

(Statistical) Machine Learning

Emilie Kaufmann

CNRS & CRIStAL, Inria



emilie.kaufmann@univ-lille.fr

Cours 2018/2019

Organisation du cours:

- ▶ 4 séances de cours/TD (19 et 26 septembre, 10 et 26 octobre)
- ▶ 2 séances de cours/TP (24 octobre et 7 novembre)

Validation:

$$\text{Note Finale} = \frac{E + CC}{2}$$

- ▶ E: note de l'examen final
- ▶ CC: note de contrôle continu (rapport de TPs)

Site web du cours:

<http://chercheurs.lille.inria.fr/ekaufman/ML.html>

Introduction: Les succès de l'apprentissage automatique

L'apprentissage supervisé

But: à partir d'une base de données d'exemples "étiquetés", apprendre à étiqueter automatiquement de nouvelles données

→ prédictions intelligentes en généralisant à partir d'exemples

Détection de spam

Variables explicatives: texte de l'e-mail

From	Subject
Diana Coleman	Are you talkin' to me?
Martha Brooks	Step inside and taste the thrill of underworld Molls
@ Rosalind Mims	Important Docs
@ Dianna Underwood	Important Docs
Annu Chauhan	Best SEO Proposal !!
Sue Mockridge	-----SPAM----- Invoice 398499 April
Theresa Cox	This is the place you've been looking for
Jane Davis	This is the place you've been looking for
Michelle Jenkins	£88 On The House, No Catch, Keep Your Earnings
Diana Robinson	£88 On The House, No Catch, Keep Your Earnings
S.Rhodes	-----SPAM----- Acknowledge This Email !!! O
Helen Perez	Your name is on the winners list
Allison	-----SPAM----- Your 2 unread messages will be deleted in a few days

Etiquette (Label): Spam / No Spam

www.kaggle.com/uciml/sms-spam-collection-dataset

Apprentissage de la qualité d'un vin

Variables explicatives: attributs physico-chimiques



- 1) Alcohol
- 2) Malic acid
- 3) Ash
- 4) Alcalinity of ash
- 5) Magnesium
- 6) Total phenols
- 7) Flavanoids
- 8) Nonflavanoid phenols
- 9) Proanthocyanins
- 10) Color intensity
- 11) Hue
- 12) Proline
- 13) OD280/OD315 of diluted wines

Variable à prédire: note de 0 à 10

<https://archive.ics.uci.edu/ml/datasets/wine>

Marketing personnalisé / Aide à la décision

- ▶ Banque : prédiction de la “créditabilité”

Predictor (Categorical)	Levels and Proportions				
	No Account	None	Below 200 DM	200 DM or Above	
Account Balance (%)	27.40%	26.90%	6.30%	39.40%	
Payment Status	Delayed	Other Credits	Paid Up	No Problem with current credits	Previous Credits Paid
(%)	4.0%	4.9%	53.0%	8.8%	29.3%
Savings/Stock Value	None	Below 100 DM	[100, 500]	[500, 1000]	Above 1000
	60.3%	10.3%	6.3%	4.8%	18.3%
Length of Current Employment	Unemployed	< 1 Year	[1, 4]	[4, 7]	Above 7
	4.2%	17.2%	33.9%	17.4%	25.5%
Installment %	Above 35%	(25% - 35%)	[20% - 25%]	Below 20%	
	13.6%	23.1%	15.7%	47.6%	
Occupation	Unemployed, unskilled	Unskilled permanent resident	Skilled	Executive	
	2.2%	20%	63%	14.8%	
Sex and Marital Status	Male, Divorced	Male Single	Male, Married/Widowed	Female	
	5.0%	31.0%	54.8%	9.2%	
Duration in Current Address	< 1 Year	[1, 4]	[4, 7]	Above 7	
	13.0%	30.8%	14.9%	41.3%	
Type of Apartment	Free	Rented	Owned		
	17.9%	71.4%	10.7%		
Most Valuable Asset	None	Car	Life Insurance	Real Estate	
	28.2%	23.2%	33.3%	15.4%	
No. of Credits at Bank	1	2 or 3	4 or 5	Above 6	
	63.3%	33.3%	2.8%	0.06%	
Guarantor	None	Co-applicant	Guarantor		
	90.7%	4.1%	5.2%		
Concurrent Credits	Other Banks	Dept Stores	None		
	13.9%	4.7%	81.4%		
No of Dependents	3 or More	Less than 3			
	64.1%	15.8%			
Telephone	Yes	No			
	40.4%	59.6%			
Foreign Worker	Yes	No			
	3.7%	96.3%			

ex: German Credit Dataset

Marketing personnalisé / Aide à la décision

- ▶ Assurance : prédition de sinistres



Allstate Claim Prediction Challenge

A key part of insurance is charging each customer the appropriate price for the risk they represent.

\$10,000 · 102 teams · 6 years ago

[Overview](#) [Data](#) [Discussion](#) [Leaderboard](#) [Rules](#) [Team](#)



Allstate Claims Severity

How severe is an insurance claim?

3,055 teams · 10 months ago

[Overview](#) [Data](#) [Kernels](#) [Discussion](#) [Leaderboard](#) [Rules](#) [Team](#) [My Submissions](#) [Late Submission](#)

Kaggle

Marketing personnalisé / Aide à la décision

- ▶ Assurance : marketing

Allstate Purchase Prediction Challenge



Predict a purchased policy based on transaction history

\$50,000 · 1,568 teams · 3 years ago

[Overview](#) [Data](#) [Discussion](#) [Leaderboard](#) [Rules](#)

[Late Submission](#)



**BNP PARIBAS
CARDIF**

BNP Paribas Cardif Claims Management

Can you accelerate BNP Paribas Cardif's claims management process?

\$30,000 · 2,926 teams · a year ago

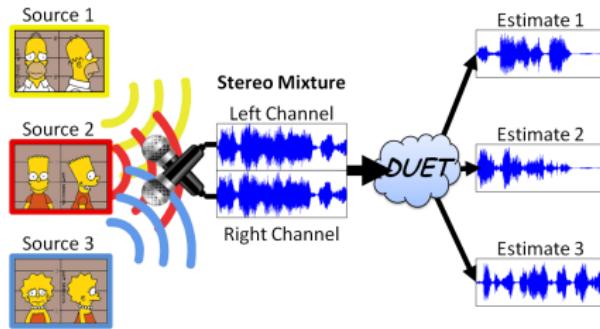
[Overview](#) [Data](#) [Kernels](#) [Discussion](#) [Leaderboard](#) [Rules](#)

[Late Submission](#)

Kaggle

Application au traitement du signal

- ▶ séparation de sources



- ▶ débruitage
- ▶ classification de musique
- ▶ reconnaissance vocale ...

Applications en vision artificielle

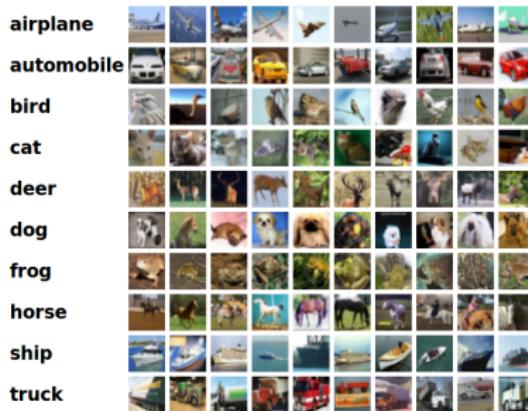
- ▶ Reconnaissance de caractères



MNIST dataset
[Démo]

Applications en vision artificielle

▶ Classification d'images



CIFAR dataset

- ▶ Segmentation et détection d'objets
- ▶ Annotation automatique d'images [Démo]
- ▶ ... et de vidéos ! [Démo]

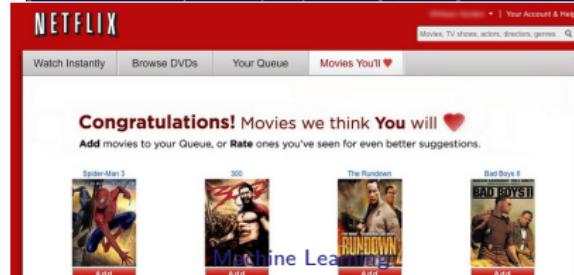
Apprentissage par renforcement (RL)

But: apprendre à l'aide d'**interactions** répétées avec son environnement (ex: **action** → **récompense**)

→ prise de décision automatique

Quelques applications

- ▶ robotique
- ▶ systèmes de dialogue
- ▶ gestion d'une grille énergétique
- ▶ AI pour les jeux
- ▶ systèmes de recommandations



RL et finance

Reinforcement Learning for Optimized Trade Execution

Yuriy Nevmyvaka

Lehman Brothers, 745 Seventh Av., New York, NY 10019, USA

Yi Feng

Michael Kearns[†]

University of Pennsylvania, Philadelphia, PA 19104, USA

yuriy.nevmyvaka@lehman.com

fengyi@cis.upenn.edu

mkearns@cis.upenn.edu

Abstract

We present the first large-scale empirical application of reinforcement learning to the important problem of optimized trade execution in modern financial markets. Our experiments are based on 1.5 years of millisecond time-scale limit order data from NASDAQ, and demonstrate the promise of reinforcement learning methods to market microstructure problems. Our learning algorithm introduces and exploits a natural "low-impact" factorization of the state space.

every strategy's profitability will depend on how well it is implemented.

For most of the history of the U.S. equity markets, the only information available to an agent attempting to optimize trade execution was the sequence of prices of already-executed trades, and the current bid and ask (the best outstanding buy and sell prices offered). But in recent years, electronic markets such as NASDAQ have begun releasing, in real time, all of the outstanding buy and sell limit order prices and volumes (often referred to as order book or market microstructure data). It is not difficult to see that such data might be extremely valuable for the problem of optimized execution, since the distribution of

RL et finance

DEEP PORTFOLIO MANAGEMENT

A Deep Reinforcement Learning Framework for the Financial Portfolio Management Problem

Zhengyao Jiang

ZHENGYAO.JIANG15@STUDENT.XJTLU.EDU.CN

Dixing Xu

DIXING.XU15@STUDENT.XJTLU.EDU.CN

Department of Computer Sciences and Software Engineering

Jinjun Liang

JINJUN.LIANG@XJTLU.EDU.CN

Department of Mathematical Sciences

Xi'an Jiaotong-Liverpool University

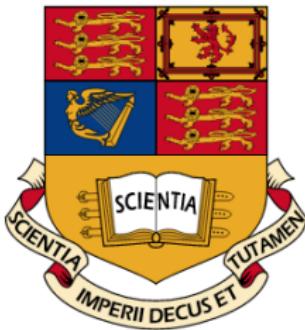
Suzhou, SU 215123, P. R. China

Editor: XZY ABCDE

Abstract

Financial portfolio management is the process of constant redistribution of a fund into different financial products. This paper presents a financial-model-free Reinforcement Learning framework to provide a deep machine learning solution to the portfolio management problem. The framework consists of the Ensemble of Identical Independent Evaluators (EIIE) topology, a Portfolio-Vector Memory (PVM), an Online Stochastic Batch Learning (OSBL) scheme, and a fully exploiting and explicit reward function. This framework is realized in three instants in this work with a Convolutional Neural Network (CNN), a basic Recurrent Neural Network (RNN), and a Long Short-Term Memory (LSTM). They are, along with a number of recently reviewed or published portfolio-selection strategies, examined in three back-test experiments with a trading period of 30 minutes in a cryptocurrency market. Cryptocurrencies are electronic and decentralised alternatives to government-issued money, with Bitcoin as the best known example of a cryptocurrency. All three instances of

RL et finance



IMPERIAL COLLEGE LONDON

DEPARTMENT OF COMPUTING

An Investigation into the Use of Reinforcement Learning Techniques within the Algorithmic Trading Domain

Outline

Théorie de l'apprentissage statistique

Algorithmes

Les k -plus proches voisins

Régression et régularisation

Arbres de décision

Les méthodes d'ensemble

Bagging et forêts aléatoires

Boosting: l'exemple des arbres de régression

Adaboost et Gradient Boosting

Retour sur l'évaluation d'un algorithme

Apprentissage séquentiel

Online Convex Optimization

Prediction de séquences individuelles

Prise de décision séquentielle

Modèles de bandit stochastiques

L'algorithme UCB

Outline

Théorie de l'apprentissage statistique

Algorithmes

Les k -plus proches voisins

Régression et régularisation

Arbres de décision

Les méthodes d'ensemble

Bagging et forêts aléatoires

Boosting: l'exemple des arbres de régression

Adaboost et Gradient Boosting

Retour sur l'évaluation d'un algorithme

Apprentissage séquentiel

Online Convex Optimization

Prediction de séquences individuelles

Prise de décision séquentielle

Modèles de bandit stochastiques

L'algorithme UCB

Cadre

On observe une base de données contenant des paires “variables explicatives / étiquette”

$$\mathcal{D}_n = \{(X_i, Y_i)\}_{i=1,\dots,n} \in \mathcal{X} \times \mathcal{Y}$$

Typiquement $\mathcal{X} = \mathbb{R}^d$ et

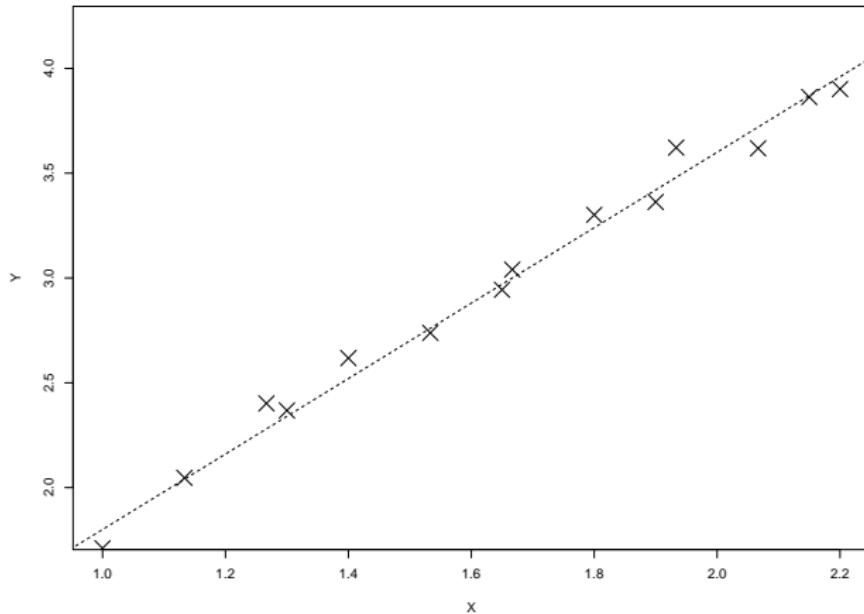
- ▶ $\mathcal{Y} = \{0, 1\}$: classification binaire
- ▶ $3 \leq |\mathcal{Y}| < \infty$: classification multi-classes
- ▶ $\mathcal{Y} = \mathbb{R}$: régression

