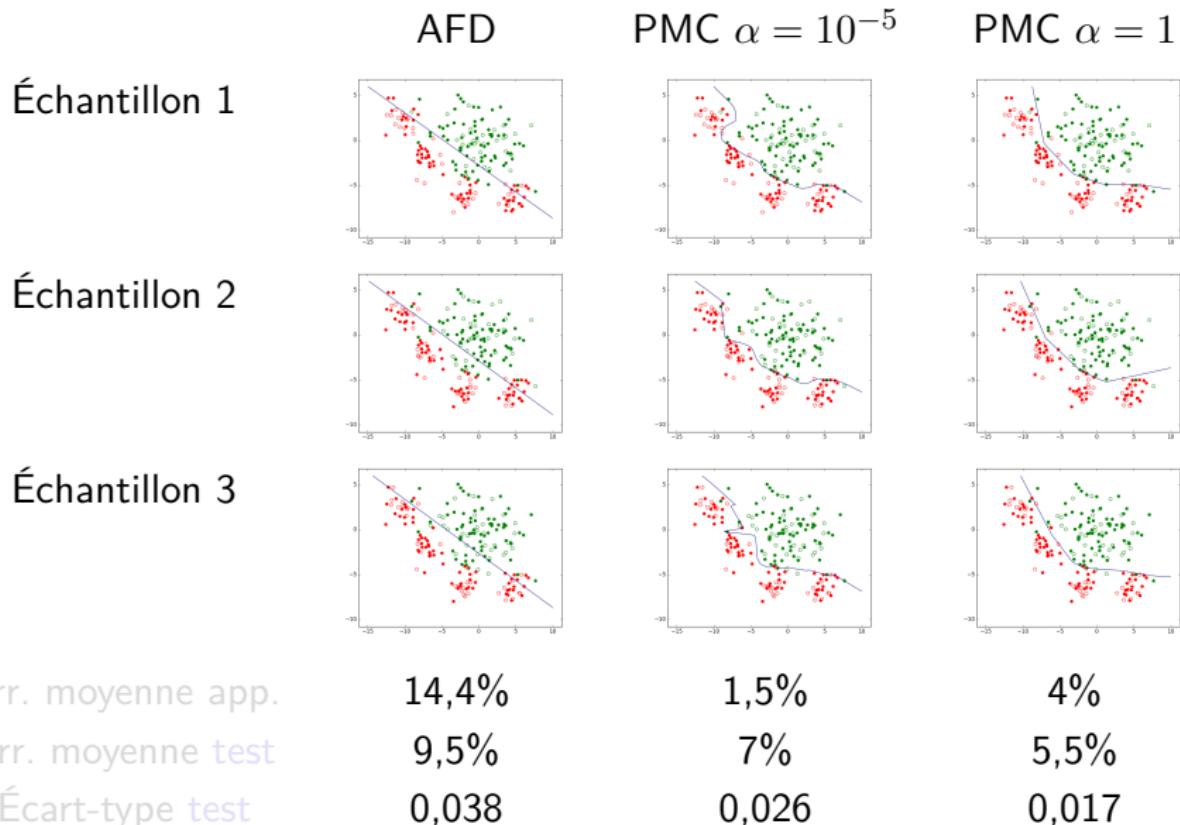
## ► Considérons

- ▶  $f_{\mathcal{D}_N}^*$  la fonction de  $\mathcal{F}$  qui minimise l'erreur empirique  $R_{\mathcal{D}_N}$
- ▶  $f^*$  la fonction de  $\mathcal{F}$  qui minimise l'erreur de généralisation  $R$ , alors

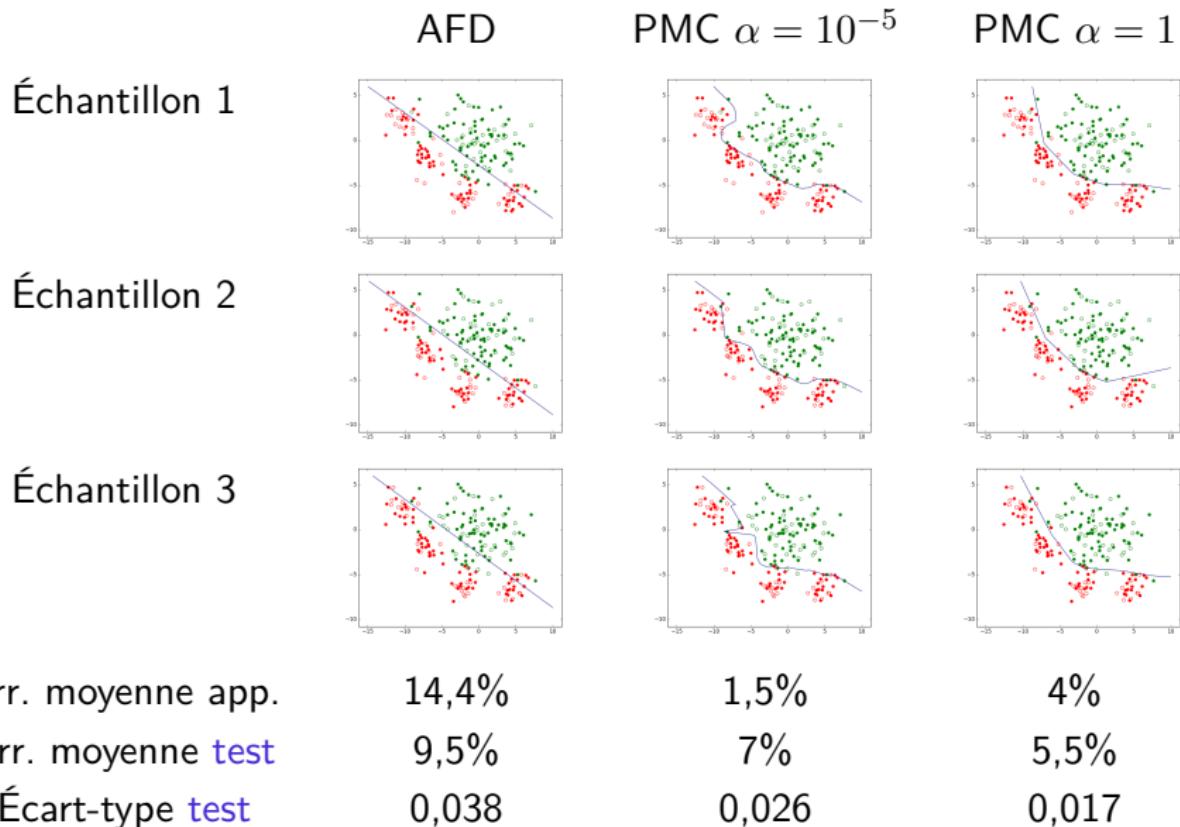
$$R(f_{\mathcal{D}_N}^*) = R^* + [R(f^*) - R^*] + [R(f_{\mathcal{D}_N}^*) - R(f^*)]$$

1.  $R^*$  est l'erreur résiduelle (ou erreur de Bayes), borne inférieure
  - ▶ Strictement positive en présence de bruit : suivant le bruit, à un même  $x$  peuvent correspondre plusieurs valeurs de  $y$
2.  $[R(f^*) - R^*]$  est l'erreur d'approximation ( $\geq 0$ ) car  $\mathcal{F}$  ne contient pas nécessairement la « vraie » fonction de décision
  - ▶ Nulle seulement si  $R^*$  peut être atteint par une fonction de  $\mathcal{F}$
3.  $[R(f_{\mathcal{D}_N}^*) - R(f^*)]$  est l'erreur d'estimation ( $\geq 0$ )
  - ▶ La fonction de  $\mathcal{F}$  qui minimise l'erreur d'apprentissage n'est pas nécessairement celle qui minimise l'erreur de généralisation

# Capacité, erreur d'approximation et erreur d'estimation



# Capacité, erreur d'approximation et erreur d'estimation



## Capacité, erreur d'approximation et erreur d'estimation (2)

Capacité AFD linéaire < capacité PMC  $\alpha = 1$  < capacité PMC  $\alpha = 10^{-5}$

### 1. Famille linéaire (modèles obtenus ici par AFD)

- ▶ Erreur d'apprentissage élevée donc capacité insuffisante pour ce problème
- ⇒ Erreur d'approximation élevée (fort biais)

### 2. Famille définie par PMC 1 couche cachée de 100 neurones, avec coefficient « d'oubli » $\alpha = 10^{-5}$

- ▶ Erreur d'approximation probablement faible car erreur d'apprentissage faible ⇒ capacité suffisante
- ▶ Erreur de test bien plus élevée, variance supérieure à PMC  $\alpha = 1$
- ⇒ Erreur d'estimation élevée