On chercher à construire un **prédicteur** $\hat{g}_n : \mathcal{X} \rightarrow \mathcal{Y}$, càd une fonction dépendant des données telle que pour une nouvelle observation (\mathbf{X}, \mathbf{Y})

$$\hat{g}_n(\mathbf{X}) \simeq \mathbf{Y}.$$

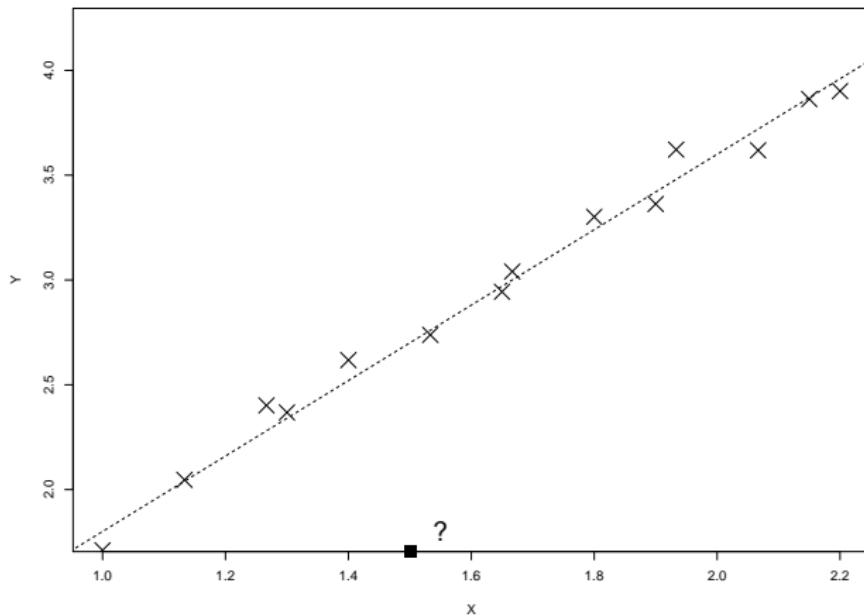
Illustration

Régression: $\mathcal{X} = \mathbb{R}$, $\mathcal{Y} = \mathbb{R}$



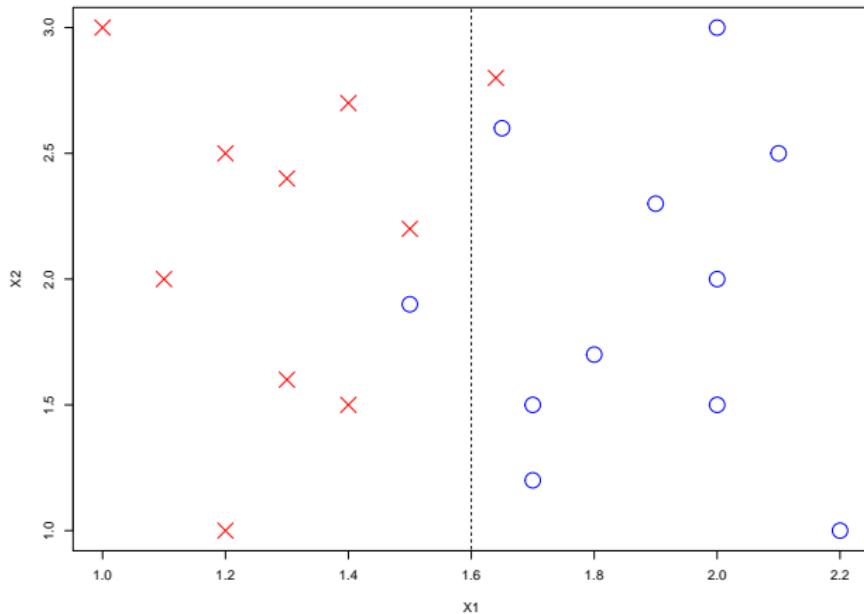
Illustration

Régression: $\mathcal{X} = \mathbb{R}$, $\mathcal{Y} = \mathbb{R}$



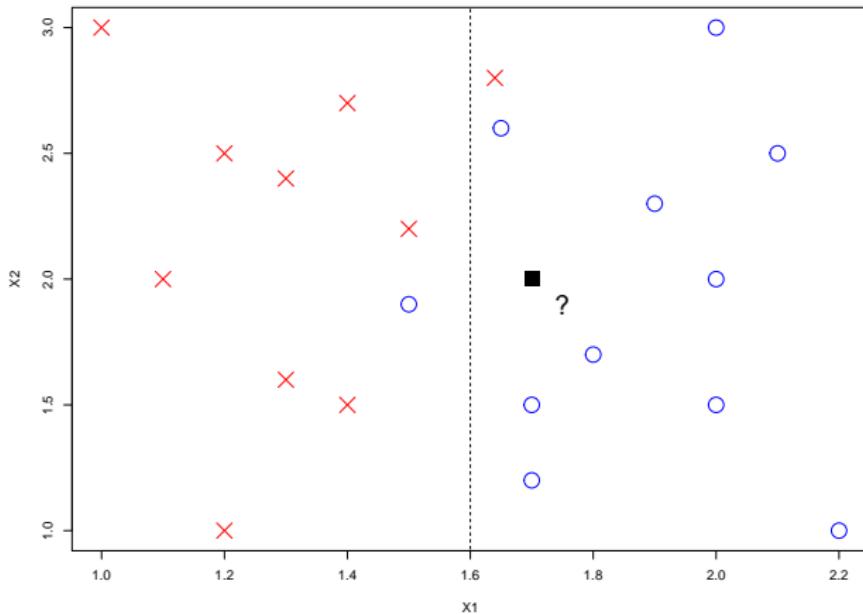
Illustration

Classification: $\mathcal{X} = \mathbb{R}^2$, $\mathcal{Y} = \{0, 1\}$



Illustration

Classification: $\mathcal{X} = \mathbb{R}^2$, $\mathcal{Y} = \{0, 1\}$



Formalisation probabiliste

La **base d'apprentissage** $\mathcal{D}_n = \{(X_i, Y_i)\}_{i=1,\dots,n}$ est formée de réplications i.i.d. d'une variable aléatoire

$$(\mathbf{X}, \mathbf{Y}) \sim \mathbb{P}.$$

où \mathbb{P} est une loi de probabilité sur $\mathcal{X} \times \mathcal{Y}$.

On cherche $\hat{g}_n = \hat{g}_n(\mathcal{D}_n)$ une fonction $\mathcal{X} \rightarrow \mathcal{Y}$ telle que si

$$(\mathbf{X}, \mathbf{Y}) \sim \mathbb{P} \quad \text{II} \quad \mathcal{D}_n$$

les variables aléatoires $\hat{g}_n(\mathbf{X})$ et \mathbf{Y} soient “proches”

→ Comment mesurer cette proximité?

Fonction de perte et risque

Définition

Etant donnée une **fonction de perte**

$$\ell : \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R}^+$$

on définit le **risque d'un prédicteur** $g : \mathcal{X} \rightarrow \mathcal{Y}$ par

$$R(g) = \mathbb{E} [\ell(g(\mathbf{X}), \mathbf{Y})] \in \mathbb{R}^+.$$

⚠ Pour un prédicteur \hat{g}_n construit à partir des données

$$R(\hat{g}_n) = \mathbb{E}_{(\mathbf{X}, \mathbf{Y}) \sim \mathbb{P}} [\ell(\hat{g}_n(\mathbf{X}), \mathbf{Y})]$$

est une variable aléatoire.

Exemples de fonctions de perte

- ▶ **Perte quadratique** (régression): $\ell(a, b) = (a - b)^2$

$$R(g) = \mathbb{E}[(g(\mathbf{X}) - \mathbf{Y})^2]$$

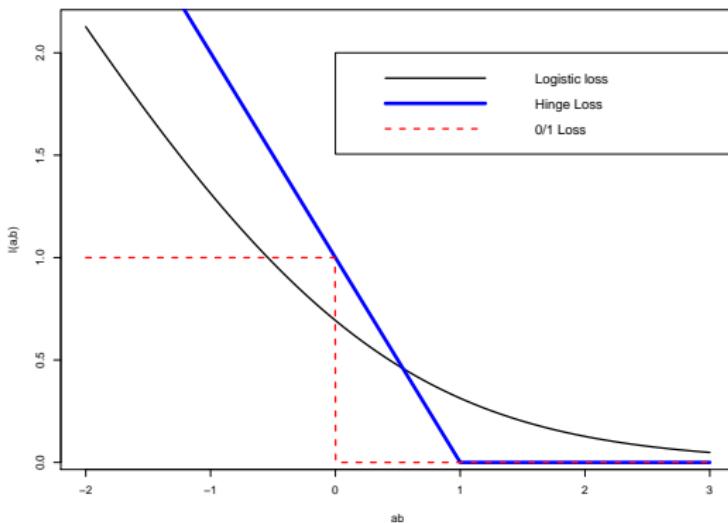
- ▶ **Perte 0-1** (classification): $\ell(a, b) = \mathbb{1}_{(a \neq b)}$

$$R(g) = \mathbb{P}(g(\mathbf{X}) \neq \mathbf{Y})$$

Alternatives

$\mathcal{Y} = \{-1, 1\}$ (ou $\mathcal{Y} = \mathbb{R}$)

- ▶ Perte 0-1 : $\ell(a, b) = \mathbb{1}_{(a \neq b)} = \mathbb{1}_{(ab = -1)} = \mathbb{1}_{(ab < 0)}$
- ▶ Perte logistique : $\ell(a, b) = \ln(1 + e^{-ab})$
- ▶ Hinge loss : $\ell(a, b) = \max(0, 1 - ab)$



Prédicteur de Bayes

Définition

Un **prédicteur de Bayes** est un prédicteur optimal, si les données sont tirées sous \mathbb{P} , au sens où

$$g^*(x) \in \operatorname{argmin}_{g \in \mathcal{Y}^{\mathcal{X}}} \mathbb{E} [\ell(g(\mathbf{X}), \mathbf{Y})]$$

(oracle théorique, car \mathbb{P} est généralement inconnue)

On appelle **risque de Bayes** le risque d'un prédicteur de Bayes:

$$R^* = R(g^*).$$

But: construire un prédicteur \hat{g}_n tel que $R(\hat{g}_n) \simeq R^*$
... sans connaître \mathbb{P}

Prédicteur de Bayes

Définition

Un **prédicteur de Bayes** est un prédicteur optimal, si les données sont tirées sous \mathbb{P} , au sens où

$$g^*(x) \in \operatorname{argmin}_{g \in \mathcal{Y}^{\mathcal{X}}} R(g)$$

(oracle théorique, car \mathbb{P} est généralement inconnue)

On appelle **risque de Bayes** le risque d'un prédicteur de Bayes:

$$R^* = R(g^*).$$

But: construire un prédicteur \hat{g}_n tel que $R(\hat{g}_n) \simeq R^*$
... sans connaître \mathbb{P}

Prédicteur de Bayes

Théorème

Si pour tout $x \in \mathcal{X}$ l'application $y \mapsto \mathbb{E}[\ell(y, \mathbf{Y}) | \mathbf{X} = x]$ possède un minimum, alors un prédicteur de Bayes est

$$g^*(x) \in \operatorname{argmin}_{y \in \mathcal{Y}} \mathbb{E}[\ell(y, \mathbf{Y}) | \mathbf{X} = x]$$

Prédicteur de Bayes

Théorème

Si pour tout $x \in \mathcal{X}$ l'application $y \mapsto \mathbb{E}[\ell(y, \mathbf{Y}) | \mathbf{X} = x]$ possède un minimum, alors un prédicteur de Bayes est

$$g^*(x) \in \operatorname{argmin}_{y \in \mathcal{Y}} \mathbb{E}[\ell(y, \mathbf{Y}) | \mathbf{X} = x]$$

- ▶ **Régression:** $\mathcal{Y} = \mathbb{R}$ et $\ell(a, b) = (a - b)^2$

$$g^*(x) = \eta^*(x) \quad \text{où} \quad \eta^*(x) = \mathbb{E}[\mathbf{Y} | \mathbf{X} = x]$$

(η^* est appelée **fonction de régression**)

Prédicteur de Bayes

Théorème

Si pour tout $x \in \mathcal{X}$ l'application $y \mapsto \mathbb{E}[\ell(y, \mathbf{Y}) | \mathbf{X} = x]$ possède un minimum, alors un prédicteur de Bayes est

$$g^*(x) \in \operatorname{argmin}_{y \in \mathcal{Y}} \mathbb{E}[\ell(y, \mathbf{Y}) | \mathbf{X} = x]$$

- ▶ **Classification:** $\mathcal{Y} = \{a_1, \dots, a_K\}$ et $\ell(a, b) = \mathbb{1}_{a \neq b}$

$$g^*(x) = \operatorname{argmax}_{y \in \mathcal{Y}} \mathbb{P}(\mathbf{Y} = y | \mathbf{X} = x)$$

Prédicteur de Bayes

Théorème

Si pour tout $x \in \mathcal{X}$ l'application $y \mapsto \mathbb{E}[\ell(y, \mathbf{Y}) | \mathbf{X} = x]$ possède un minimum, alors un prédicteur de Bayes est

$$g^*(x) \in \operatorname{argmin}_{y \in \mathcal{Y}} \mathbb{E}[\ell(y, \mathbf{Y}) | \mathbf{X} = x]$$

- ▶ **Classification binaire:** $\mathcal{Y} = \{0, 1\}$ et $\ell(a, b) = \mathbb{1}_{a \neq b}$

$$g^*(x) = \mathbb{1}_{(\eta^*(x) > 1/2)}$$

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

⚠ écriture un peu différente pour $\mathcal{Y} = \{-1, 1\}$

Lien classification/régression

$\mathcal{Y} = \{0, 1\} \subseteq \mathbb{R}$. On peut voir \mathcal{D}_n comme appartenant à

$$\underbrace{(\mathcal{X} \times \{0, 1\})^n}_{\text{classification}} \quad \text{ou à} \quad \underbrace{(\mathcal{X} \times \mathbb{R})^n}_{\text{régression}}$$

Idée du **plug-in**:

- ▶ si $\hat{\eta}_n$ est un bon régresseur
- ▶ ... alors $\hat{g}_n(x) = \mathbb{1}_{(\hat{\eta}_n(x) > 1/2)}$ est un bon classifieur

Théorème

$$R^{\text{class}}(\hat{g}_n) - R^{\text{class}}(g^*) \leq 2\sqrt{R^{\text{reg}}(\hat{\eta}_n) - R^{\text{reg}}(\eta^*)}.$$

Risque empirique

\mathbb{P} inconnu: on ne peut pas calculer le risque d'un prédicteur...

Définition

Le **risque empirique** du prédicteur g (calculé sur \mathcal{D}_n) est

$$\hat{R}_n(g) = \frac{1}{n} \sum_{i=1}^n \ell(g(X_i), Y_i)$$

→ pour la classification, on parle d'**erreur empirique**

Propriétés statistiques:

- ▶ $\mathbb{E}[\hat{R}_n(g)] = R(g)$
- ▶ $\hat{R}_n(g) \xrightarrow{P.S.} R(g)$ (Loi des Grands Nombres)
- ▶ $\sqrt{n} \left(\hat{R}_n(g) - R(g) \right) \xrightarrow{\mathcal{L}} \mathcal{N}(0, \text{Var}[\ell(g(\mathbf{X}), \mathbf{Y})])$
(Théorème Centrale Limite)

Risque empirique

\mathbb{P} inconnu: on ne peut pas calculer le risque d'un prédicteur...

Définition

Le **risque empirique** du prédicteur g (calculé sur \mathcal{D}_n) est

$$\hat{R}_n(g) = \frac{1}{n} \sum_{i=1}^n \mathbb{1}_{(g(X_i) \neq Y_i)}$$

→ pour la classification, on parle d'**erreur empirique**

Propriétés statistiques:

- ▶ $\mathbb{E}[\hat{R}_n(g)] = R(g)$
- ▶ $\hat{R}_n(g) \xrightarrow{P.S.} R(g)$ (Loi des Grands Nombres)
- ▶ $\sqrt{n} \left(\hat{R}_n(g) - R(g) \right) \xrightarrow{\mathcal{L}} \mathcal{N}(0, \text{Var}[\ell(g(\mathbf{X}), \mathbf{Y})])$
(Théorème Centrale Limite)

Minimisation du risque empirique

On voudrait approximer

$$g^* \in \operatorname{argmin}_{g \in \mathcal{Y}^{\mathcal{X}}} R(g)$$

- ▶ **Idée 1:** minimisation naïve du risque empirique

$$\hat{g}_n \in \operatorname{argmin}_{g \in \mathcal{Y}^{\mathcal{X}}} \hat{R}_n(g)$$

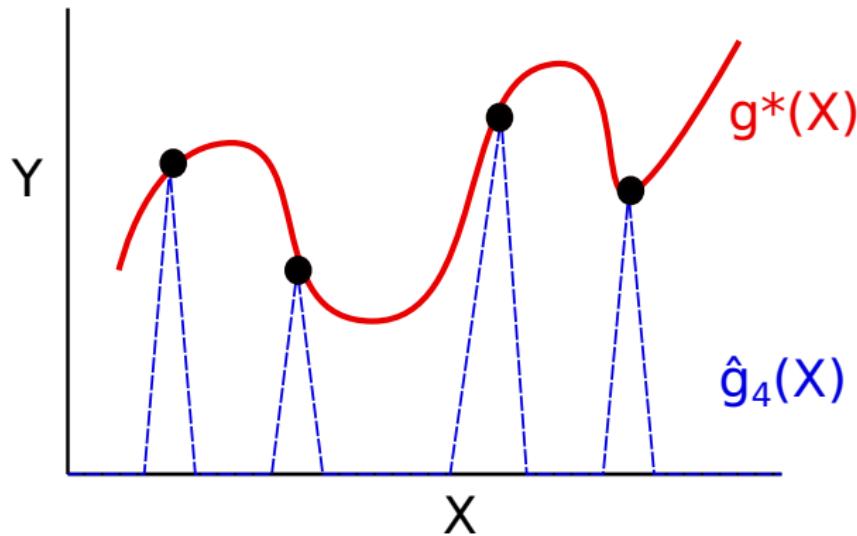
- Mauvaise idée !

Minimisation du risque empirique

Toute fonction vérifiant

$$\forall i \in \{1, \dots, n\}, \hat{g}_n(X_i) = Y_i$$

est telle que $\hat{R}_n(\hat{g}_n) = 0\dots$



Minimisation du risque empirique

- ▶ **Correction:** prédicteur minimisant le risque empirique

$$\hat{g}_n^{\text{ERM}} \in \operatorname*{argmin}_{g \in \mathcal{G}} \hat{R}_n(g)$$

sur une classe \mathcal{G} de fonctions ($\mathcal{G} \subset \mathcal{Y}^{\mathcal{X}}$)

- pas trop petite pour pouvoir contenir une bonne approximation du prédicteur de Bayes
- pas trop grande pour éviter le sur-apprentissage (coller trop aux données, mal généraliser)
- et pour laquelle on sait “résoudre” le problème d’optimisation

(**Empirical Risk Minimization**)

Minimisation du risque empirique

- ▶ **Correction:** prédicteur minimisant le risque empirique

$$\hat{g}_n^{\text{ERM}} \in \operatorname*{argmin}_{g \in \mathcal{G}} \hat{R}_n(g)$$

sur une classe \mathcal{G} de fonctions. ($\mathcal{G} \subset \mathcal{Y}^{\mathcal{X}}$)

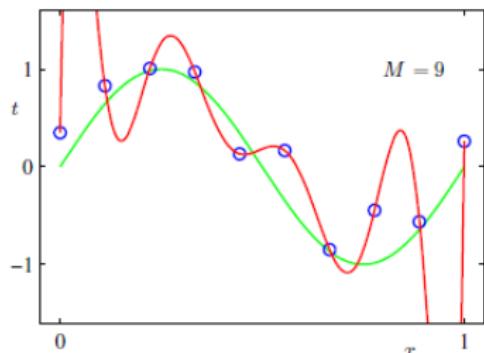
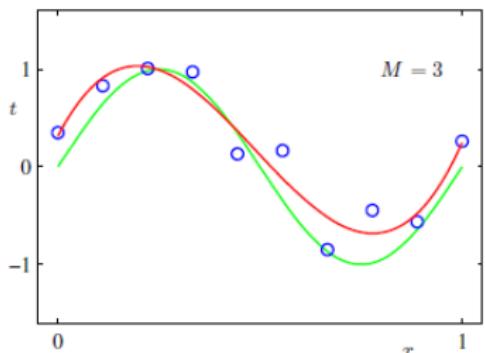
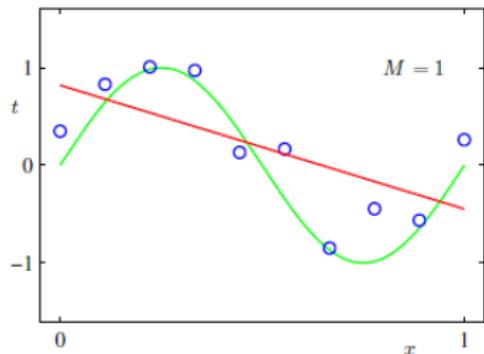
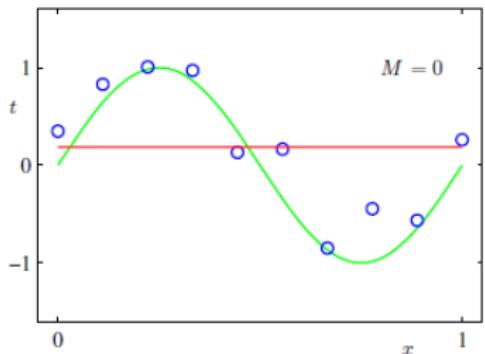
Exemples:

- $\mathcal{X} = \mathbb{R}^2, \mathcal{Y} = \{0, 1\}, \mathcal{G} = \{\mathbb{1}_{[a,b] \times [c,d]}, (a, b, c, d) \in \mathbb{R}^4\}$
(indicatrices de rectangles)
- $\mathcal{X} = \mathbb{R}, \mathcal{Y} = \mathbb{R}, \mathcal{G} = \{\text{polynômes de degré } \leq M\}$
- $\mathcal{X} = \mathbb{R}^d, \mathcal{Y} = \mathbb{R}, \mathcal{G} = \{x \mapsto a^T x + b, a \in \mathbb{R}^d, b \in \mathbb{R}\}$
(fonctions affines de \mathbb{R}^d)

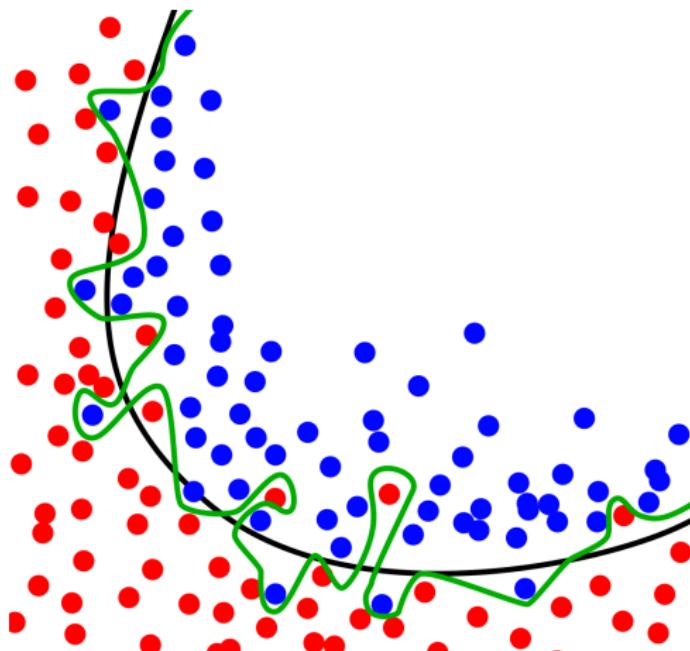
...

Sur-apprentissage

\mathcal{G} trop grand: on colle trop aux données



Sur-apprentissage



(exemple pour la classification)

Décomposition du risque

On introduit l'**excès de risque** d'un prédicteur

$$\mathcal{E}(g) := R(g) - R(g^*) \geq 0$$

Avec les notations

$$g^* \in \operatorname{argmin}_{g \in \mathcal{Y}^{\mathcal{X}}} R(g)$$

$$g_{\mathcal{G}}^* \in \operatorname{argmin}_{g \in \mathcal{G}} R(g)$$

on a

$$\mathcal{E}\left(\hat{g}_n^{\text{ERM}}\right) = \underbrace{R\left(\hat{g}_n^{\text{ERM}}\right) - R\left(g_{\mathcal{G}}^*\right)}_{\substack{\text{erreur d'estimation} \\ (\text{variance})}} + \underbrace{R\left(g_{\mathcal{G}}^*\right) - R\left(g^*\right)}_{\substack{\text{erreur d'approximation} \\ (\text{biais})}}$$

- pour un prédicteur de type ERM, la “taille” de \mathcal{G} contrôle le **compromis biais/variance**

Décomposition du risque

On introduit l'**excès de risque** d'un prédicteur

$$\mathcal{E}(g) := R(g) - R(g^*) \geq 0$$

Avec les notations

$$g^* \in \operatorname{argmin}_{g \in \mathcal{Y}^{\mathcal{X}}} R(g)$$

$$g_{\mathcal{G}}^* \in \operatorname{argmin}_{g \in \mathcal{G}} R(g)$$

on a

$$\mathcal{E}\left(\hat{g}_n^{\text{ERM}}\right) = \underbrace{R\left(\hat{g}_n^{\text{ERM}}\right) - R\left(g_{\mathcal{G}}^*\right)}_{\substack{\text{erreur d'estimation} \\ (\text{variance})}} + \underbrace{R\left(g_{\mathcal{G}}^*\right) - R\left(g^*\right)}_{\substack{\text{erreur d'approximation} \\ (\text{biais})}}$$

- pour un prédicteur de type ERM, la “taille” de \mathcal{G} contrôle le **compromis biais/variance**

Contrôle du terme stochastique

En fonction de la classe \mathcal{G} , comment contrôler l'erreur stochastique

$$R \left(\hat{g}_n^{\text{ERM}} \right) - R \left(g_{\mathcal{G}}^* \right) ?$$

Définition

Un prédicteur est **consistant** sur la classe \mathcal{G} si

$$R \left(\hat{g}_n^{\text{ERM}} \right) - R \left(g_{\mathcal{G}}^* \right) \xrightarrow[n \rightarrow \infty]{p.s.} 0$$

- on veut aller plus loin et proposer des garanties **non asymptotiques**, valables avec forte probabilité

Contrôle du terme stochastique

En fonction de la classe \mathcal{G} , comment contrôler l'erreur stochastique

$$R\left(\hat{g}_n^{\text{ERM}}\right) - R\left(g_{\mathcal{G}}^*\right) ?$$

Lemme crucial

$$R\left(\hat{g}_n^{\text{ERM}}\right) - R\left(g_{\mathcal{G}}^*\right) \leq 2 \sup_{g \in \mathcal{G}} \left| \hat{R}_n(g) - R(g) \right|$$

- Revient à contrôler uniformément la déviation d'une variable aléatoire par rapport à sa moyenne...

Cas où \mathcal{G} est fini

Hypothèses:

- ▶ \mathcal{G} est fini : $|\mathcal{G}| < \infty$
- ▶ fonction de perte bornée: $\forall (y, y') \in \mathcal{Y}^2, A < \ell(y, y') < B$

Theorème

Pour tout $\delta \in]0, 1[$, avec probabilité plus grande que $1 - \delta$,

$$R\left(\hat{g}_n^{\text{ERM}}\right) - R\left(g_{\mathcal{G}}^*\right) \leq (B - A) \sqrt{\frac{2 \ln(2|\mathcal{G}|\delta^{-1})}{n}}$$

- erreur décroissante avec la taille n de l'échantillon
- erreur croissante avec le cardinal de \mathcal{G}

Outils pour la preuve

- ▶ Inégalité de Hoeffding

Lemme (Hoeffding, 1955)

Soit V_1, \dots, V_n des variables aléatoires i.i.d. telles que $A \leq V_i \leq B$

$$\mathbb{P} \left(\frac{1}{n} \sum_{i=1}^n V_i - \mathbb{E}[V] > t \right) \leq \exp \left(- \frac{2nt^2}{(B-A)^2} \right)$$

- ▶ Borne de l'union

$$\mathbb{P} \left(\bigcup_{i \in \mathbb{N}} A_i \right) \leq \sum_{i \in \mathbb{N}} \mathbb{P}(A_i)$$

Cas où \mathcal{G} est fini

Hypothèses:

- ▶ \mathcal{G} est fini : $|\mathcal{G}| < \infty$
- ▶ fonction de perte bornée: $\forall (y, y') \in \mathcal{Y}^2, A < \ell(y, y') < B$

Theorème

Pour tout $\delta \in]0, 1[$, avec probabilité plus grande que $1 - \delta$,

$$R\left(\hat{g}_n^{\text{ERM}}\right) - R(g_{\mathcal{G}}^*) \leq (B - A) \sqrt{\frac{2[\ln |\mathcal{G}| + \ln 2 + \ln(1/\delta)]}{n}}$$

- erreur décroissante avec la taille n de l'échantillon
- erreur croissante avec $\ln |\mathcal{G}|$

Cas où \mathcal{G} est infini

Le borne précédente n'est plus informative...

Pour la **classification binaire**, Vapnik et Chervonenkis on proposé une nouvelle notion de complexité, appelée **VC-dimension**

- complexité liée au nombre de prédictions différentes que l'on peut obtenir sur \mathcal{D}_n à l'aide de fonctions de \mathcal{G}

Résultat fondamental de la théorie de l'apprentissage

Théorème

$\mathcal{Y} = \{0, 1\}$ et $\ell(y, y') = \mathbb{1}_{(y \neq y')}$ la perte 0 – 1. Pour tout $\delta \in]0, 1[$, avec probabilité plus grande que $1 - \delta$,

$$R(\hat{g}_n^{\text{ERM}}) - R(g_{\mathcal{G}}^*) \leq 4\sqrt{\frac{2\text{VC}(\mathcal{G}) \ln(n+1)}{n}} + \sqrt{\frac{3 \ln(2/\delta)}{n}}$$

Preuve: omise

Remarque: ce résultat utilise encore

$$R(\hat{g}_n^{\text{ERM}}) - R(g_{\mathcal{G}}^*) \leq 2 \sup_{\mathcal{G}} |\hat{R}_n(g) - R(g)|$$

et l'inégalité du théorème s'applique aussi au terme de droite.