### 3. Famille définie par PMC 1 couche cachée de 100 neurones, avec coefficient « d'oubli » $\alpha = 1$

- ▶ Somme assez faible entre erreur d'approximation et erreur d'estimation, meilleure généralisation que les deux autres familles
- ▶ Erreur de test assez faible et proche de l'erreur d'apprentissage

## Capacité, erreur d'approximation et erreur d'estimation (2)

Capacité AFD linéaire < capacité PMC  $\alpha = 1$  < capacité PMC  $\alpha = 10^{-5}$

### 1. Famille linéaire (modèles obtenus ici par AFD)

- ▶ Erreur d'apprentissage élevée donc capacité insuffisante pour ce problème
  - ⇒ Erreur d'approximation élevée (fort biais)

### 2. Famille définie par PMC 1 couche cachée de 100 neurones, avec coefficient « d'oubli » $\alpha = 10^{-5}$

- ▶ Erreur d'approximation probablement faible car erreur d'apprentissage faible ⇒ capacité suffisante
- ▶ Erreur de test bien plus élevée, variance supérieure à PMC  $\alpha = 1$ 
  - ⇒ Erreur d'estimation élevée

### 3. Famille définie par PMC 1 couche cachée de 100 neurones, avec coefficient « d'oubli » $\alpha = 1$

- ▶ Somme assez faible entre erreur d'approximation et erreur d'estimation, meilleure généralisation que les deux autres familles
- ▶ Erreur de test assez faible et proche de l'erreur d'apprentissage

## Capacité, erreur d'approximation et erreur d'estimation (2)

Capacité AFD linéaire < capacité PMC  $\alpha = 1$  < capacité PMC  $\alpha = 10^{-5}$

### 1. Famille linéaire (modèles obtenus ici par AFD)

- ▶ Erreur d'apprentissage élevée donc capacité insuffisante pour ce problème
- ⇒ Erreur d'approximation élevée (fort **biais**)

### 2. Famille définie par PMC 1 couche cachée de 100 neurones, avec coefficient « d'oubli » $\alpha = 10^{-5}$

- ▶ Erreur d'approximation probablement faible car erreur d'apprentissage faible ⇒ capacité suffisante
- ▶ Erreur de test bien plus élevée, variance supérieure à PMC  $\alpha = 1$
- ⇒ Erreur d'estimation élevée

### 3. Famille définie par PMC 1 couche cachée de 100 neurones, avec coefficient « d'oubli » $\alpha = 1$

- ▶ Somme assez faible entre erreur d'approximation et erreur d'estimation, meilleure généralisation que les deux autres familles
- ▶ Erreur de test assez faible et proche de l'erreur d'apprentissage

## Capacité, erreur d'approximation et erreur d'estimation (2)

Capacité AFD linéaire < capacité PMC  $\alpha = 1$  < capacité PMC  $\alpha = 10^{-5}$

### 1. Famille linéaire (modèles obtenus ici par AFD)

- ▶ Erreur d'apprentissage élevée donc capacité insuffisante pour ce problème
- ⇒ Erreur d'approximation élevée (fort **biais**)

### 2. Famille définie par PMC 1 couche cachée de 100 neurones, avec coefficient « d'oubli » $\alpha = 10^{-5}$

- ▶ Erreur d'approximation probablement faible car erreur d'apprentissage faible ⇒ capacité suffisante
- ▶ Erreur de test bien plus élevée, variance supérieure à PMC  $\alpha = 1$
- ⇒ Erreur d'estimation élevée

### 3. Famille définie par PMC 1 couche cachée de 100 neurones, avec coefficient « d'oubli » $\alpha = 1$

- ▶ Somme assez faible entre erreur d'approximation et erreur d'estimation, meilleure généralisation que les deux autres familles
- ▶ Erreur de test assez faible et proche de l'erreur d'apprentissage

## Capacité, erreur d'approximation et erreur d'estimation (2)

Capacité AFD linéaire < capacité PMC  $\alpha = 1$  < capacité PMC  $\alpha = 10^{-5}$

### 1. Famille linéaire (modèles obtenus ici par AFD)

- ▶ Erreur d'apprentissage élevée donc capacité insuffisante pour ce problème
- ⇒ Erreur d'approximation élevée (fort **biais**)

### 2. Famille définie par PMC 1 couche cachée de 100 neurones, avec coefficient « d'oubli » $\alpha = 10^{-5}$

- ▶ Erreur d'approximation probablement faible car erreur d'apprentissage faible ⇒ capacité suffisante
- ▶ Erreur de test bien plus élevée, variance supérieure à PMC  $\alpha = 1$
- ⇒ Erreur d'estimation élevée

### 3. Famille définie par PMC 1 couche cachée de 100 neurones, avec coefficient « d'oubli » $\alpha = 1$

- ▶ Somme assez faible entre erreur d'approximation et erreur d'estimation, meilleure généralisation que les deux autres familles
- ▶ Erreur de test assez faible et proche de l'erreur d'apprentissage

## Capacité, erreur d'approximation et erreur d'estimation (2)

Capacité AFD linéaire < capacité PMC  $\alpha = 1$  < capacité PMC  $\alpha = 10^{-5}$

### 1. Famille linéaire (modèles obtenus ici par AFD)

- ▶ Erreur d'apprentissage élevée donc capacité insuffisante pour ce problème
- ⇒ Erreur d'approximation élevée (fort **biais**)

### 2. Famille définie par PMC 1 couche cachée de 100 neurones, avec coefficient « d'oubli » $\alpha = 10^{-5}$

- ▶ Erreur d'approximation probablement faible car erreur d'apprentissage faible ⇒ capacité suffisante
- ▶ Erreur de test bien plus élevée, **variance** supérieure à PMC  $\alpha = 1$
- ⇒ Erreur d'estimation élevée

### 3. Famille définie par PMC 1 couche cachée de 100 neurones, avec coefficient « d'oubli » $\alpha = 1$

- ▶ Somme assez faible entre erreur d'approximation et erreur d'estimation, meilleure généralisation que les deux autres familles
- ▶ Erreur de test assez faible et proche de l'erreur d'apprentissage

## Capacité, erreur d'approximation et erreur d'estimation (2)

Capacité AFD linéaire < capacité PMC  $\alpha = 1$  < capacité PMC  $\alpha = 10^{-5}$

### 1. Famille linéaire (modèles obtenus ici par AFD)

- ▶ Erreur d'apprentissage élevée donc capacité insuffisante pour ce problème
- ⇒ Erreur d'approximation élevée (fort **biais**)

### 2. Famille définie par PMC 1 couche cachée de 100 neurones, avec coefficient « d'oubli » $\alpha = 10^{-5}$

- ▶ Erreur d'approximation probablement faible car erreur d'apprentissage faible ⇒ capacité suffisante
- ▶ Erreur de test bien plus élevée, **variance** supérieure à PMC  $\alpha = 1$
- ⇒ Erreur d'estimation élevée

### 3. Famille définie par PMC 1 couche cachée de 100 neurones, avec coefficient « d'oubli » $\alpha = 1$

- ▶ Somme assez faible entre erreur d'approximation et erreur d'estimation, meilleure généralisation que les deux autres familles
- ▶ Erreur de test assez faible et proche de l'erreur d'apprentissage

## Capacité, erreur d'approximation et erreur d'estimation (2)

Capacité AFD linéaire < capacité PMC  $\alpha = 1$  < capacité PMC  $\alpha = 10^{-5}$

1. Famille linéaire (modèles obtenus ici par AFD)
  - ▶ Erreur d'apprentissage élevée donc capacité insuffisante pour ce problème
  - ⇒ Erreur d'approximation élevée (fort **biais**)
2. Famille définie par PMC 1 couche cachée de 100 neurones, avec coefficient « d'oubli »  $\alpha = 10^{-5}$ 
  - ▶ Erreur d'approximation probablement faible car erreur d'apprentissage faible ⇒ capacité suffisante
  - ▶ Erreur de test bien plus élevée, **variance** supérieure à PMC  $\alpha = 1$
  - ⇒ Erreur d'estimation élevée
3. Famille définie par PMC 1 couche cachée de 100 neurones, avec coefficient « d'oubli »  $\alpha = 1$ 
  - ▶ Somme assez faible entre erreur d'approximation et erreur d'estimation, meilleure généralisation que les deux autres familles
  - ▶ Erreur de test assez faible et proche de l'erreur d'apprentissage

## Capacité, erreur d'approximation et erreur d'estimation (2)

Capacité AFD linéaire < capacité PMC  $\alpha = 1$  < capacité PMC  $\alpha = 10^{-5}$