VC dimension d'une classe de fonctions

$g : \mathcal{X} \rightarrow \{0, 1\}^n$ prend au plus 2^n valeurs...

$$\mathcal{G}((x_i)_{1 \leq i \leq n}) := \{b = (b_1, \dots, b_n) \in \{0, 1\}^n : \exists g \in \mathcal{G} : \forall i, b_i = g(x_i)\}$$

(ensemble des classifications possibles sur les données $(x_i)_{1 \leq i \leq n}$)

Définition

Le **coeffcient d'éclatement** de \mathcal{G} est

$$\mathcal{S}_{\mathcal{G}}(n) = \max_{(x_i)_{1 \leq i \leq n} \in \mathcal{X}^n} |\mathcal{G}((x_i)_{1 \leq i \leq n})|$$

Définition

La **dimension de Vapnik-Chervonenkis** de \mathcal{G} est

$$\text{VC}(\mathcal{G}) = \max\{V \in \mathbb{N} : \mathcal{S}_{\mathcal{G}}(V) = 2^V\}$$

VC dimension d'une classe de fonctions

$$(\text{VC}(\mathcal{G}) \geq k)$$

$$\Leftrightarrow \left(\exists (x_1, \dots, x_k) \in \mathcal{X}^k : \mathcal{G}(x_1, \dots, x_k) = \{0, 1\}^k \right)$$

“ \mathcal{G} éclate l'ensemble $\{x_1, \dots, x_k\}$ ”

$$(\text{VC}(\mathcal{G}) < k)$$

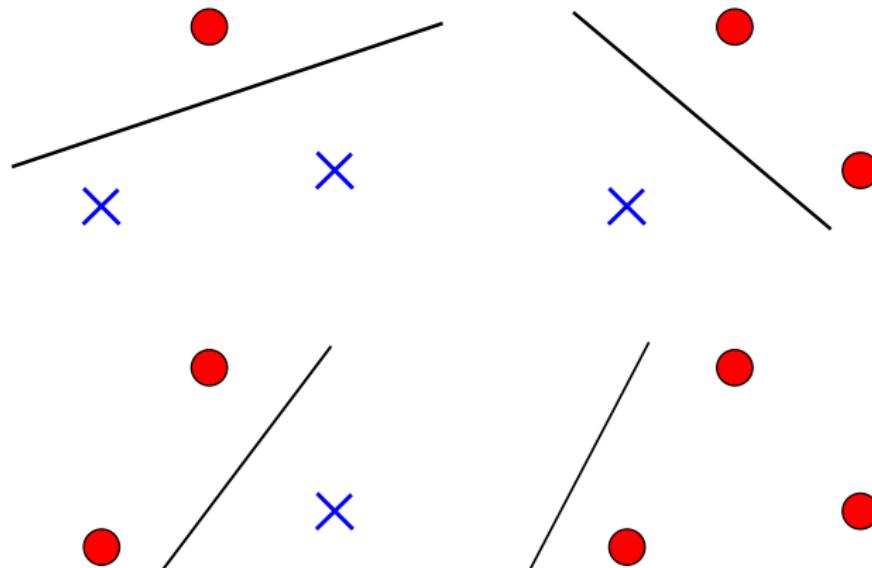
$$\Leftrightarrow \left(\forall (x_1, \dots, x_k) \in \mathcal{X}^k : \mathcal{G}(x_1, \dots, x_k) \neq \{0, 1\}^k \right)$$

“pour tous points (x_1, \dots, x_k) , il existe une configuration $(y_1, \dots, y_k) \in \{0, 1\}^k$ qui n'est prédite par aucune fonction de \mathcal{G} ”

Exemple

$\mathcal{X} = \mathbb{R}^2$ et $\mathcal{G} = \{\text{séparateurs linéaires}\}$

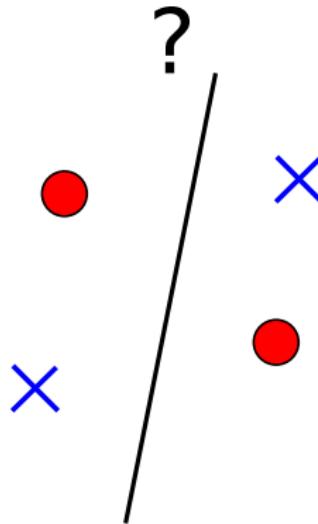
- ▶ $\text{VC}(\mathcal{G}) \geq 3$: donnons un exemple d'un ensemble de trois points éclaté par \mathcal{G}



Exemple

$\mathcal{X} = \mathbb{R}^2$ et $\mathcal{G} = \{\text{séparateurs linéaires}\}$

- ▶ $\text{VC}(\mathcal{G}) < 4$: pour tout ensemble de quatre points, \mathcal{G} ne peut pas fournir toutes les classifications possibles dans $\{0, 1\}^4$.



$\mathcal{X} = \mathbb{R}^d$, la VC-dimension des séparateurs linéaires est $d + 1$

Outline

Théorie de l'apprentissage statistique

Algorithmes

Les k -plus proches voisins

Régression et régularisation

Arbres de décision

Les méthodes d'ensemble

Bagging et forêts aléatoires

Boosting: l'exemple des arbres de régression

Adaboost et Gradient Boosting

Retour sur l'évaluation d'un algorithme

Apprentissage séquentiel

Online Convex Optimization

Prediction de séquences individuelles

Prise de décision séquentielle

Modèles de bandit stochastiques

L'algorithme UCB

La théorie suggère des prédicteurs

- ▶ **Minimisation du risque empirique**
- choisir une classe de fonction \mathcal{G} ...
- ...et une fonction de perte ℓ

La théorie suggère des prédicteurs

- ▶ **Minimisation du risque empirique**
- choisir une **classe de fonction \mathcal{G}** ...
- ...et une fonction de perte ℓ

- ▶ **Approche “plug-in”**

\mathbb{P} connue: calcul du prédicteur optimal $g^* = g^*(\mathbb{P})$.

\mathbb{P} inconnue : estimer g^* à partir de \mathcal{D}_n .

Exemple: soit $\hat{\eta}_n$ un **estimateur de la fonction de régression**

- $\hat{g}_n(x) = \hat{\eta}_n(x)$ pour la régression
- $\hat{g}_n(x) = \mathbb{1}_{(\hat{\eta}_n(x) > 1/2)}$ pour la classification binaire dans $\{0, 1\}$

En pratique: comment évaluer la qualité d'un prédicteur ?

Problème: on ne peut pas calculer

$$R(\hat{g}_n) = \mathbb{E}_{(\mathbf{X}, \mathbf{Y}) \sim \mathbb{P}} [\ell(\hat{g}_n(\mathbf{X}), \mathbf{Y})]$$

car la distribution \mathbb{P} est inconnue...

- ▶ **Idée 1:** l'estimer par le **risque (empirique)** d'apprentissage

$$\hat{R}_n(\hat{g}_n) = \frac{1}{n} \sum_{i=1}^n \ell(\hat{g}_n(X_i), Y_i)$$

En pratique: comment évaluer la qualité d'un prédicteur ?

Problème: on ne peut pas calculer

$$R(\hat{g}_n) = \mathbb{E}_{(\mathbf{X}, \mathbf{Y}) \sim \mathbb{P}} [\ell(\hat{g}_n(\mathbf{X}), \mathbf{Y})]$$

car la distribution \mathbb{P} est inconnue...

- ▶ **Idée 1:** l'estimer par le **risque (empirique)** d'apprentissage

$$\hat{R}_n(\hat{g}_n) = \frac{1}{n} \sum_{i=1}^n \ell(\hat{g}_n(X_i), Y_i)$$

NON !

- $\hat{g}_n = \hat{g}_n(\mathcal{D}_n)$, on s'attend donc à ce que $\hat{R}_n(\hat{g}_n)$ soit faible !
(peu d'erreurs sur la base d'apprentissage)
- ne traduit pas le **pouvoir de généralisation** du prédicteur

En pratique: comment évaluer la qualité d'un prédicteur ?

Problème: on ne peut pas calculer

$$R(\hat{g}_n) = \mathbb{E}_{(\mathbf{X}, \mathbf{Y}) \sim \mathbb{P}} [\ell(\hat{g}_n(\mathbf{X}), \mathbf{Y})]$$

car la distribution \mathbb{P} est inconnue...

► **Idée 2:** étant donnée une base de test

$$\mathcal{D}_{n_t}^{(t)} = (X_i'', Y_i'')_{1 \leq i \leq n_t}$$

supposée tirée sous \mathbb{P} indépendamment de \mathcal{D}_n , utiliser le risque (empirique) de test

$$\hat{R}_{n_t}^{\text{test}}(\hat{g}_n) = \frac{1}{n_t} \sum_{i=1}^{n_t} \ell(\hat{g}_n(X_i''), Y_i'')$$

En pratique: comment évaluer la qualité d'un prédicteur ?

Problème: on ne peut pas calculer

$$R(\hat{g}_n) = \mathbb{E}_{(\mathbf{X}, \mathbf{Y}) \sim \mathbb{P}} [\ell(\hat{g}_n(\mathbf{X}), \mathbf{Y})]$$

car la distribution \mathbb{P} est inconnue...

► **Idée 2:** étant donnée une **base de test**

$$\mathcal{D}_{n_t}^{(t)} = (X_i'', Y_i'')_{1 \leq i \leq n_t}$$

supposée tirée sous \mathbb{P} indépendamment de \mathcal{D}_n , utiliser le risque (empirique) de test

$$\hat{R}_{n_t}^{\text{test}}(\hat{g}_n) = \frac{1}{n_t} \sum_{i=1}^{n_t} \ell(\hat{g}_n(X_i''), Y_i'')$$

→ **estimateur sans biais:** $\mathbb{E} \left[\hat{R}_{n_t}^{\text{test}}(\hat{g}_n) \mid \mathcal{D}_n \right] = R(\hat{g}_n)$

Outline

Théorie de l'apprentissage statistique

Algorithmes

Les *k*-plus proches voisins

Régression et régularisation

Arbres de décision

Les méthodes d'ensemble

Bagging et forêts aléatoires

Boosting: l'exemple des arbres de régression

Adaboost et Gradient Boosting

Retour sur l'évaluation d'un algorithme

Apprentissage séquentiel

Online Convex Optimization

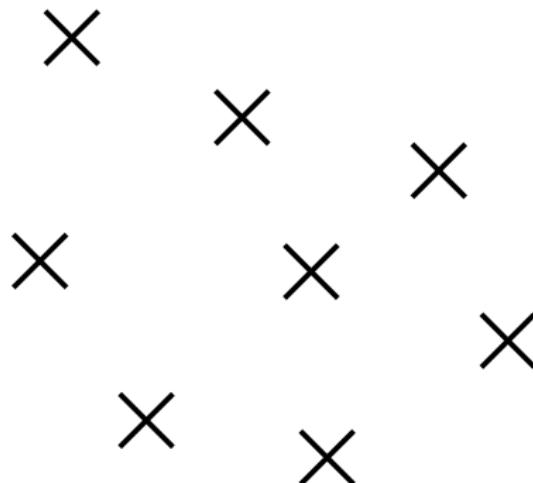
Prediction de séquences individuelles

Prise de décision séquentielle

Modèles de bandit stochastiques

L'algorithme UCB

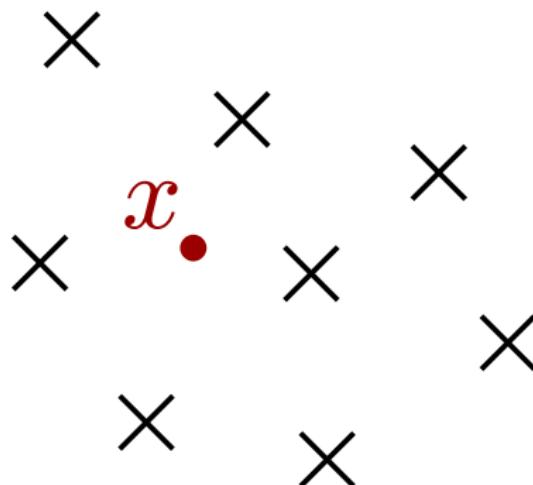
Une approche par “moyennage local”



Base d'apprentissage: $(X_i)_{1 \leq i \leq 8}$

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

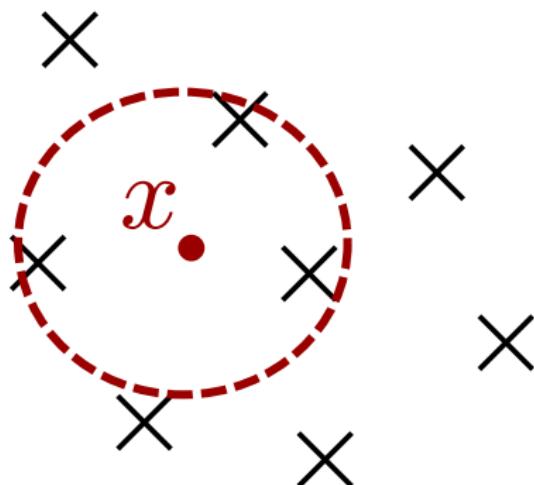
Une approche par “moyennage local”



Base d'apprentissage: $(X_i)_{1 \leq i \leq 8}$

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

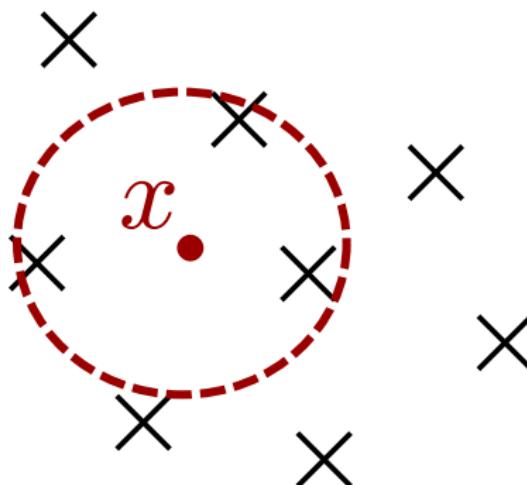
Une approche par “moyennage local”



Base d'apprentissage: $(X_i)_{1 \leq i \leq 8}$

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

Une approche par “moyennage local”



k plus proches voisins (k -nearest neighbours, k -nn):

Pour prédire l'étiquette de x , “moyenner” les étiquettes Y_i des points de la base d'apprentissage les plus proches de x

Définition formelle

Soit d une distance sur \mathcal{X} . Pour chaque x , on peut ordonner les points de la base d'apprentissage par leur distance à x :

$$d(x, X_{(1)}) \leq d(x, X_{(2)}) \leq \cdots \leq d(x, X_{(n)}).$$

Définition formelle

Soit d une distance sur \mathcal{X} . Pour chaque x , on peut ordonner les points de la base d'apprentissage par leur distance à x :

$$d(x, X_{\pi_1^{(n)}(x)}) \leq d(x, X_{\pi_2^{(n)}(x)}) \leq \cdots \leq d(x, X_{\pi_n^{(n)}(x)}).$$

$\pi_i^{(n)}(x)$: indice du i -ème point de \mathcal{D}_n le plus proche de x

Définition formelle

Soit d une distance sur \mathcal{X} . Pour chaque x , on peut ordonner les points de la base d'apprentissage par leur distance à x :

$$d(x, X_{\pi_1^{(n)}(x)}) \leq d(x, X_{\pi_2^{(n)}(x)}) \leq \cdots \leq d(x, X_{\pi_n^{(n)}(x)}).$$

$\pi_i^{(n)}(x)$: indice du i -ème point de \mathcal{D}_n le plus proche de x

Prédicteur des k -plus proches voisins

Pour $k \in \{1, \dots, n\}$, le prédicteur k -ppv est tel que

$$\hat{g}_n^{k-\text{ppv}}(x) = \text{"moyenne" des } \left\{ Y_{\pi_i^{(n)}(x)}, 1 \leq i \leq k \right\}$$

Définition formelle

► Régression:

$$\hat{g}_n^{k-\text{ppv}}(x) = \frac{1}{k} \sum_{i=1}^k Y_{\pi_i^{(n)}(x)}$$

(moyenne arithmétique)

Scikit-learn: fonction `KNeighborsRegressor`

► Classification:

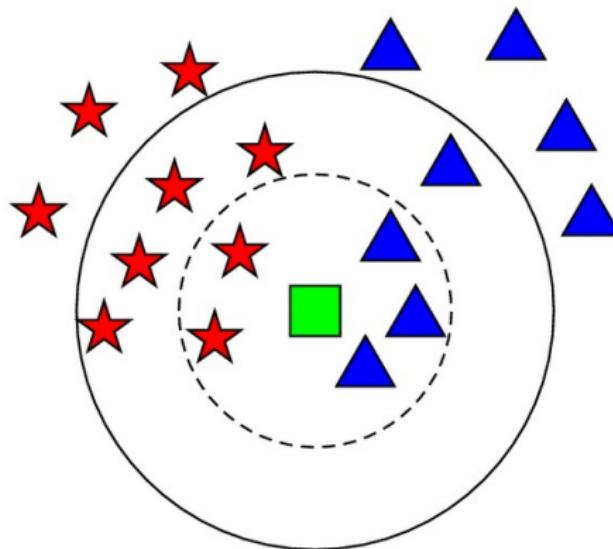
$$\hat{g}_n^{k-\text{ppv}}(x) = \operatorname{argmax}_{y \in \mathcal{Y}} \sum_{i=1}^k \mathbb{1}_{\left(Y_{\pi_i^{(n)}(x)} = y\right)}$$

(classe majoritaire)

Scikit-learn: fonction `KNeighborsClassifier`

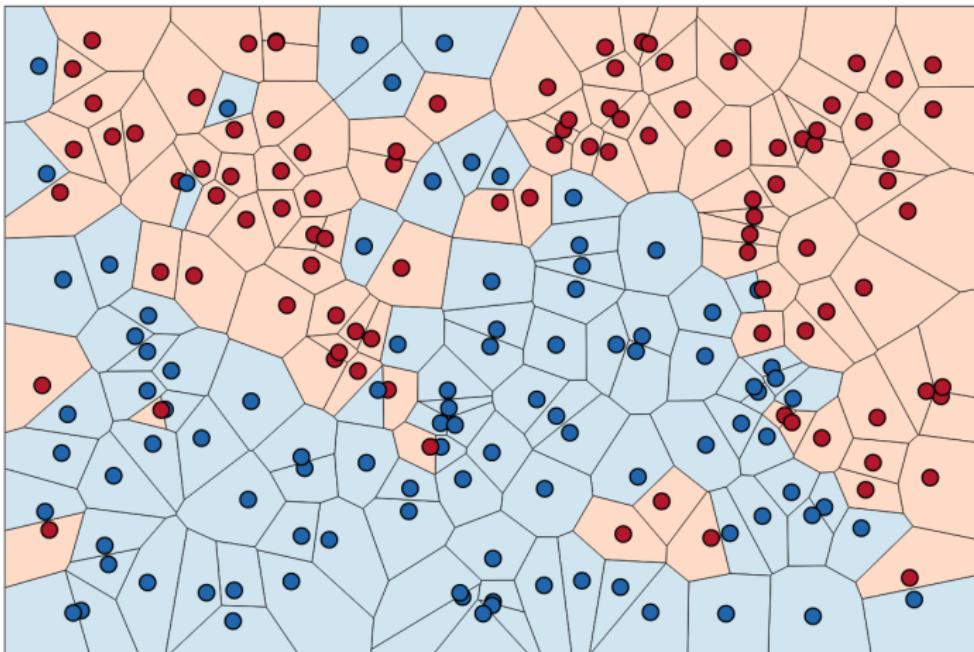
Visualisation (classification binaire)

$$d(x, y) = ||x - y||$$



- Le classifieur 5-ppv prédit \triangle
- Le classifieur 10-ppv prédit \star

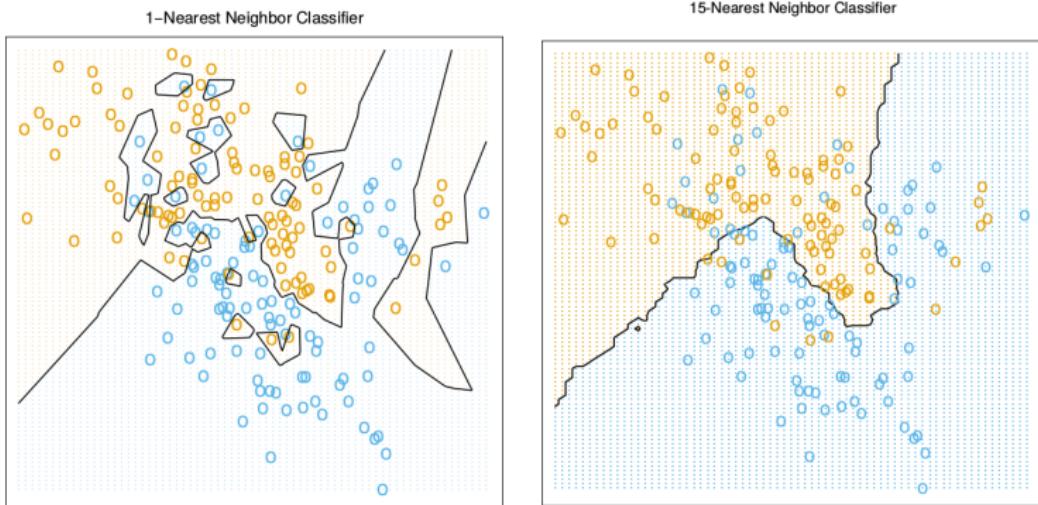
Visualisation (classification binaire)



Classifieur du plus-proche voisin

Source: [\[link\]](#) 60 / 173

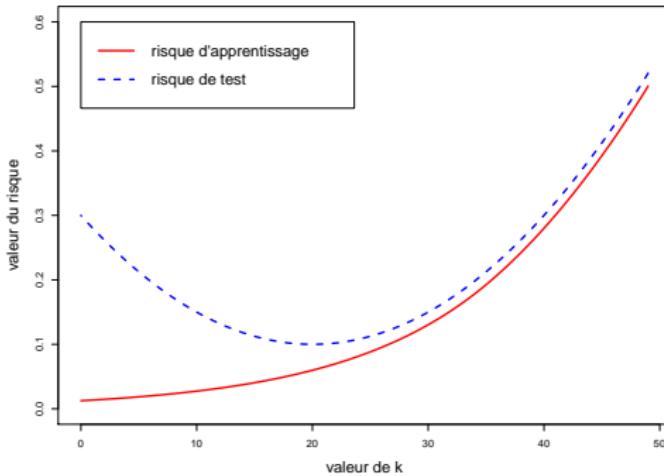
Visualisation (classification binaire)



Source: [Hastie et Tibshirani]

Comment choisir k ?

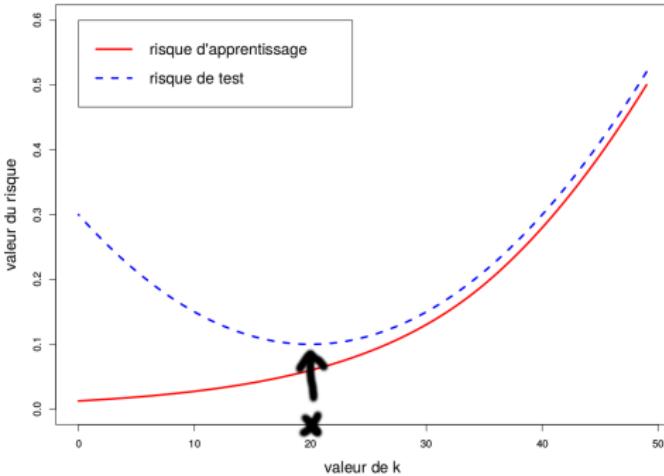
Evaluons le **risque d'apprentissage** et le **risque de test** pour différentes valeurs de k .



- petites valeurs de k : forte **variance** (sur-apprentissage)
- grandes valeurs de k : fort **biais**
(le classifieur appris est trop simple)

Comment choisir k ?

Peut-on sélectionner la valeur \hat{k} ci-dessous ?



$$\hat{k} = \hat{k}(\mathcal{D}_n, \mathcal{D}_m^{(t)})$$

ajusté pour minimiser le risque sur cette base de test particulière...

(⚠️ sur-apprentissage)

Comment choisir k ?

Une correction possible: **l'approche par validation**

- ▶ une **base d'apprentissage** \mathcal{D}_n pour construire le classifieur pour différentes valeurs de k : $\forall k, \hat{g}_n^{k-\text{ppv}} = \hat{g}_n^{k-\text{ppv}}(\mathcal{D}_n)$
- ▶ une **base de validation** $\mathcal{D}_m^{(v)}$ pour sélectionner le paramètre

$$\hat{k}_n \in \operatorname{argmin}_k \underbrace{\frac{1}{n_v} \sum_{i=1}^{n_v} \ell(\hat{g}_n^{k-\text{ppv}}(X'_i), Y'_i)}_{\text{risque de validation } \hat{R}_{nv}^{\text{val}}(\hat{g}_n^{k-\text{ppv}})} \quad \left(\hat{k}_n = \hat{k}_n(\mathcal{D}_n, \mathcal{D}_m^{(v)}) \right)$$

risque de validation $\hat{R}_{nv}^{\text{val}}(\hat{g}_n^{k-\text{ppv}})$: estimé sans biais de $R(\hat{g}_n^{k-\text{ppv}})$

- ▶ une **base de test** $\mathcal{D}_{nt}^{(t)}$ pour évaluer le classifieur obtenu:

$$\underbrace{\frac{1}{n_t} \sum_{i=1}^{n_t} \ell(\hat{g}_n^{\hat{k}_n-\text{ppv}}(X''_i), Y''_i)}_{\text{risque de test } \hat{R}_{nt}^{\text{test}}(\hat{g}_n^{\hat{k}_n-\text{ppv}})}$$

risque de test $\hat{R}_{nt}^{\text{test}}(\hat{g}_n^{\hat{k}_n-\text{ppv}})$: estimateur sans biais de $R(\hat{g}_n^{\hat{k}_n-\text{ppv}})$

Outline

Théorie de l'apprentissage statistique

Algorithmes

Les k -plus proches voisins

Régression et régularisation

Arbres de décision

Les méthodes d'ensemble

Bagging et forêts aléatoires

Boosting: l'exemple des arbres de régression

Adaboost et Gradient Boosting

Retour sur l'évaluation d'un algorithme

Apprentissage séquentiel

Online Convex Optimization

Prediction de séquences individuelles

Prise de décision séquentielle

Modèles de bandit stochastiques

L'algorithme UCB

Deux modèles génératifs

$\mathcal{X} = \mathbb{R}^d$. Deux modèles paramétriques de génération de données dépendant de $\theta \in \mathbb{R}^d$.

- ▶ **Régression linéaire:** $\mathcal{Y} = \mathbb{R}$

$$\mathbf{Y} = \langle \mathbf{X}, \theta \rangle + \epsilon, \quad \epsilon \sim \mathcal{N}(0, \sigma^2)$$

La fonction de régression est $\eta(x) = \langle x, \theta \rangle$ pour $x \in \mathbb{R}^d$.

- ▶ **Régression logistique:** $\mathcal{Y} = \{-1, 1\}$

$$\mathbb{P}(\mathbf{Y} = 1 | \mathbf{X} = x) = \frac{1}{1 + e^{-\langle x, \theta \rangle}}$$

Le classifieur de Bayes est $g^*(x) = \text{sgn}(\langle x, \theta \rangle)$.

Deux modèles génératifs

$\mathcal{X} = \mathbb{R}^d$. Deux modèles paramétriques de génération de données dépendant de $\theta \in \mathbb{R}^d$.

- ▶ **Régression linéaire:** $\mathcal{Y} = \mathbb{R}$

$$\mathbf{Y} = \langle \mathbf{X}, \theta \rangle + \epsilon, \quad \epsilon \sim \mathcal{N}(0, \sigma^2)$$

La fonction de régression est $\eta(x) = \langle x, \theta \rangle$ pour $x \in \mathbb{R}^d$.

- ▶ **Régression logistique:** $\mathcal{Y} = \{-1, 1\}$

$$\mathbb{P}(\mathbf{Y} = 1 | \mathbf{X} = x) = \frac{1}{1 + e^{-\langle x, \theta \rangle}}$$

Le classifieur de Bayes est $g^*(x) = \text{sgn}(\langle x, \theta \rangle)$.

- Estimer θ pour proposer des prédicteur “plug-in”

Maximum de vraisemblance

► Régression linéaire:

$$\hat{\theta}_n \in \operatorname{argmin}_{\theta \in \mathbb{R}^d} \sum_{i=1}^n (Y_i - \langle X_i | \theta \rangle)^2$$

(estimateur des moindres carrés
→ solution explicite)

Scikit-learn: fonction `LinearRegression` (`linear_model`)

► Régression logistique:

$$\hat{\theta}_n \in \operatorname{argmin}_{\theta \in \mathbb{R}^d} \sum_{i=1}^n \ln \left(1 + e^{-Y_i \langle X_i, \theta \rangle} \right)$$

(problème d'optimisation convexe
→ solution approchée)

Scikit-learn: fonction `LogisticRegression` (`linear_model`)

Minimisation du risque empirique

En posant $g_\theta(x) = \langle x | \theta \rangle$,

► **Régression linéaire:**

$$\hat{\theta}_n \in \operatorname{argmin}_{\theta \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^n (Y_i - g_\theta(X_i))^2$$

(estimateur des moindres carrés
→ solution explicite)

► **Régression logistique:**

$$\hat{\theta}_n \in \operatorname{argmin}_{\theta \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^n \ln \left(1 + e^{-Y_i g_\theta(X_i)} \right)$$

(problème d'optimisation convexe
→ solution approchée)

→ Deux problèmes de minimisation du risque empirique !

Minimisation du risque empirique

En notant \mathcal{G} l'ensemble des fonctions linéaires,

► **Régression linéaire:**

$$\hat{g}_n \in \operatorname{argmin}_{g \in \mathcal{G}} \frac{1}{n} \sum_{i=1}^n (Y_i - g(X_i))^2$$

(estimateur des moindres carrés
→ solution explicite)

► **Régression logistique:**

$$\hat{g}_n \in \operatorname{argmin}_{g \in \mathcal{G}} \frac{1}{n} \sum_{i=1}^n \ln \left(1 + e^{-Y_i g(X_i)} \right)$$

(problème d'optimisation convexe
→ solution approchée)

→ Deux problèmes de minimisation du risque empirique !

Généralisation de ces problèmes

On introduit un terme de pénalisation:

$$\hat{\theta}_n^\lambda \in \operatorname{argmin}_{\theta \in \mathbb{R}^d} \left(\underbrace{\sum_{i=1}^n (Y_i - \langle X_i | \theta \rangle)^2}_{\text{terme d'attache aux données}} + \lambda \underbrace{\sum_{k=1}^d \theta_k^2}_{\|\theta\|^2} \right)$$

λ est appelé paramètre de régularisation.

Généralisation de ces problèmes

On introduit un terme de pénalisation:

$$\hat{\theta}_n^\lambda \in \operatorname{argmin}_{\theta \in \mathbb{R}^d} \left(\underbrace{\sum_{i=1}^n (Y_i - \langle X_i | \theta \rangle)^2}_{\text{terme d'attache aux données}} + \lambda \underbrace{\sum_{k=1}^d \theta_k^2}_{\|\theta\|^2} \right)$$

λ est appelé paramètre de régularisation.

Interprétation: Il existe un réel $b > 0$ tel que si

$$\tilde{\mathcal{G}}_b = \{g : g(x) = \langle x | \theta \rangle, \|\theta\| \leq b\}$$

alors

$$\hat{\theta}_n^\lambda \in \operatorname{argmin}_{g \in \tilde{\mathcal{G}}_b} \frac{1}{n} \sum_{i=1}^n (Y_i - g(X_i))^2$$

→ ERM avec un ensemble de fonction $\tilde{\mathcal{G}}_b$ plus petit...