1. Famille linéaire (modèles obtenus ici par AFD)
  - ▶ Erreur d'apprentissage élevée donc capacité insuffisante pour ce problème
  - ⇒ Erreur d'approximation élevée (fort **biais**)
2. Famille définie par PMC 1 couche cachée de 100 neurones, avec coefficient « d'oubli »  $\alpha = 10^{-5}$ 
  - ▶ Erreur d'approximation probablement faible car erreur d'apprentissage faible ⇒ capacité suffisante
  - ▶ Erreur de test bien plus élevée, **variance** supérieure à PMC  $\alpha = 1$
  - ⇒ Erreur d'estimation élevée
3. Famille définie par PMC 1 couche cachée de 100 neurones, avec coefficient « d'oubli »  $\alpha = 1$ 
  - ▶ Somme assez faible entre erreur d'approximation et erreur d'estimation, meilleure généralisation que les deux autres familles
  - ▶ Erreur de test assez faible et proche de l'erreur d'apprentissage

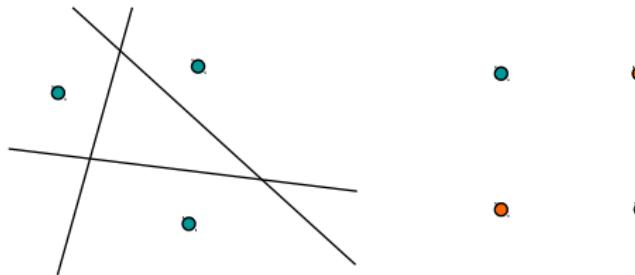
## Capacité, erreur d'approximation et erreur d'estimation (2)

Capacité AFD linéaire < capacité PMC  $\alpha = 1$  < capacité PMC  $\alpha = 10^{-5}$

1. Famille linéaire (modèles obtenus ici par AFD)
  - ▶ Erreur d'apprentissage élevée donc capacité insuffisante pour ce problème
  - ⇒ Erreur d'approximation élevée (fort **biais**)
2. Famille définie par PMC 1 couche cachée de 100 neurones, avec coefficient « d'oubli »  $\alpha = 10^{-5}$ 
  - ▶ Erreur d'approximation probablement faible car erreur d'apprentissage faible ⇒ capacité suffisante
  - ▶ Erreur de test bien plus élevée, **variance** supérieure à PMC  $\alpha = 1$
  - ⇒ Erreur d'estimation élevée
3. Famille définie par PMC 1 couche cachée de 100 neurones, avec coefficient « d'oubli »  $\alpha = 1$ 
  - ▶ Somme assez faible entre erreur d'approximation et erreur d'estimation, meilleure généralisation que les deux autres familles
  - ▶ Erreur de test assez faible et proche de l'erreur d'apprentissage

# Comment mesurer la capacité ?

- ▶ Considérons un ensemble de  $N$  vecteurs  $\{\mathbf{x}_i\}_{1 \leq i \leq N} \in \mathbb{R}^p \rightarrow$ 
  - ▶ il y a  $2^N$  façons différentes de le séparer en 2 parties
- ▶ **Définition** : la famille  $\mathcal{F}$  de fonctions  $f: \mathbb{R}^p \rightarrow \{-1, 1\}$  pulvérise  $\{\mathbf{x}_i\}_{1 \leq i \leq N}$  si toutes les  $2^N$  séparations peuvent être construites avec des fonctions de  $\mathcal{F}$
- ▶ **Définition** (Vapnik-Chervonenkis) :  $\mathcal{F}$  est de VC-dimension  $h$  si elle pulvérise au moins  $h$  vecteurs et aucun ensemble de  $h + 1$  vecteurs
- ▶ Exemple : la VC-dimension des hyperplans de  $\mathbb{R}^p$  est  $h = p + 1$ 
  - ▶ Dans  $\mathbb{R}^2$ , les droites pulvérissent le triplet à gauche mais aucun quadruplet



## Théorème

Soit  $R_{\mathcal{D}_N}(f)$  le risque empirique défini par la fonction de perte  $L_{01}(\mathbf{x}, y, f) = \mathbf{1}_{f(\mathbf{x}) \neq y}$ ; si la VC-dimension de  $\mathcal{F}$  est  $h < \infty$  alors pour toute  $f \in \mathcal{F}$ , avec une probabilité au moins égale à  $1 - \delta$  ( $0 < \delta < 1$ ), on a

$$R(f) \leq R_{\mathcal{D}_N}(f) + \underbrace{\sqrt{\frac{h \left( \log \frac{2N}{h} + 1 \right) - \log \frac{\delta}{4}}{N}}}_{B(N, \mathcal{F})} \quad \text{pour } N > h$$

- ▶  $B(N, \mathcal{F})$  diminue quand  $N \uparrow$ , quand  $h \downarrow$  et quand  $\delta \uparrow$
- ▶  $B(N, \mathcal{F})$  ne fait pas intervenir le nombre de variables
- ▶  $B(N, \mathcal{F})$  ne fait pas intervenir la loi conjointe  $P$
- résultat dans le pire des cas, intéressant d'un point de vue théorique

## Lien entre capacité et généralisation (2)

Conséquences de l'existence d'une borne :

$$R(f) \leq R_{\mathcal{D}_N}(f) + B(N, \mathcal{F})$$

et compte-tenu l'expression de la borne  $B(N, \mathcal{F})$  :

► Famille  $\mathcal{F}$  de capacité trop faible

- ⇒  $B(N, \mathcal{F})$  faible mais  $R_{\mathcal{D}_N}(f)$  (erreur d'apprentissage) élevé(e)
- ⇒ absence de garantie intéressante pour  $R(f)$

► Famille  $\mathcal{F}$  de capacité trop élevée

- ⇒  $R_{\mathcal{D}_N}(f)$  probablement faible mais  $B(N, \mathcal{F})$  élevée
- ⇒ absence de garantie intéressante pour  $R(f)$

► Famille  $\mathcal{F}$  de capacité « adéquate »

- ⇒  $R_{\mathcal{D}_N}(f)$  probablement faible et  $B(N, \mathcal{F})$  plutôt faible
- ⇒ garantie intéressante pour  $R(f)$  !

## Lien entre capacité et généralisation (2)

Conséquences de l'existence d'une borne :

$$R(f) \leq R_{\mathcal{D}_N}(f) + B(N, \mathcal{F})$$

et compte-tenu l'expression de la borne  $B(N, \mathcal{F})$  :

- ▶ Famille  $\mathcal{F}$  de capacité trop faible
  - ⇒  $B(N, \mathcal{F})$  faible mais  $R_{\mathcal{D}_N}(f)$  (erreur d'apprentissage) élevé(e)
  - ⇒ absence de garantie intéressante pour  $R(f)$
- ▶ Famille  $\mathcal{F}$  de capacité trop élevée
  - ⇒  $R_{\mathcal{D}_N}(f)$  probablement faible mais  $B(N, \mathcal{F})$  élevée
  - ⇒ absence de garantie intéressante pour  $R(f)$
- ▶ Famille  $\mathcal{F}$  de capacité « adéquate »
  - ⇒  $R_{\mathcal{D}_N}(f)$  probablement faible et  $B(N, \mathcal{F})$  plutôt faible
  - ⇒ garantie intéressante pour  $R(f)$  !

## Lien entre capacité et généralisation (2)

Conséquences de l'existence d'une borne :

$$R(f) \leq R_{\mathcal{D}_N}(f) + B(N, \mathcal{F})$$

et compte-tenu l'expression de la borne  $B(N, \mathcal{F})$  :

- ▶ Famille  $\mathcal{F}$  de capacité trop faible
  - ⇒  $B(N, \mathcal{F})$  faible mais  $R_{\mathcal{D}_N}(f)$  (erreur d'apprentissage) élevé(e)
  - ⇒ absence de garantie intéressante pour  $R(f)$
- ▶ Famille  $\mathcal{F}$  de capacité trop élevée
  - ⇒  $R_{\mathcal{D}_N}(f)$  probablement faible mais  $B(N, \mathcal{F})$  élevée
  - ⇒ absence de garantie intéressante pour  $R(f)$
- ▶ Famille  $\mathcal{F}$  de capacité « adéquate »
  - ⇒  $R_{\mathcal{D}_N}(f)$  probablement faible et  $B(N, \mathcal{F})$  plutôt faible
  - ⇒ garantie intéressante pour  $R(f)$  !