Généralisation de ces problèmes

Le problème peut se réécrire matriciellement:

$$\hat{\theta}_n^{\lambda-\text{ridge}} \in \underset{\theta \in \mathbb{R}^d}{\operatorname{argmin}} \left(\|Y - \mathbf{X}\theta\|^2 + \lambda \|\theta\|^2 \right)$$

→ Pénalisation par la **norme euclidienne**

$$\|\theta\|^2 = \sum_{k=1}^d \theta_k^2.$$

Solution exacte:

$$\hat{\theta}_n^{\lambda-\text{ridge}} = (\mathbf{X}^T \mathbf{X} + \lambda \text{Id})^{-1} \mathbf{X}^T Y$$

(regression ridge)

Grande dimension et parcimonie

$\mathcal{X} = \mathbb{R}^d$.

$$\hat{\theta}_n^{\lambda-\text{ridge}} = (\mathbf{X}^T \mathbf{X} + \lambda \text{Id})^{-1} \mathbf{X}^T Y$$

La matrice $\mathbf{X}^T \mathbf{X} + \lambda \text{Id}$ est de taille $d \times d$

→ inversion coûteuse si d est très grand

Hypothèse de parcimonie: Il existe $d_0 \ll d$ tel que le vecteur θ n'a que d_0 composantes non nulles

Si on connaît le **support de θ** (l'emplacement des composantes non nulles), le calcul de $\hat{\theta}_n^{\lambda-\text{ridge}}$ est moins coûteux et par exemple

$$\mathbb{E} \left[R \left(\hat{\theta}_n^0 \right) - R \left(\theta \right) \right] \leq C \times \frac{\sigma^2 d_0}{n}.$$

→ support inconnu...

Régularisation créant la parcimonie

Idée: Introduire une régularisation pour forcer la solution du problème à n'avoir que peu de composantes non nulles

- ▶ Régularisation par la norme ℓ_0

$$||\theta||_0 = \text{Card}\{\kappa : \theta_\kappa \neq 0\}$$

$$\hat{\theta}_n^\lambda \in \operatorname*{argmin}_{\theta \in \mathbb{R}^d} \left(||Y - X\theta||^2 + \lambda ||\theta||_0 \right)$$

- **Garanties théoriques** : pour le choix $\lambda \simeq \ln(d)/n$, on a

$$\mathbb{E} \left[R \left(\hat{\theta}_n^0 \right) - R \left(\theta \right) \right] \leq C \times \frac{\sigma^2 d_0 \ln(d)}{n}.$$

- **Implémentation** : très complexe pour d grand
(énumérer tous les supports possibles...)

L'estimateur LASSO

Idée: Introduire une régularisation pour forcer la solution du problème à n'avoir que peu de composantes non nulles

- ▶ Régularisation par la norme ℓ_1

$$\|\theta\|_1 = \sum_{k=1}^d |\theta_k|$$

$$\hat{\theta}_n^{\lambda-\text{LASSO}} \in \operatorname*{argmin}_{\theta \in \mathbb{R}^d} \left(\|Y - \mathbf{X}\theta\|^2 + \lambda \|\theta\|_1 \right)$$

(estimateur LASSO)

Least Absolute Shrinkage and Selection Operator

- favorise la parcimonie
- des algorithmes efficaces pour le calculer

L'estimateur LASSO

Idée: Introduire une régularisation pour forcer la solution du problème à n'avoir que peu de composantes non nulles

- ▶ Régularisation par la norme ℓ_1

$$||\theta||_1 = \sum_{k=1}^d |\theta_k|$$

$$\hat{\theta}_n^{\lambda-\text{LASSO}} \in \operatorname*{argmin}_{\theta \in \mathbb{R}^d} \left(||Y - X\theta||^2 + \lambda ||\theta||_1 \right)$$

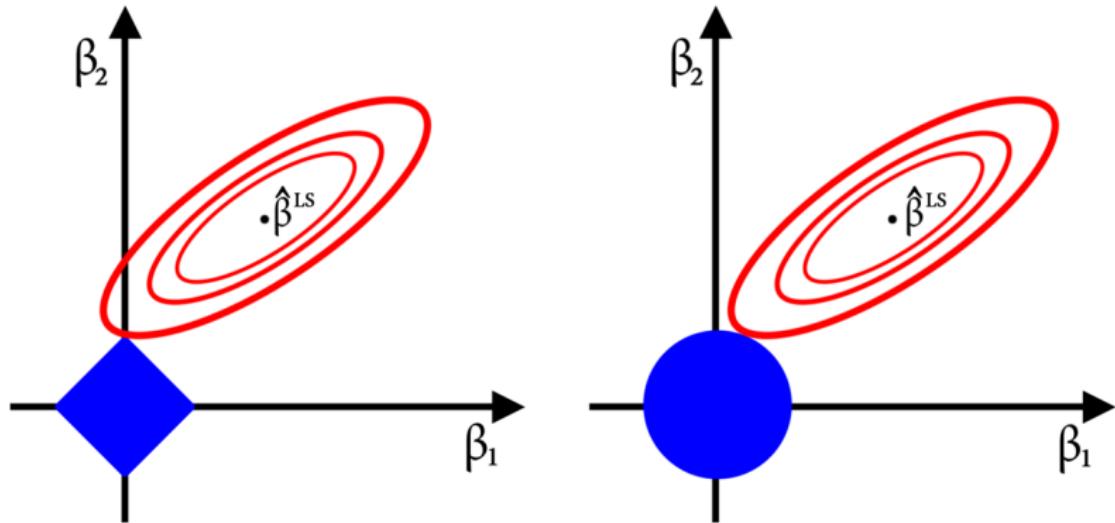
(estimateur LASSO)

Least Absolute Shrinkage and Selection Operator

- **Garanties théoriques:** pour $\lambda \simeq \ln(d)/n$,

$$\mathbb{E} \left[||\hat{\theta}_n^{\lambda-\text{LASSO}} - \theta|| \right] \leq Cd_0 \sqrt{\frac{\ln(d)}{n}}.$$

Régularisation ℓ_1 vs. régularisation ℓ_2



En pratique

- ▶ Principe plus général: “régulariser” le risque empirique permet d’ajouter des contraintes (de diminuer la taille de \mathcal{G})
- ▶ Scikit-learn: [RidgeClassifier](#), [Lasso](#)
- ▶ La régularisation ℓ_1 peut aussi s’appliquer pour la régression logistique ([LogisticRegression](#)), et au-delà
- ▶ Comment résoudre les problèmes d’optimisation associés ?
On peut utiliser des variations de la [descente de gradient](#)
(voir plus tard et dans le cours de Réseaux de Neurones)

Comment choisir le paramètre de régularisation ?

Le problème de **choix de paramètre d'algorithme** est un problème récurrent en apprentissage.

Une solution: **l'approche par validation**

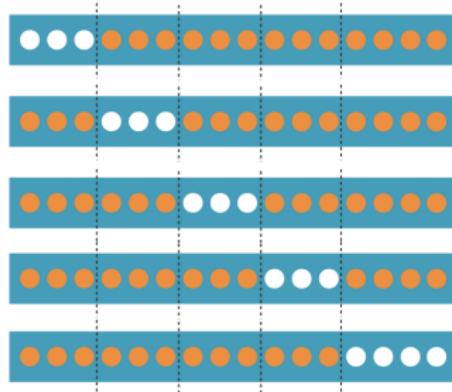
- retirer une partie de la base d'apprentissage (n_v observations) pour en faire une **base de validation** sur laquelle tester les prédicteurs $\hat{g}_{n-n_v}^\lambda$ pour différents λ

Inconvénient:

- on “sacrifie” des données pour la validation, alors que la qualité du prédicteur augmente avec le nombre de données utilisées pour l’entraînement

Comment choisir le paramètre de régularisation ?

Une alternative: **la validation croisée**



Choisir λ minimisant le **risque de validation croisée** (Cross-Validation)

$$\hat{R}_n^{CV}(\lambda) = \frac{1}{B} \sum_{b=1}^B \hat{R}_{n_v}^{\text{val}} \left(\hat{g}_{n_e}^\lambda \left(\overline{\mathcal{D}^{(b)}} \right) \right)$$

où $(\mathcal{D}^{(b)})_{b=1}^B$ partition de \mathcal{D}_n en B ensembles de taille $n_e = n - n_v$.

(entraîner sur $\overline{\mathcal{D}^{(b)}}$, valider sur $\mathcal{D}^{(b)}$)

- évite de “sacrifier” des données
- principe général pouvant s'utiliser dès lors qu'il s'agit de sélectionner le paramètre d'un algorithme d'apprentissage

Outline

Théorie de l'apprentissage statistique

Algorithmes

Les k -plus proches voisins

Régression et régularisation

Arbres de décision

Les méthodes d'ensemble

Bagging et forêts aléatoires

Boosting: l'exemple des arbres de régression

Adaboost et Gradient Boosting

Retour sur l'évaluation d'un algorithme

Apprentissage séquentiel

Online Convex Optimization

Prediction de séquences individuelles

Prise de décision séquentielle

Modèles de bandit stochastiques

L'algorithme UCB

Une classe de fonction particulière

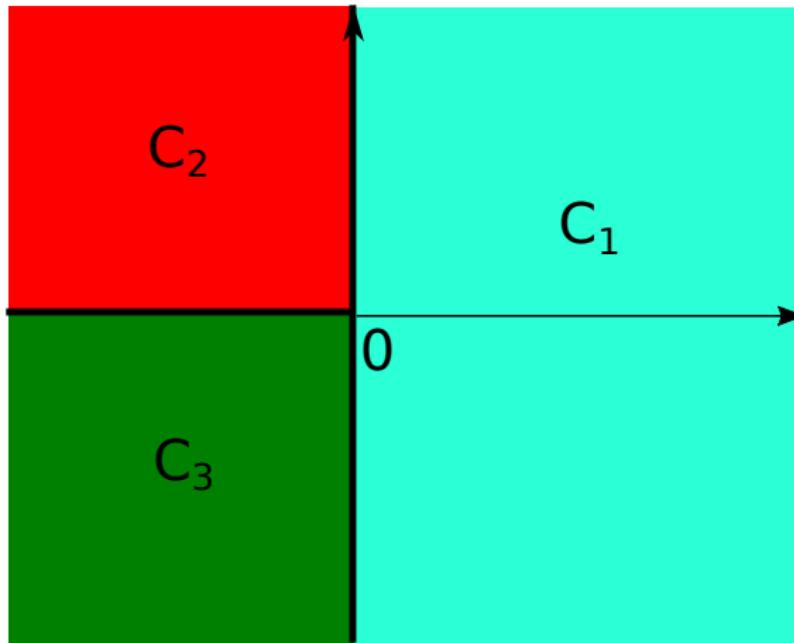
On souhaiterait calculer

$$\hat{g}_n \in \operatorname{argmin}_{g \in \mathcal{A}} \frac{1}{n} \sum_{i=1}^n \ell(g(X_i), Y_i),$$

où \mathcal{A} est l'ensemble des arbres de décision.

$$\mathcal{A} = \left\{ g : g(x) = \sum_{i=1}^K c_i \mathbb{1}_{\mathcal{C}_i}(x), K \in \mathbb{N}, \mathbb{R}^d = \underbrace{\bigcup_{i=1}^K \mathcal{C}_i}_{\begin{array}{l} \text{partition hiérarchique} \\ \text{de } \mathbb{R}^d \text{ par séparation successives} \\ \text{selon les coordonnées} \end{array}}, c = (c_i)_i \in \mathcal{Y}^K \right\}$$

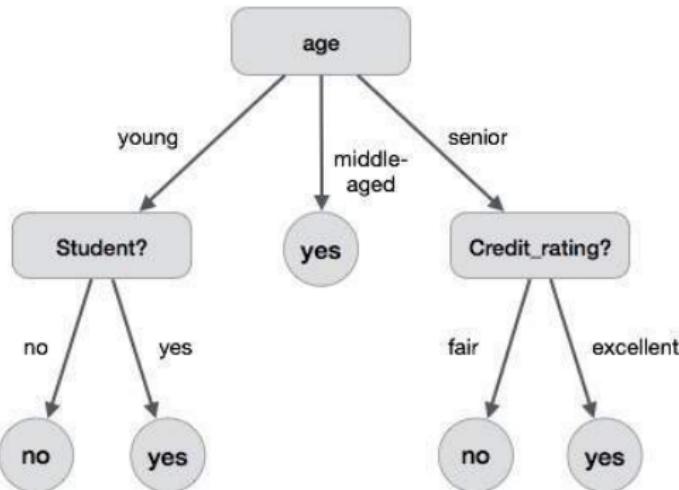
Exemple



$$\mathcal{X} = \mathbb{R}^2, \quad g(x) = \mathbb{1}_{(x^1 > 0)} + 2\mathbb{1}_{(x^1 < 0, x^2 \geq 0)} + 3\mathbb{1}_{(x^1 < 0, x^2 < 0)}$$

Exemple

$$\mathcal{X} = \mathbb{R}^3$$



X_i^1 = age du client i

X_i^2 = le client i est-il un étudiant?

X_i^3 = le client i a-t-il un bon historique de crédit?

Y_i = le client i va-t-il rembourser son crédit ?

Retour à la minimisation du risque empirique

On souhaiterait calculer

$$\hat{g}_n \in \operatorname{argmin}_{g \in \mathcal{A}} \frac{1}{n} \sum_{i=1}^n \ell(g(X_i), Y_i), \quad (1)$$

où \mathcal{A} est l'ensemble des arbres de décision.

La classe de fonctions \mathcal{A} est très complexe...

- ▶ impossible de résoudre (1)
- résolution approchée par la construction itérative d'un arbre ayant un faible risque empirique

L'algorithme CART

nœud \mathcal{N} : contient un sous-ensemble de $\mathcal{D}_n = \{(X_i, Y_i)\}$.

CART (\mathcal{N})

- ▶ Si \mathcal{N} est un nœud terminal , retourner $\{\mathcal{N}\}$
- ▶ Sinon,
 - pour chaque coordonnée $k = 1, \dots, d$,
électionner la meilleure coupe possible:

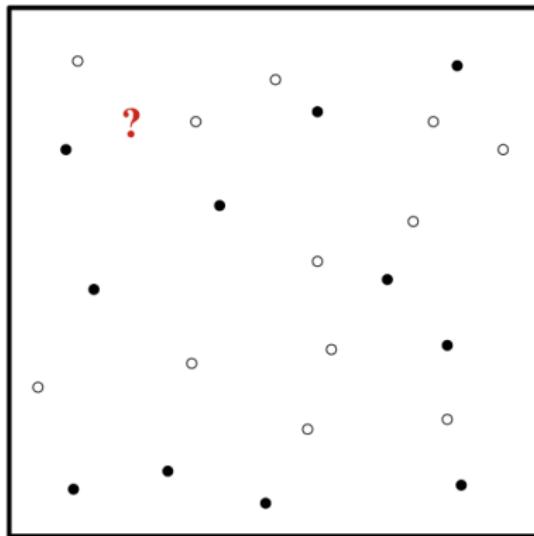
$$\mathcal{N}_1^k = \{i \in \mathcal{N} : X_i^{(k)} < c_k\} \quad \mathcal{N}_2^k = \{i \in \mathcal{N} : X_i^{(k)} \geq c_k\}$$

- Sélectionner la meilleure coordonée k^*
- Ajouter les noeuds $\mathcal{N}_1^{k^*}$ et $\mathcal{N}_2^{k^*}$ comme fils du nœud \mathcal{N}

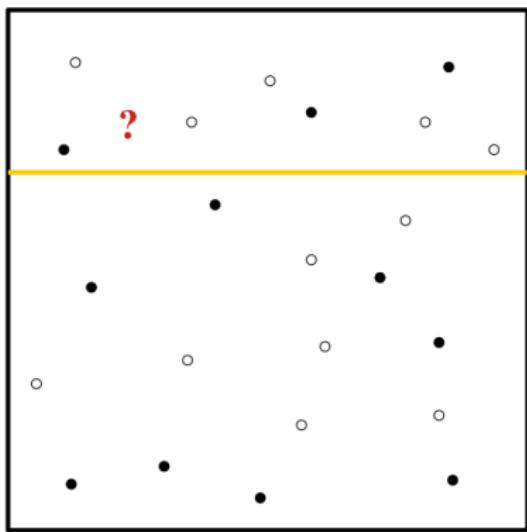
$$\mathcal{T} = \mathcal{T} \cup \{\text{CART}(\mathcal{N}_1^{k^*}), \text{CART}(\mathcal{N}_2^{k^*})\}$$

$\mathcal{T} = \{\mathcal{N}_1\}$ avec $\mathcal{N}_1 = \{1, \dots, n\}$
Appliquer $\text{CART}(\mathcal{N}_1)$.