## Lien entre capacité et généralisation (2)

Conséquences de l'existence d'une borne :

$$R(f) \leq R_{\mathcal{D}_N}(f) + B(N, \mathcal{F})$$

et compte-tenu l'expression de la borne  $B(N, \mathcal{F})$  :

- ▶ Famille  $\mathcal{F}$  de capacité trop faible
  - ⇒  $B(N, \mathcal{F})$  faible mais  $R_{\mathcal{D}_N}(f)$  (erreur d'apprentissage) élevé(e)
  - ⇒ absence de garantie intéressante pour  $R(f)$
- ▶ Famille  $\mathcal{F}$  de capacité trop élevée
  - ⇒  $R_{\mathcal{D}_N}(f)$  probablement faible mais  $B(N, \mathcal{F})$  élevée
  - ⇒ absence de garantie intéressante pour  $R(f)$
- ▶ Famille  $\mathcal{F}$  de capacité « adéquate »
  - ⇒  $R_{\mathcal{D}_N}(f)$  probablement faible et  $B(N, \mathcal{F})$  plutôt faible
  - ⇒ garantie intéressante pour  $R(f)$  !

## Lien entre capacité et généralisation (2)

Conséquences de l'existence d'une borne :

$$R(f) \leq R_{\mathcal{D}_N}(f) + B(N, \mathcal{F})$$

et compte-tenu l'expression de la borne  $B(N, \mathcal{F})$  :

- ▶ Famille  $\mathcal{F}$  de capacité trop faible
  - ⇒  $B(N, \mathcal{F})$  faible mais  $R_{\mathcal{D}_N}(f)$  (erreur d'apprentissage) élevé(e)
  - ⇒ absence de garantie intéressante pour  $R(f)$
- ▶ Famille  $\mathcal{F}$  de capacité trop élevée
  - ⇒  $R_{\mathcal{D}_N}(f)$  probablement faible mais  $B(N, \mathcal{F})$  élevée
  - ⇒ absence de garantie intéressante pour  $R(f)$
- ▶ Famille  $\mathcal{F}$  de capacité « adéquate »
  - ⇒  $R_{\mathcal{D}_N}(f)$  probablement faible et  $B(N, \mathcal{F})$  plutôt faible
  - ⇒ garantie intéressante pour  $R(f)$  !

## Lien entre capacité et généralisation (2)

Conséquences de l'existence d'une borne :

$$R(f) \leq R_{\mathcal{D}_N}(f) + B(N, \mathcal{F})$$

et compte-tenu l'expression de la borne  $B(N, \mathcal{F})$  :

- ▶ Famille  $\mathcal{F}$  de capacité trop faible
  - ⇒  $B(N, \mathcal{F})$  faible mais  $R_{\mathcal{D}_N}(f)$  (erreur d'apprentissage) élevé(e)
  - ⇒ absence de garantie intéressante pour  $R(f)$
- ▶ Famille  $\mathcal{F}$  de capacité trop élevée
  - ⇒  $R_{\mathcal{D}_N}(f)$  probablement faible mais  $B(N, \mathcal{F})$  élevée
  - ⇒ absence de garantie intéressante pour  $R(f)$
- ▶ Famille  $\mathcal{F}$  de capacité « adéquate »
  - ⇒  $R_{\mathcal{D}_N}(f)$  probablement faible et  $B(N, \mathcal{F})$  plutôt faible
  - ⇒ garantie intéressante pour  $R(f)$  !

## Lien entre capacité et généralisation (2)

Conséquences de l'existence d'une borne :

$$R(f) \leq R_{\mathcal{D}_N}(f) + B(N, \mathcal{F})$$

et compte-tenu l'expression de la borne  $B(N, \mathcal{F})$  :

- ▶ Famille  $\mathcal{F}$  de capacité trop faible
  - ⇒  $B(N, \mathcal{F})$  faible mais  $R_{\mathcal{D}_N}(f)$  (erreur d'apprentissage) élevé(e)
  - ⇒ absence de garantie intéressante pour  $R(f)$
- ▶ Famille  $\mathcal{F}$  de capacité trop élevée
  - ⇒  $R_{\mathcal{D}_N}(f)$  probablement faible mais  $B(N, \mathcal{F})$  élevée
  - ⇒ absence de garantie intéressante pour  $R(f)$
- ▶ Famille  $\mathcal{F}$  de capacité « adéquate »
  - ⇒  $R_{\mathcal{D}_N}(f)$  probablement faible et  $B(N, \mathcal{F})$  plutôt faible
  - ⇒ garantie intéressante pour  $R(f)$  !

## Lien entre capacité et généralisation (2)

Conséquences de l'existence d'une borne :

$$R(f) \leq R_{\mathcal{D}_N}(f) + B(N, \mathcal{F})$$

et compte-tenu l'expression de la borne  $B(N, \mathcal{F})$  :

- ▶ Famille  $\mathcal{F}$  de capacité trop faible
  - ⇒  $B(N, \mathcal{F})$  faible mais  $R_{\mathcal{D}_N}(f)$  (erreur d'apprentissage) élevé(e)
  - ⇒ absence de garantie intéressante pour  $R(f)$
- ▶ Famille  $\mathcal{F}$  de capacité trop élevée
  - ⇒  $R_{\mathcal{D}_N}(f)$  probablement faible mais  $B(N, \mathcal{F})$  élevée
  - ⇒ absence de garantie intéressante pour  $R(f)$
- ▶ Famille  $\mathcal{F}$  de capacité « adéquate »
  - ⇒  $R_{\mathcal{D}_N}(f)$  probablement faible et  $B(N, \mathcal{F})$  plutôt faible
  - ⇒ garantie intéressante pour  $R(f)$  !

## Lien entre capacité et généralisation (2)

Conséquences de l'existence d'une borne :

$$R(f) \leq R_{\mathcal{D}_N}(f) + B(N, \mathcal{F})$$

et compte-tenu l'expression de la borne  $B(N, \mathcal{F})$  :

- ▶ Famille  $\mathcal{F}$  de capacité trop faible
  - ⇒  $B(N, \mathcal{F})$  faible mais  $R_{\mathcal{D}_N}(f)$  (erreur d'apprentissage) élevé(e)
  - ⇒ absence de garantie intéressante pour  $R(f)$
- ▶ Famille  $\mathcal{F}$  de capacité trop élevée
  - ⇒  $R_{\mathcal{D}_N}(f)$  probablement faible mais  $B(N, \mathcal{F})$  élevée
  - ⇒ absence de garantie intéressante pour  $R(f)$
- ▶ Famille  $\mathcal{F}$  de capacité « adéquate »
  - ⇒  $R_{\mathcal{D}_N}(f)$  probablement faible et  $B(N, \mathcal{F})$  plutôt faible
  - ⇒ garantie intéressante pour  $R(f)$  !

# Comment minimiser l'erreur d'apprentissage ?

- ▶ Dans une famille paramétrique  $\mathcal{F}$ , un modèle est défini par les valeurs d'un ensemble de paramètres, par ex.
  - ▶ Modèle linéaire pour la régression  $y = ax + b$  :  $a$  et  $b$
  - ▶ Arbre de décision : seuils des tests et choix des variables
  - ▶ Réseaux de neurones : poids des connexions
- Optimisation pour trouver les valeurs qui minimisent l'erreur
  - ▶ Solution analytique directe : cas assez rare, par ex. certains modèles linéaires
  - ▶ Algorithmes itératifs, par ex.
    - ▶ Optimisation quadratique sous contraintes d'inégalité : SVM
    - ▶ Optimisation non linéaire plus générale : PMC, réseaux profonds

# Exemple : régression linéaire

Problème de régression avec  $\mathcal{X} = \mathbb{R}^p$ ,  $\mathcal{Y} = \mathbb{R}$ ,  $\mathcal{D}_N = \{(\mathbf{x}_i, y_i)\}_{1 \leq i \leq N}$

## Régression linéaire

- ▶ Famille paramétrique :  $\hat{y} = w_0 + \sum_{j=1}^p w_j x_{ji}$ , où  $\hat{y}$  est la prédiction
- ▶ Sous forme matricielle :  $\hat{\mathbf{y}} = \mathbf{X}\mathbf{w} = \begin{pmatrix} x_{11} & \dots & x_{p1} & 1 \\ \vdots & \vdots & \vdots & \vdots \\ x_{1n} & \dots & x_{pn} & 1 \end{pmatrix} \begin{pmatrix} w_p \\ \vdots \\ w_0 \end{pmatrix}$
- ▶ On cherche le modèle (défini par le vecteur de paramètres  $\mathbf{w}^*$ ) qui minimise
  - ▶ MRE : l'erreur quadratique totale  $\sum_{i=1}^N (\hat{y}_i - y_i)^2$  sur  $\mathcal{D}_N$
  - ▶ Solution  $\mathbf{w}^* = \mathbf{X}^+ \mathbf{y}$ , où  $\mathbf{X}^+$  est la pseudo-inverse Moore-Penrose de  $\mathbf{X}$
  - ▶ Si  $\mathbf{X}^T \mathbf{X}$  est inversible, alors  $\mathbf{X}^+ = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T$

# Exemple : régression linéaire

Problème de régression avec  $\mathcal{X} = \mathbb{R}^p$ ,  $\mathcal{Y} = \mathbb{R}$ ,  $\mathcal{D}_N = \{(\mathbf{x}_i, y_i)\}_{1 \leq i \leq N}$

## Régression linéaire

- ▶ Famille paramétrique :  $\hat{y} = w_0 + \sum_{j=1}^p w_j x_{ji}$ , où  $\hat{y}$  est la prédiction
- ▶ Sous forme matricielle :  $\hat{\mathbf{y}} = \mathbf{X}\mathbf{w} = \begin{pmatrix} x_{11} & \dots & x_{p1} & 1 \\ \vdots & \vdots & \vdots & \vdots \\ x_{1n} & \dots & x_{pn} & 1 \end{pmatrix} \begin{pmatrix} w_p \\ \vdots \\ w_0 \end{pmatrix}$
- ▶ On cherche le modèle (défini par le vecteur de paramètres  $\mathbf{w}^*$ ) qui minimise
  - ▶ MRER : la somme entre l'erreur quadratique sur  $\mathcal{D}_N$  et un terme de régularisation, par exemple la régularisation Tikhonov :  
$$\sum_{i=1}^N (\hat{y}_i - y_i)^2 + \|\mathbf{w}\|_2^2$$
  - Solution  $\mathbf{w}^* = (\mathbf{X}^T \mathbf{X} + \mathbf{I}_{p+1})^{-1} \mathbf{X}^T \mathbf{y}$ , où  $\mathbf{I}_{p+1}$  est la matrice unité

- ▶ Construire un modèle décisionnel à partir de données : supervision nécessaire
- ▶ Objectif : obtenir le modèle qui présente la meilleure généralisation
- ▶ Estimer la généralisation : non à partir de l'erreur d'apprentissage
- ▶ Chercher le bon compromis entre minimisation de la capacité de la famille de modèles et minimisation de l'erreur d'apprentissage

## Évaluation de modèles

---

# Comment estimer l'erreur de généralisation

## Données de test

Par l'erreur sur des données de **test**, non utilisées pour l'apprentissage (par ex., 30% des données sont mises à l'écart pour le test)

- ▶ Apprentissage (estimation) du modèle sur les données d'apprentissage
- ▶ Estimation de l'erreur de généralisation sur les données de test
  - ▶ Réduit le nombre de données utilisées pour l'apprentissage
  - ▶ Variance élevée : autre partitionnement  $\implies$  estimation différente

## Validation croisée

*Cross-validation* : plusieurs partitions apprentissage | test

- $\Rightarrow$  estimateur de variance plus faible,
- ... tout en utilisant mieux les données disponibles !
- ▶ mais plusieurs modèles à entraîner

# Comment estimer l'erreur de généralisation

## Données de test

Par l'erreur sur des données de **test**, non utilisées pour l'apprentissage (par ex., 30% des données sont mises à l'écart pour le test)

- ▶ Apprentissage (estimation) du modèle sur les données d'apprentissage
- ▶ Estimation de l'erreur de généralisation sur les données de test
  - ▶ Réduit le nombre de données utilisées pour l'apprentissage
  - ▶ Variance élevée : autre partitionnement  $\implies$  estimation différente

## Validation croisée

*Cross-validation* : plusieurs partitions apprentissage | test

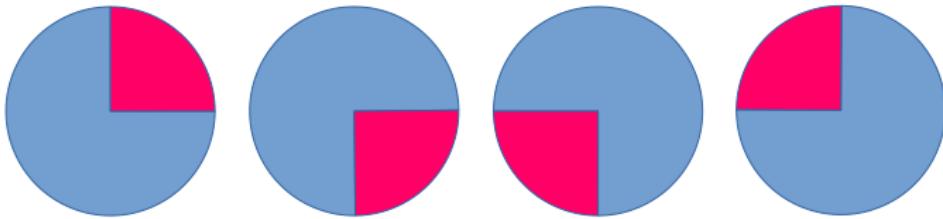
- $\Rightarrow$  estimateur de variance plus faible,
- ... tout en utilisant mieux les données disponibles !
- ▶ mais plusieurs modèles à entraîner

## 1. Méthodes exhaustives :

- ▶ *Leave  $p$  out* (LPO) :  $N - p$  données pour l'apprentissage et  $p$  pour la validation  $\Rightarrow C_N^p$  découpages donc  $C_N^p$  modèles  $\Rightarrow$  coût excessif
- ▶ *Leave one out* (LOO) :  $N - 1$  données pour l'apprentissage et 1 pour la validation  $\Rightarrow C_N^1 = N$  découpages (donc  $N$  modèles)  $\Rightarrow$  coût élevé

## 2. Méthodes non exhaustives :

- ▶ *k-fold* : partitionnement fixé des  $N$  données en  $k$  parties, apprentissage sur  $k - 1$  parties et validation sur la  $k$ -ème  $\Rightarrow k$  modèles seulement



- ▶ Échantillonnage répété (*shuffle and split*) : échantillon aléatoire de  $p$  données pour le test (les autres  $N - p$  pour l'apprentissage), on répète  $k$  fois  $\Rightarrow k$  modèles

# Validation croisée : quelle méthode préférer ?

- ▶ LPO très rarement employée car excessivement coûteuse
- ▶ LOO vs  $k$ -fold :  $k$ -fold préférée en général
  - ▶ LOO plus coûteuse car  $N \gg k$
  - ▶ Variance en général supérieure pour LOO
  - ▶ Estimation  $k$ -fold pessimiste car chaque modèle apprend sur  $\frac{k-1}{k}N < N - 1$  données
- ▶ *Shuffle and split* vs  $k$ -fold
  - ▶ Pour  $k$ -fold le nombre de modèles ( $k$ ) est lié à la proportion de données de test ( $1/k$ ), *shuffle and split* moins contraignante
  - ▶ Pour *shuffle and split* certaines données ne sont dans aucun échantillon alors que d'autres sont dans plusieurs échantillons
- ▶ Quelle que soit la méthode, tous les partitionnements peuvent être explorés en parallèle (sur processeurs multi-coeur ou plateformes distribuées)

# Validation croisée : précautions à prendre

- ▶ Classification avec classes déséquilibrées : pour s'assurer de conserver les rapports entre les classes dans tous les découpages, utiliser
  - ▶ Un partitionnement adapté pour *k-fold* (par ex. `StratifiedKFold` dans `Scikit-learn`)
  - ▶ Un échantillonnage *stratifié* pour *shuffle and split* (par ex. `StratifiedShuffleSplit` dans `Scikit-learn`)
  - ▶ LOO peut être employée telle quelle
- ▶ Observations qui ne sont pas indépendantes
  - ▶ Séries temporelles : les observations successives sont corrélées, le découpage doit être fait par séquences sur les observations ordonnées et non après *shuffle* sur les observations individuelles
  - ▶ Données groupées : dans un même groupe, les observations ne sont pas indépendantes; les données de test doivent provenir de groupes différents de ceux dont sont issues les données d'apprentissage

# Validation croisée : précautions à prendre

- ▶ Classification avec classes déséquilibrées : pour s'assurer de conserver les rapports entre les classes dans tous les découpages, utiliser
  - ▶ Un partitionnement adapté pour *k-fold* (par ex. `StratifiedKFold` dans `Scikit-learn`)
  - ▶ Un échantillonnage *stratifié* pour *shuffle and split* (par ex. `StratifiedShuffleSplit` dans `Scikit-learn`)
  - ▶ LOO peut être employée telle quelle
- ▶ Observations qui ne sont pas indépendantes
  - ▶ Séries temporelles : les observations successives sont corrélées, le découpage doit être fait par séquences sur les observations ordonnées et non après *shuffle* sur les observations individuelles
  - ▶ Données groupées : dans un même groupe, les observations ne sont pas indépendantes; les données de test doivent provenir de groupes différents de ceux dont sont issues les données d'apprentissage