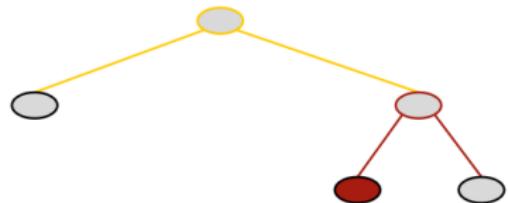
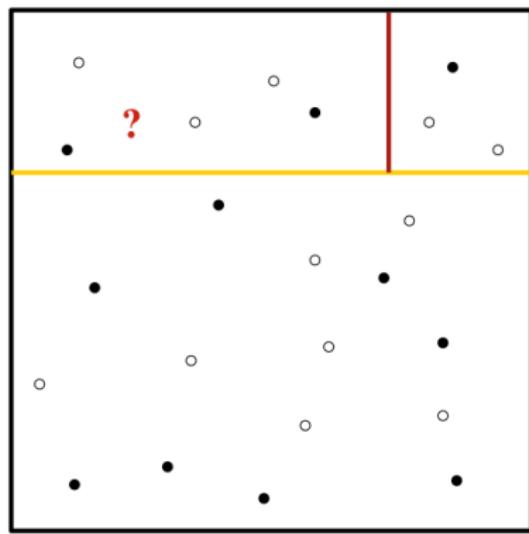
Visualisation



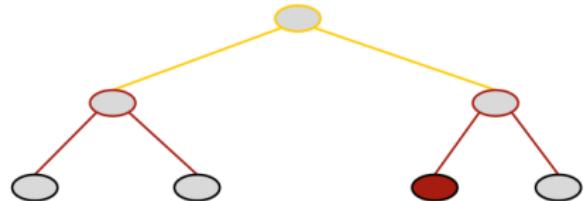
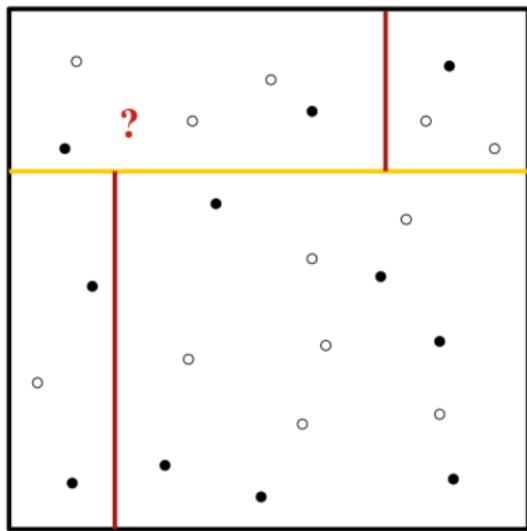
Visualisation



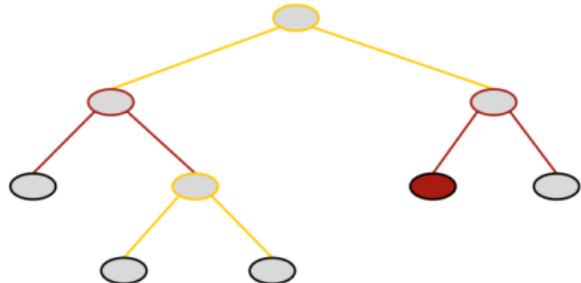
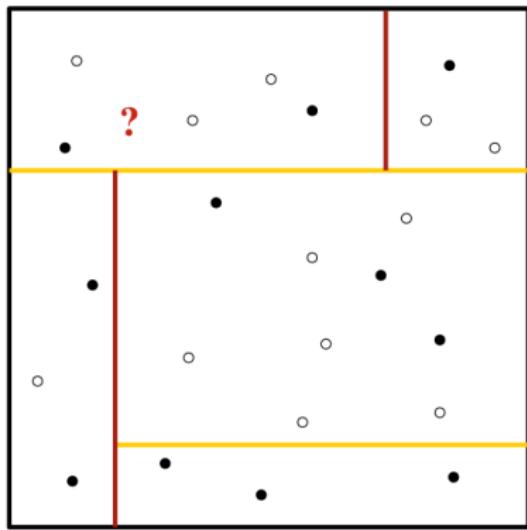
Visualisation



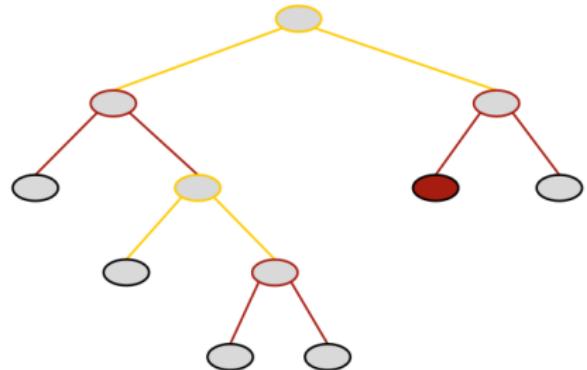
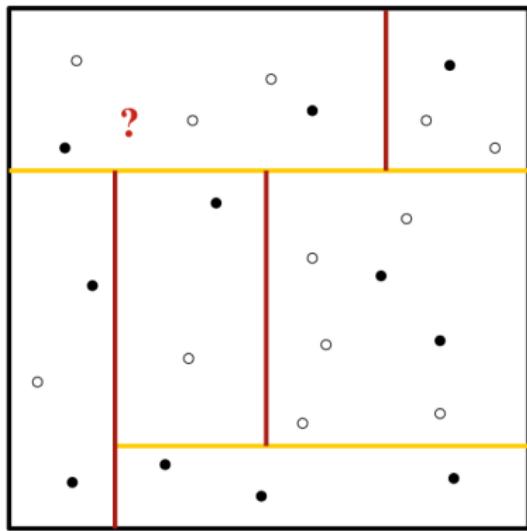
Visualisation



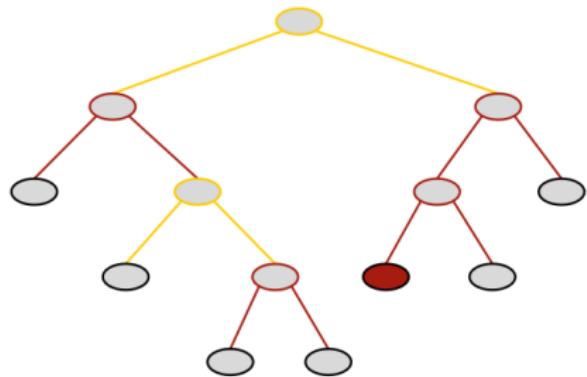
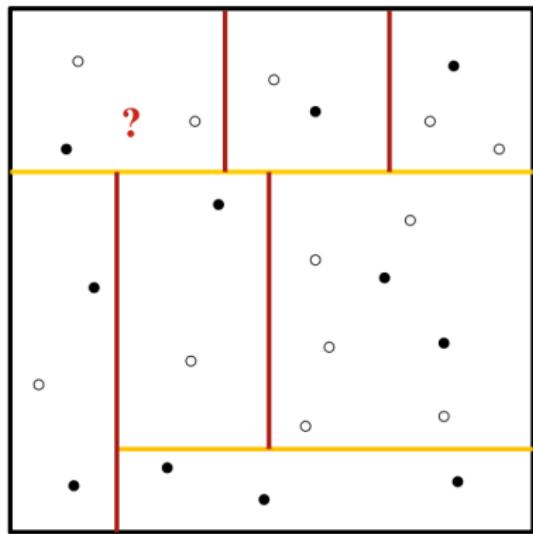
Visualisation



Visualisation



Visualisation



Paramètres de l'algorithme

- ▶ **Règle de prédiction:** Comment donner une valeur aux feuilles de l'arbre ?
 - Régression: moyenne des étiquettes au sein de la feuille
 - Classification: classe majoritaire au sein de la feuille
- ▶ **Règle d'arrêt:** faible nombre de données, arbre trop profond, plus de séparation intéressante (nœud homogène)...
- ▶ **Critère de séparation:** on choisit la séparation s minimisant un critère d'impureté Q parmi toutes les séparations possibles

$$Q\left(\mathcal{N}_s^{(1)}\right) + Q\left(\mathcal{N}_s^{(2)}\right) \quad \text{minimal}$$

Arbre de régression

Le critère d'impureté est la variance au sein d'un noeud:

$$Q(\mathcal{N}) = \min_{c \in \mathbb{R}} \frac{1}{|\mathcal{N}|} \sum_{i \in \mathcal{N}} (Y_i - c)^2$$

$$Q(\mathcal{N}) = \frac{1}{|\mathcal{N}|} \sum_{i \in \mathcal{N}} (Y_i - c_{\mathcal{N}})^2$$

où

$$c_{\mathcal{N}} = \frac{1}{|\mathcal{N}|} \sum_{i \in \mathcal{N}} Y_i.$$

Arbre de classification

Plusieurs critères d'impureté existent, basés sur le calcul de la fraction des données de \mathcal{N} possédant la classe k :

$$\hat{p}_{\mathcal{N}}(k) = \frac{1}{|\mathcal{N}|} \sum_{i \in \mathcal{N}} \mathbb{1}_{(Y_i=k)}.$$

- ▶ l'erreur de classification :

$$Q(\mathcal{N}) = 1 - \max_{k \in \mathcal{Y}} \hat{p}_{\mathcal{N}}(k)$$

- ▶ l'indice de Gini :

$$Q(\mathcal{N}) = \sum_{k \in \mathcal{Y}} \hat{p}_{\mathcal{N}}(k)(1 - \hat{p}_{\mathcal{N}}(k))$$

- ▶ la deviance (ou entropie)

$$Q(\mathcal{N}) = - \sum_{k \in \mathcal{Y}} \hat{p}_{\mathcal{N}}(k) \ln(\hat{p}_{\mathcal{N}}(k))$$

Quelle taille d'arbre choisir ?

arbre complet $\mathcal{T}^c(\mathcal{D})$: arbre construit par l'algorithme CART aux données \mathcal{D} avec pour critère d'arrêt “une seule donnée dans \mathcal{N} ”

Observation: l'arbre complet a une forte variance:

si $\mathcal{D} \amalg \mathcal{D}' \sim \mathbb{P}$, $\mathcal{T}^c(\mathcal{D})$ et $\mathcal{T}^c(\mathcal{D}')$ peuvent être très différents

Deux approches pour réduire la variance:

- **Idée 1:** limiter la profondeur ou le nombre de nœuds via une modification du critère d'arrêt
- **Idée 2:** construire l'arbre complet puis procéder à un élagage

Elagage

Elagage: recherche du sous-arbre \mathcal{T} de l'arbre complet \mathcal{T}^c minimisant le critère de coût/complexité suivant:

$$C_\alpha(\mathcal{T}) = \sum_{\mathcal{L} \in \text{Feuilles}(\mathcal{T})} |\mathcal{L}| Q(\mathcal{L}) + \alpha \times \text{Card}(\mathcal{T}),$$

$\text{Card}(\mathcal{T})$: nombre de nœuds de l'arbre \mathcal{T}

$|\mathcal{L}|$: nombre de données dans la feuille \mathcal{L}

$\alpha \in \mathbb{R}^+$: paramètre d'élagage

(la recherche s'effectue parmi une séquence \mathcal{S} de sous-arbres emboités de \mathcal{T}^c bien choisie)

- ▶ si α augmente, l'arbre obtenu

$$\mathcal{T}_\alpha^* \in \operatorname*{argmin}_{\mathcal{T} \subseteq \mathcal{S}} C_\alpha(\mathcal{T})$$

est de plus en plus petit.

En pratique

Scikit-learn: [DecisionTreeClassifier](#),
[DecisionTreeRegressor](#) (package `tree`)

- ▶ pour différentes valeurs du paramètre de profondeur `max_depth` et du nombre minimal de données engendrant l'arrêt `min_samples_leaf`, calculer [l'erreur de validation croisée](#)
 - ▶ conserver l'arbre correspondant à l'erreur minimale
 - peut s'effectuer directement à l'aide des fonctions [GridSearchCV](#) ou [RandomizedSearchCV](#) (package `model_selection`)
-
- + Méthodes facilement interprétable et visualisable (`graphviz`)
 - + Permet de prendre en compte plusieurs types de variables (on peut aller au delà de $\mathcal{X} = \mathbb{R}^d$)
 - Méthode pas toujours très robuste: petit changement dans les données → grand changement dans l'arbre

Outline

Théorie de l'apprentissage statistique

Algorithmes

Les k -plus proches voisins

Régression et régularisation

Arbres de décision

Les méthodes d'ensemble

Bagging et forêts aléatoires

Boosting: l'exemple des arbres de régression

Adaboost et Gradient Boosting

Retour sur l'évaluation d'un algorithme

Apprentissage séquentiel

Online Convex Optimization

Prediction de séquences individuelles

Prise de décision séquentielle

Modèles de bandit stochastiques

L'algorithme UCB

Observation: l'agrégation réduit la variance

Soit B estimateurs sans biais **indépendants** d'un paramètre θ :

$$\hat{\theta}^1, \dots, \hat{\theta}^B.$$

L'estimateur agrégé

$$\bar{\theta}^B = \frac{1}{B} \sum_{b=1}^B \hat{\theta}^b$$

est également sans biais mais avec une variance plus faible:

$$\begin{aligned}\mathbb{E}[\bar{\theta}^B] &= \theta \\ \text{Var}[\bar{\theta}^B] &\leq \frac{\max_{b=1,\dots,B} \text{Var}[\hat{\theta}^b]}{B}\end{aligned}$$

Appliquons cette idée au problème d'apprentissage

Soit B régresseurs construits à partir de B bases de données indépendantes:

$$\hat{g}_n^1(\mathcal{D}_n^1), \dots, \hat{g}_n^B(\mathcal{D}_n^B).$$

On définit le régresseur aggregé

$$\bar{g}^B = \frac{1}{B} \sum_{b=1}^B \hat{g}_n^B.$$

Propriété

$\ell(x, y) = (x - y)^2$. Le régresseur aggregé est tel que pour tout $x \in \mathcal{X}$,

$$\text{Var} [\bar{g}^B(x)] \leq \frac{1}{B} \times \max_{b=1, \dots, B} \text{Var} [\hat{g}_n^b(x)]$$

Appliquons cette idée au problème d'apprentissage

Soit B régresseurs construits à partir de B bases de données indépendantes:

$$\hat{g}_n^1(\mathcal{D}_n^1), \dots, \hat{g}_n^B(\mathcal{D}_n^B).$$

On définit le régresseur aggregé

$$\bar{g}^B = \frac{1}{B} \sum_{b=1}^B \hat{g}_n^B.$$

Propriété

$\ell(x, y) = (x - y)^2$. Le régresseur aggregé est tel que pour tout $x \in \mathcal{X}$,

$$\text{Var} [\bar{g}^B(x)] \leq \frac{1}{B} \times \max_{b=1, \dots, B} \text{Var} [\hat{g}_n^b(x)]$$

→ nécessite beaucoup trop de données !

Une solution: le ré-échantillonnage

Bootstrap: pour une procédure d'estimation $\hat{\theta}_n = \hat{\theta}(\mathcal{D}_n)$, calculer l'estimateur à partir de plusieurs ré-échantillonnages des données.

$$\forall b = 1, \dots, B, \quad \hat{\theta}_n^{(b)} = \hat{\theta}(\mathcal{D}_n^{(b)}),$$

où $\mathcal{D}_n^{(b)}$ est une nouvelle base de données contenant n observations tirées uniformément avec remise parmi les n observations de \mathcal{D}_n .