## Validation croisée : précautions à prendre

- ▶ Classification avec classes déséquilibrées : pour s'assurer de conserver les rapports entre les classes dans tous les découpages, utiliser
  - ▶ Un partitionnement adapté pour *k-fold* (par ex. `StratifiedKFold` dans `Scikit-learn`)
  - ▶ Un échantillonnage *stratifié* pour *shuffle and split* (par ex. `StratifiedShuffleSplit` dans `Scikit-learn`)
  - ▶ LOO peut être employée telle quelle
- ▶ Observations qui ne sont pas indépendantes
  - ▶ Séries temporelles : les observations successives sont corrélées, le découpage doit être fait par séquences sur les observations ordonnées et non après *shuffle* sur les observations individuelles
  - ▶ Données groupées : dans un même groupe, les observations ne sont pas indépendantes; les données de test doivent provenir de groupes différents de ceux dont sont issues les données d'apprentissage

## Validation croisée : précautions à prendre

- ▶ Classification avec classes déséquilibrées : pour s'assurer de conserver les rapports entre les classes dans tous les découpages, utiliser
  - ▶ Un partitionnement adapté pour *k-fold* (par ex. `StratifiedKFold` dans `Scikit-learn`)
  - ▶ Un échantillonnage *stratifié* pour *shuffle and split* (par ex. `StratifiedShuffleSplit` dans `Scikit-learn`)
  - ▶ LOO peut être employée telle quelle
- ▶ Observations qui ne sont pas indépendantes
  - ▶ Séries temporelles : les observations successives sont corrélées, le découpage doit être fait par séquences sur les observations ordonnées et non après *shuffle* sur les observations individuelles
  - ▶ Données groupées : dans un même groupe, les observations ne sont pas indépendantes; les données de test doivent provenir de groupes différents de ceux dont sont issues les données d'apprentissage

## Validation croisée : précautions à prendre

- ▶ Classification avec classes déséquilibrées : pour s'assurer de conserver les rapports entre les classes dans tous les découpages, utiliser
  - ▶ Un partitionnement adapté pour *k-fold* (par ex. `StratifiedKFold` dans `Scikit-learn`)
  - ▶ Un échantillonnage *stratifié* pour *shuffle and split* (par ex. `StratifiedShuffleSplit` dans `Scikit-learn`)
  - ▶ LOO peut être employée telle quelle
- ▶ Observations qui ne sont pas indépendantes
  - ▶ Séries temporelles : les observations successives sont corrélées, le découpage doit être fait *par séquences* sur les observations ordonnées et non après *shuffle* sur les observations individuelles
  - ▶ Données groupées : dans un même groupe, les observations ne sont pas indépendantes ; les données de test doivent provenir de groupes différents de ceux dont sont issues les données d'apprentissage

## Validation croisée : précautions à prendre

- ▶ Classification avec classes déséquilibrées : pour s'assurer de conserver les rapports entre les classes dans tous les découpages, utiliser
  - ▶ Un partitionnement adapté pour *k-fold* (par ex. StratifiedKFold dans Scikit-learn)
  - ▶ Un échantillonnage *stratifié* pour *shuffle and split* (par ex. StratifiedShuffleSplit dans Scikit-learn)
  - ▶ LOO peut être employée telle quelle
- ▶ Observations qui ne sont pas indépendantes
  - ▶ Séries temporelles : les observations successives sont corrélées, le découpage doit être fait *par séquences* sur les observations ordonnées et non après *shuffle* sur les observations individuelles
  - ▶ Données groupées : dans un même groupe, les observations ne sont pas indépendantes ; les données de test doivent provenir de groupes différents de ceux dont sont issues les données d'apprentissage

## Validation croisée : précautions à prendre

- ▶ Classification avec classes déséquilibrées : pour s'assurer de conserver les rapports entre les classes dans tous les découpages, utiliser
  - ▶ Un partitionnement adapté pour *k-fold* (par ex. StratifiedKFold dans Scikit-learn)
  - ▶ Un échantillonnage *stratifié* pour *shuffle and split* (par ex. StratifiedShuffleSplit dans Scikit-learn)
  - ▶ LOO peut être employée telle quelle
- ▶ Observations qui ne sont pas indépendantes
  - ▶ Séries temporelles : les observations successives sont corrélées, le découpage doit être fait *par séquences* sur les observations ordonnées et non après *shuffle* sur les observations individuelles
  - ▶ Données groupées : dans un même groupe, les observations ne sont pas indépendantes ; les données de test doivent provenir de groupes *differents* de ceux dont sont issues les données d'apprentissage

## Évaluation pour la classification : matrice de confusion

Prédiction \ Vérité terrain	Classe 1	Classe 2	Classe 3
Classe 1	10	2	3
Classe 2	7	15	0
Classe 3	3	5	10

- ▶  $k^{\text{e}}$  ligne,  $l^{\text{e}}$  colonne : combien d'observations de la classe  $l$  ont été prédites comme étant de la classe  $k$  ?
- ▶  $C[k, l] = |\{x_i \text{ tel que } z_i = l \text{ et } f_{\theta}(x_i) = k\}|$

**Exactitude (*overall accuracy*) : % d'observations bien classées**

Avec  $K$  le nombre de classes, l'exactitude s'obtient par :

$$OA = \frac{\sum_{k=1}^K C[k, k]}{\sum_{k=1}^K \sum_{l=1}^K C[k, l]} \in [0, 1]$$

# Évaluation pour la classification : matrice de confusion

Prédiction \ Vérité terrain	Classe 1	Classe 2	Classe 3
Classe 1	10	2	3
Classe 2	7	15	0
Classe 3	3	5	10

## Précision et rappel

- ▶ Précision : fraction de prédictions correctes pour la classe  $k$
- ▶ Rappel : fraction de la classe  $k$  correctement prédite

$$P[k] = \frac{C[k, k]}{\sum_{i=1}^K C[i, k]}, R[k] = \frac{C[k, k]}{\sum_{i=1}^K C[k, i]}$$

## Évaluation pour la classification : matrice de confusion

Prédiction \ Vérité terrain	Classe 1	Classe 2	Classe 3
Classe 1	10	2	3
Classe 2	7	15	0
Classe 3	3	5	10

### Score F1

Moyenne harmonique entre précision et rappel

$$F1[k] = 2 \cdot \frac{P[k]R[k]}{P[k] + R[k]}$$

$$\text{Macro-F1} = \frac{1}{K} \sum_{i=1}^K F1[i]$$

# Matrice de confusion binaire

Dans le cas spécifique d'un problème à seulement deux classes (les positifs et les négatifs) :

Prédiction \ Vérité terrain	Classe $\oplus$	Classe $\ominus$
Classe $\oplus$	Vrais positifs	Faux positifs
Classe $\ominus$	Faux négatifs	Vrais négatifs

## Précision et rappel

En notant  $tp$  les vrais positifs,  $fp$  les faux positifs et  $fn$  les faux négatifs :

$$P = \frac{tp}{tp + fp}, \quad R = \frac{tp}{tp + fn}$$

# Évaluation pour la régression

## Coefficient de détermination linéaire $R^2$

$$R^2 = 1 - \frac{\sum_{i=1}^n (y_i - f_\theta(x_i))^2}{(y_i - \bar{y})^2} \in ]-\infty, 1]$$

où  $\bar{y} = \frac{1}{n} \sum_{i=1}^n y_i$  est la moyenne des vérités terrain.

- ▶ Un modèle qui renvoie systématiquement la moyenne a un  $R^2 = 0$
- ▶ Un modèle moins bon a un coefficient  $R^2 < 0$

## Erreur quadratique moyenne

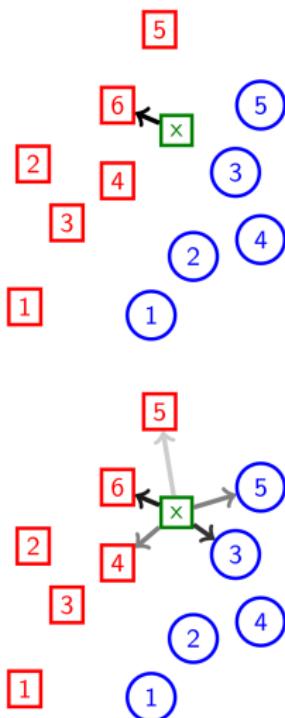
$$\text{RMSE} = \frac{1}{n} \sum_{i=1}^n \sqrt{(y_i - f_\theta(x_i))^2}$$

- ▶ Dans certains cas, on préfère l'erreur absolue qui est moins sensible aux données aberrantes.

## Méthodes de base

---

# $k$ plus proches voisins



- ▶  $\mathcal{D} = \{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\}$
- ▶ Classification :
  - ▶ On associe à  $x$  la classe  $y$  majoritaire parmi ses  $k$  voisins les plus proches. Pour  $k = 1$  :

$$f(x) = y_i \text{ avec } i = \arg \min_{j \in [1, n]} \|x_j - x\|$$

- ▶ Régression :
  - ▶ On associe à  $x$  la moyenne  $\hat{y}$  des valeurs  $\hat{y}_i$  de ses  $k$  voisins les plus proches :

$$f(x) = \frac{1}{k} \sum_{i \in i_1, \dots, i_k} y_i$$

$$\text{avec } (i_1, i_2, \dots, i_k) = \arg \min_{j \in [1, n]} \|x_j - x\|$$

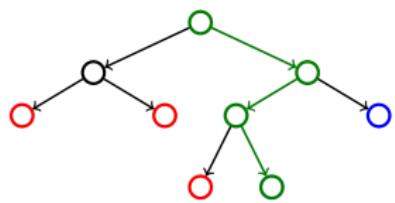
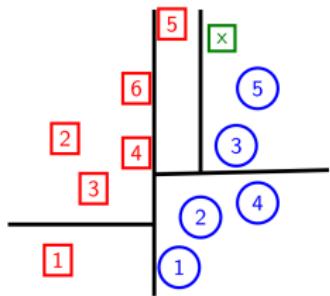
Variante : pondérer les voisins selon leur distance à  $x$

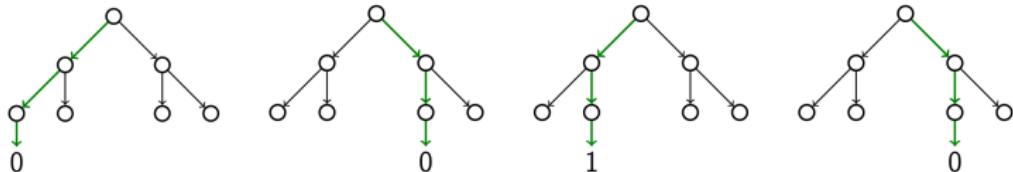
- ▶ Cascade de décisions binaires sur  $x$  : la variable  $x[i]$  est-elle supérieure ou inférieure à un certain seuil ?

$$x[i] >? \tau$$

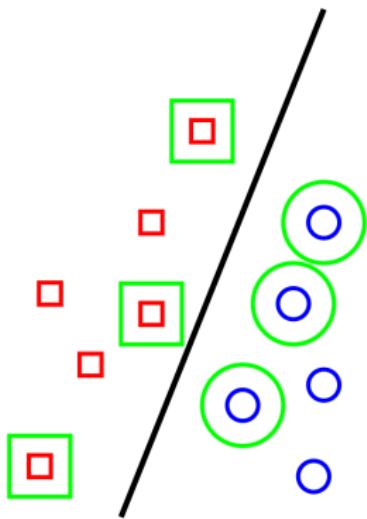
- ▶ Possible de l'implémenter à la main à partir de connaissances expertes, ou de l'apprendre automatiquement

- ▶ On détermine le seuil  $\tau$  du test de sorte à maximiser la séparation dans l'enfant gauche et l'enfant droite de l'arbre
- ⇒ augmenter la pureté d'un noeud  $\rightarrow$  maximiser le nombre d'observations  $x$  qui ont la même classe  $y$
- ▶ Lorsque un noeud ne représente plus que quelques observations, on s'arrête  $\rightarrow$  feuille





- ▶ Ensemble de  $m$  arbres de décision appris sur des sous-ensembles aléatoires du jeu de données
- ▶ Prédiction de la forêt aléatoire = classe prédite par la majorité des arbres individuels
- ▶ Principe de l'apprentissage *par ensemble* : de nombreux classifieurs faibles peuvent former un classifieur fort

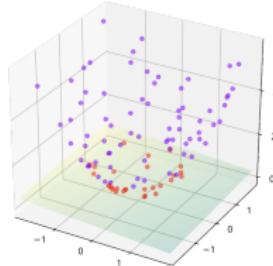
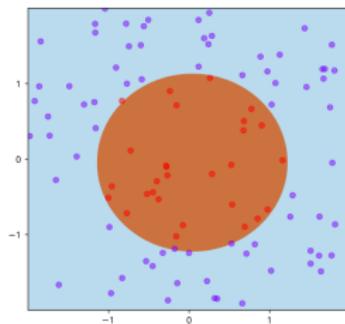


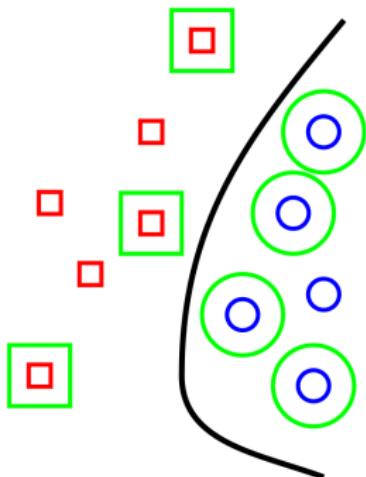
- ▶ On cherche un *hyperplan* séparateur entre deux classes = frontière linéaire
- ▶ Contrainte : maximiser la marge entre le plan séparateur et les observations les plus proches du plan
- ▶ Le plan de séparation est déterminé uniquement par les points les plus proches de la frontière de décision : ce sont les *vecteurs de support*
- ▶ Hyperparamètre  $C$  : pénalise la présence d'observations mal classées, c'est-à-dire du mauvais côté de la frontière

# Astuce du noyau

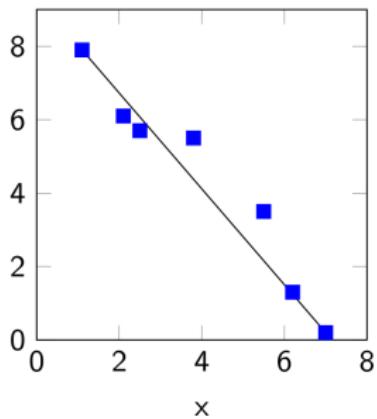
Transformer un problème non-linéaire en modèle linéaire grâce à une projection  $\varphi$  bien choisie

- ▶  $\varphi(x)$  est souvent difficile (voire impossible) à calculer...
- ▶ Mais une SVM a seulement besoin du produit scalaire !
- ▶ Il suffit donc d'avoir une fonction noyau  $K(x, y) = \varphi(x) \cdot \varphi(y)$  qui calcule ce même produit scalaire.





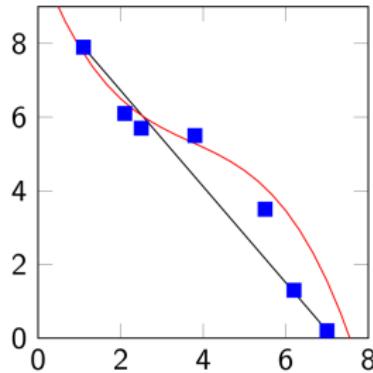
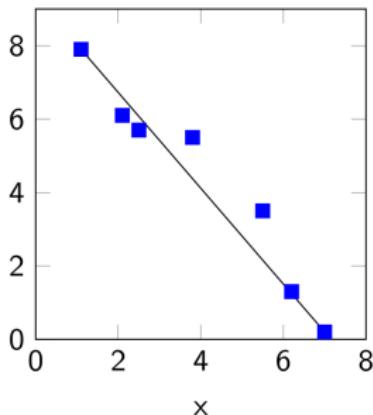
- ▶ Extension des SVM au cas où les données ne sont pas *linéairement séparables*
- ▶ Noyau : fonction de comparaison ( $\sim$  produit scalaire)
  - ▶  $K(x, x') = \exp\left(-\frac{\|x-x'\|^2}{2\sigma^2}\right)$



- ▶ Cherche un (hyper)plan qui minimise l'erreur aux moindres carrés lorsque l'on projette les observations sur ce plan
- ▶ En notant  $\theta \in \mathbb{R}^d$  les paramètres du plan en dimension  $d$  et  $b$  le biais :

$$\theta^*, b^* = \arg \min_{\theta \in \mathbb{R}^d, b \in \mathbb{R}} \sum_{i=1}^n \|\theta x_i + b - y_i\|$$

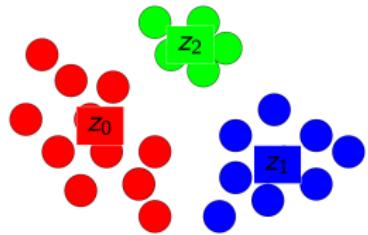
- ▶ Possibilité de l'étendre en incluant des variables additionnelles, par exemple les monômes des variables existantes pour la régression polynomiale



- ▶ Cherche un (hyper)plan qui minimise l'erreur aux moindres carrés lorsque l'on projette les observations sur ce plan
- ▶ En notant  $\theta \in \mathbb{R}^d$  les paramètres du plan en dimension  $d$  et  $b$  le biais :

$$\theta^*, b^* = \arg \min_{\theta \in \mathbb{R}^d, b \in \mathbb{R}} \sum_{i=1}^n \|\theta x_i + b - y_i\|$$

- ▶ Possibilité de l'étendre en incluant des variables additionnelles, par exemple les monômes des variables existantes pour la régression polynomiale



- ▶ K-means est un algorithme de classification automatique *non-supervisé* (“clustering”)
- ▶ À partir d’observations  $\mathcal{D} = (x_1, x_2, \dots, x_n)$ , l’objectif est de déterminer  $k$  groupes  $C_p$  tels que :
- ▶ les groupes sont des sous-ensemble du jeu de données

$$\forall p \in [1; k], \quad C_p \subset \mathcal{D}$$

- ▶ les groupes sont mutuellement exclusifs

$$\forall (p, q) \in [1; k]^2, \quad p \neq q \implies C_p \cap C_q = \emptyset$$

- ▶ si deux observations appartiennent au même groupe, alors elles sont proches

$$\forall (x_i, x_j) \in \mathcal{D}^2, \quad x_i \in C_p \text{ et } x_j \in C_p \implies \|x_i - x_j\| \leq \tau$$

- ▶ On cherche à minimiser la somme des variances intra-groupe, c'est-à-dire :

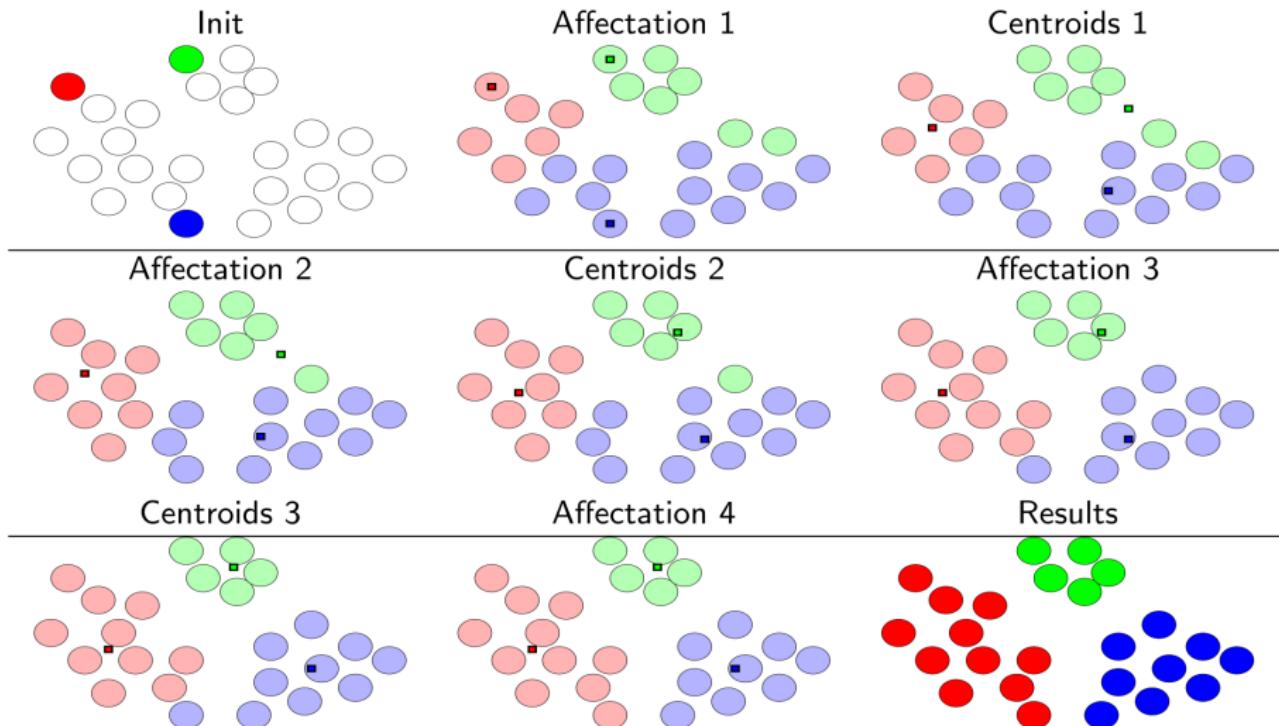
$$C_1, \dots, C_k = \arg \min \sum_{i=1}^k \sum_{x_j \in C_i} \|x_j - m_i\|^2$$

avec  $m_i$  le barycentre (centroïde) du groupe  $C_i$ .

- ▶ Pour ce faire, l'algorithme des k-moyennes agit de la façon suivante :

1. tirer  $k$  observations au hasard qui représentent  $m_1^{(0)}, \dots, m_k^{(0)}$  les centres initiaux
2. Jusqu'à ce que les centres ne se déplacent plus :
  - 2.1 Ajouter chaque observation au groupe dont le centroïde est le plus proche :  $\text{groupe}(x_i) = \arg \min_j \|x_i - m_j\|^2$
  - 2.2 Mettre à jour les centres :  $m_i^{(t+1)} = \frac{1}{|C_i^{(t)}|} \sum_{x_j \in C_i} x_j$

# k-means : illustration

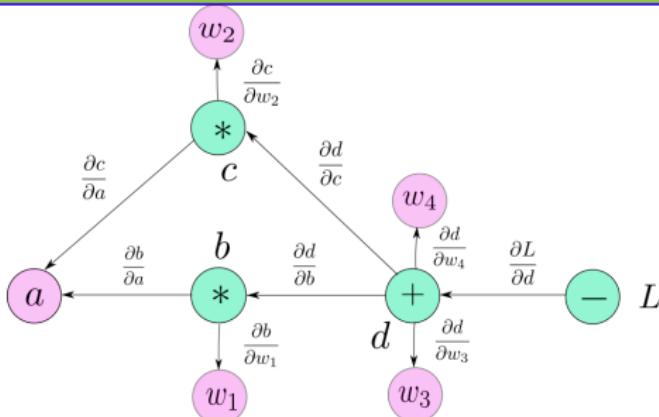


- ▶ On écrit  $f_\theta(x)$  comme une suite d'opérations élémentaires (multiplications matricielles, convolutions, fonctions de transfert) *différentiables* paramétrées par  $\theta$
- ▶ Sous réserve que la fonction de coût  $\mathcal{L}$  soit différentiable par rapport aux paramètres  $\theta$ , optimisation par descente de gradient :

$$\theta_{t+1} \leftarrow \theta_t + \nabla_{\theta_t} \mathcal{L}(y_i, f_{\theta_t}(x_i))$$

- ▶ Coûteux en calcul mais très grande capacité d'approximation !

# Réseaux de neurones : graphe de calcul différentiable



$$b = w_1 \cdot a$$

$$c = w_2 \cdot a$$

$$d = w_3 \cdot b + w_4 \cdot c$$

$$L = 10 - d$$

Les bibliothèques d'apprentissage profond implémentent automatiquement la **rétropropagation**, c'est-à-dire le calcul des dérivées partielles de l'erreur par rapport aux paramètres du modèle grâce à la dérivation des fonctions composées :

$$\frac{\partial L}{\partial w} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial w}$$

## Conclusion

---

- ▶ Ai-je besoin de l'apprentissage automatique ?
- ▶ Mon problème se prête-t-il à l'apprentissage automatique ?
- ▶ Ai-je des données ?
- ▶ De quelle tâche s'agit-il ? Classification, régression ? Autre chose ?
- ▶ Quelle fonction de perte puis-je utiliser ? Quelle métrique d'évaluation ?  
Quelles familles de modèles ?
- ▶ Entraîner les modèles
- ▶ Évaluer les modèles sur un ensemble de test séparé pour évaluer  
l'erreur de généralisation, idéalement par validation croisée

# Quelques références bibliographiques

- ▶ C.-A. Azencott, *Introduction au Machine Learning*, Éditions Eyrolles, 2018.
- ▶ A. Géron, *Machine Learning avec scikit-learn ; Deep Learning avec Keras*, Éditions Dunod, 2021.
- ▶ I. Goodfellow, Y. Bengio, A. Courville. *Deep Learning*. MIT Press, 2016. <http://www.deeplearningbook.org>. [?]
- ▶ A. Amor, L. Estève, O. Grisel, G. Lemaître, G. Varoquaux, T. Schmitt, MOOC Machine learning with scikit-learn, France Université Numérique, <https://www.fun-mooc.fr/fr/cours/machine-learning-python-scikit-learn/>.