($\mathcal{D}_n^{(b)}$ est appelé échantillon bootstrap)

- ▶ en pratique, tirer $I_1, \dots, I_n \stackrel{i.i.d.}{\sim} \mathcal{U}(\{1, \dots, n\})$ et définir $\mathcal{D}_n^{(b)} = \{(X_{I_j}, Y_{I_j})\}_{1 \leq j \leq n}$
- ▶ revient à tirer $\mathcal{D}_n^{(b)}$ sous la loi empirique des observations \hat{F}_n

Une solution: le ré-échantillonnage

Bootstrap: pour une procédure d'estimation $\hat{\theta}_n = \hat{\theta}(\mathcal{D}_n)$, calculer l'estimateur à partir de plusieurs ré-échantillonnages des données.

$$\forall b = 1, \dots, B, \quad \hat{\theta}_n^{(b)} = \hat{\theta}(\mathcal{D}_n^{(b)}),$$

où $\mathcal{D}_n^{(b)}$ est une nouvelle base de données contenant n observations tirées uniformément avec remise parmi les n observations de \mathcal{D}_n .

($\mathcal{D}_n^{(b)}$ est appelé échantillon bootstrap)

- estimateur bootstrap: approximation de $\mathbb{E}_{\mathcal{D} \sim \hat{F}_n} [\hat{\theta}(\mathcal{D})]$ par

$$\bar{\theta}^B = \frac{1}{B} \sum_{b=1}^B \hat{\theta}^{(b)}$$

(les échantillons bootstrap permettent aussi l'estimation de la distribution de $\hat{\theta}(\mathcal{D}) \rightarrow$ intervalles de confiance)

Le Bagging

Bagging = Bootstrap Aggregating

Idée: combiner un ensemble de B prédicteurs entraînés sur des échantillons bootstrap de \mathcal{D}_n

$$\forall b = 1, \dots, B, \quad \hat{g}_n^{(b)} = \hat{g}_n(\mathcal{D}^{(b)})$$

- ▶ **Régression:** moyenne des prédicteurs

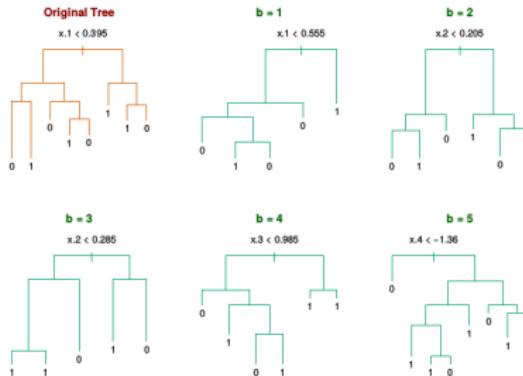
$$\bar{g}_n^B(x) = \frac{1}{B} \sum_{b=1}^B \hat{g}_n^{(b)}(x)$$

- ▶ **Classification:** vote majoritaire

$$\bar{g}_n^B(x) = \operatorname{argmax}_{k \in \mathcal{Y}} \sum_{b=1}^B \mathbb{1}_{(\hat{g}_n^{(b)}(x)=k)}$$

Le Bagging pour les Arbres de Décision

Plutôt que d'élager un arbre complet, on agrège plusieurs arbres complets construits à partir d'échantillons bootstrap



Agrégation d'arbres = forêt (arbres très variés...)

Propriétés:

- ▶ L'erreur Out-Of-Bag fournit une estimation du risque
- ▶ Pas de sur-apprentissage lorsque B augmente
- ▶ Potentielle réduction de variance (corrélations...)

Random Forest

$$\mathcal{X} = \mathbb{R}^d$$

Idée: agrégation d'arbres décorrelés.

RANDOM FOREST [Leo Breiman, 2001]

Paramètres: B (nombre d'arbres)

$m < d$ (nombre de variables sélectionnées)

Pour $b = 1, \dots, B$:

- ▶ Tirer un échantillon bootstrap des données $\mathcal{D}_n^{(b)}$
- ▶ Construire un arbre *complet* $T^{(b)}$ à partir de $\mathcal{D}_n^{(b)}$; en partant d'un arbre réduit à un nœud contenant toutes les données, et pour un nœud \mathcal{N} non terminal ($|\mathcal{N}| > n_{\min}$):
 - ▶ sélectionner au hasard m variables parmi d
 - ▶ choisir la meilleure variable/coupe parmi ces m variables
 - ▶ séparer le nœud en deux nœuds fils

Retourner: l'ensemble d'arbres $\{T_b, b = 1, \dots, B\}$

Random Forest: recommandations pratiques

- ▶ Si $m = d$ (pas de sélection de variables), Random Forest coincide avec le Bagging
- ▶ Comme pour le bagging, on peut prendre B assez grand, typiquement $B = 500$
- ▶ Le choix de m par défaut est
 - $m = \sqrt{d}$ pour la classification
 - $m = d/3$ pour la régression

(on peut aussi chercher le meilleur m par validation croisée)
- ▶ Scikit-learn: `RandomForestClassifier` (package `ensemble`)

Outline

Théorie de l'apprentissage statistique

Algorithmes

Les k -plus proches voisins

Régression et régularisation

Arbres de décision

Les méthodes d'ensemble

Bagging et forêts aléatoires

Boosting: l'exemple des arbres de régression

Adaboost et Gradient Boosting

Retour sur l'évaluation d'un algorithme

Apprentissage séquentiel

Online Convex Optimization

Prediction de séquences individuelles

Prise de décision séquentielle

Modèles de bandit stochastiques

L'algorithme UCB

L'idée du Boosting

Bagging: agrégation d'arbres qui pourraient être construits en parallèle (de manière aussi décorrelée que possible pour R.F.)

v.s.

Boosting: agrégation **séquentielle** d'arbres

- on ajoute les arbres un après l'autre pour booster les performances du prédicteur précédent
(améliorer progressivement ses performances)
- pas d'utilisation de ré-échantillonage
- l'idée est de booster des **prédicteurs faibles**
(i.e. des arbres de profondeurs faibles)

Boosting d'arbres de régression

BOOSTED REGRESSION TREES

Paramètres: B (nombre d'arbres)

p (nombre de noeuds des arbres)

$\lambda \in \mathbb{R}^+$ (contrôle l'importance donnée aux nouveaux arbres)

Initialisation: $\hat{g}(x) = 0$, $R_i = Y_i$ pour $i \in \{1, \dots, n\}$

Pour $b = 1, \dots, B$:

- ▶ Entraîner un arbre \hat{g}^b ayant p nœuds à partir de la base de données $\mathcal{D}' = \{(X_i, R_i)\}_{1 \leq i \leq n}$
- ▶ Mettre à jour \hat{g} en prenant en compte le nouvel arbre

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

- ▶ Mettre à jour les résidus, i.e. les erreurs faites par \hat{g} :

$$R_i \leftarrow R_i - \lambda \hat{g}^b(X_i)$$

Retourner: le régresseur \hat{g}

Considérations pratiques

- ▶ Le paramètre p qui contrôle la taille des arbres (nombre de séparations) est appelé **paramètre d'interactions** (au plus p variables mises en jeu)
- ▶ on peut avoir de bonnes performances avec $p = 1$ (stumps), et généralement on prend p pas trop grand
- ▶ λ contrôle la **vitesse d'apprentissage**: plus λ est faible, plus il faudra choisir B grand
- ▶ contrairement au Bagging, prendre **B trop grand** peut conduire à du sur-apprentissage
- ▶ il faut donc choisir B (et λ) en utilisant une **approche de validation / une validation croisée**

L'algorithme précédent est un cas particulier d'une approche appelée GRADIENT BOOSTING, qui peut s'appliquer en régression et en classification, et qui généralise l'algorithme ADABOOST.

Outline

Théorie de l'apprentissage statistique

Algorithmes

Les k -plus proches voisins

Régression et régularisation

Arbres de décision

Les méthodes d'ensemble

Bagging et forêts aléatoires

Boosting: l'exemple des arbres de régression

Adaboost et Gradient Boosting

Retour sur l'évaluation d'un algorithme

Apprentissage séquentiel

Online Convex Optimization

Prediction de séquences individuelles

Prise de décision séquentielle

Modèles de bandit stochastiques

L'algorithme UCB

La révolution du Boosting

Leo Breiman, 1996:

“AdaBoost with trees is the best off-the-shelf classifier in the world”

(this was before Gradient Boosting,
Random Forest and... Deep Learning)

L'algorithme Adaboost

Classification binaire: $\mathcal{Y} = \{-1, 1\}$

$$\hat{g}_n(x) = \text{sgn} \left(\sum_{b=1}^B \alpha_b \hat{g}_n^b(x) \right)$$

Le principe d'ADABOOST:

- ▶ on assigne à chaque observation i un poids w_i
- ▶ à chaque étape, un nouveau classifieur \hat{g}_n^b est entraîné sur une version pondérée de la base de donnée \mathcal{D}_n
- ▶ on calcule un score de confiance α_b pour le classifieur \hat{g}_n^b
- ▶ on met à jour les poids en augmentant les poids w_i des observations i mal prédites par le classifieur global
 - chaque classifieur améliore les prédictions sur les données mal prédites par le classifieur précédent

L'algorithme Adaboost

ADABOOST [Freund et Shapire 1997]

Paramètre: B (nombre d'itérations)
une famille de classifieurs faibles \mathcal{G}

Initialisation: poids des observations $w_i = \frac{1}{n}$ pour $i \in \{1, \dots, n\}$.

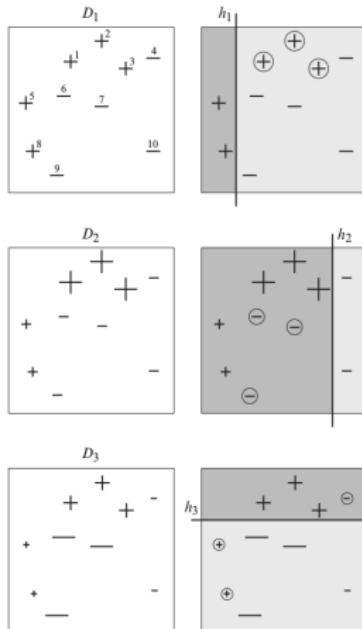
Pour $b = 1, \dots, B$:

- ▶ Calculer \hat{g}_n^b , l'ERM pour la classe \mathcal{G} sur la base de données pondérée $\mathcal{D}_n^w = (X_i, Y_i, w_i)_{1 \leq i \leq n}$
- ▶ Calculer l'erreur pondérée de \hat{g}^b : $e_b = \frac{\sum_{i=1}^n w_i \mathbb{1}_{(Y_i \neq \hat{g}_n^b(X_i))}}{\sum_{i=1}^n w_i}$
- ▶ Calculer le score de confiance $\alpha_b = \ln \left(\frac{1-e_b}{e_b} \right)$
- ▶ Mettre à jour les poids

$$w_i \leftarrow w_i \exp \left(\alpha_b \mathbb{1}_{(Y_i \neq \hat{g}_n^b(X_i))} \right).$$

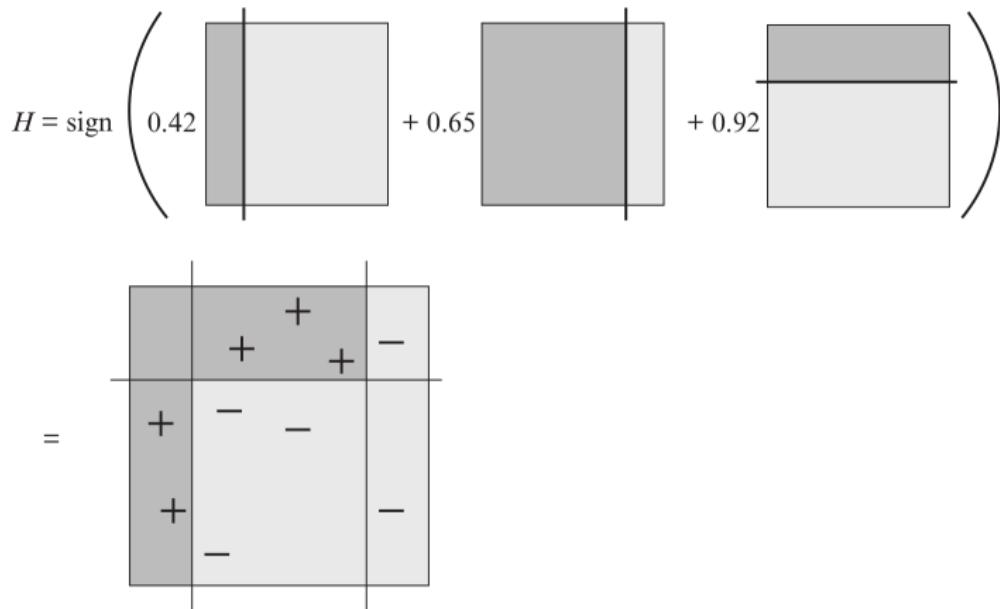
Retourner: le classifieur $\hat{g}_n(x) = \operatorname{sgn} \left(\sum_{b=1}^B \alpha_b \hat{g}_n^b(x) \right)$

Adaboost: Illustration



[Freund et Shapire]

Adaboost: Illustration



[Freund et Shapire]

Une interprétation d'Adaboost

Forward Stagewise Additive Models

$G_0(x) = 0$. On construit itérativement G_B avec, pour $b = 1, \dots, B$,

$$\begin{aligned}(\beta_b, g_b) &\in \underset{\beta \in \mathbb{R}, g \in \mathcal{G}}{\operatorname{argmin}} \sum_{i=1}^n \ell(Y_i, G_{b-1}(X_i) + \beta \times g(X_i)) \\ G_b(x) &= G_{b-1}(x) + \beta_b \times g_b(x)\end{aligned}$$

(construction pas à pas d'un modèle additif)

[Friedman 01] : ADABOOST vérifie $\hat{g}_n(x) = \operatorname{sgn}(G_b(x))$ pour

- ▶ la fonction de perte exponentielle: $\ell(a, b) = \exp(-ab)$
- ▶ une classe \mathcal{G} de prédicteurs binaires dans $\{-1, 1\}$

Une interprétation d'Adaboost

Forward Stagewise Additive Models

$G_0(x) = 0$. On construit itérativement G_B avec, pour $b = 1, \dots, B$,

$$\begin{aligned}(\beta_b, g_b) &\in \underset{\beta \in \mathbb{R}, g \in \mathcal{G}}{\operatorname{argmin}} \sum_{i=1}^n \ell(Y_i, G_{b-1}(X_i) + \beta \times g(X_i)) \\ G_b(x) &= G_{b-1}(x) + \beta_b \times g_b(x)\end{aligned}$$

(construction pas à pas d'un modèle additif)

[Friedman 01] : ADABOOST vérifie $\hat{g}_n(x) = \operatorname{sgn}(G_b(x))$ pour

- ▶ la fonction de perte exponentielle: $\ell(a, b) = \exp(-ab)$
- ▶ une classe \mathcal{G} de prédicteurs binaires dans $\{-1, 1\}$
- **idée générale:** ajouter pas-à-pas un prédicteur (pondéré) au prédicteur précédent pour réduire le risque empirique

D'autres procédures de ce type

Forward Stagewise Additive Models

$G_0(x) = 0$. On construit itérativement G_B avec, pour $b = 1, \dots, B$,

$$T_b \in \underset{T \in \mathcal{G}}{\operatorname{argmin}} \sum_{i=1}^n \ell(Y_i, G_{b-1}(X_i) + T(X_i))$$

$$G_b(x) = G_{b-1}(x) + \lambda T_b(x)$$

BOOSTED REGRESSION TREE : le prédicteur $G_b(x)$ pour

- ▶ la fonction de perte quadratique: $\ell(a, b) = (a - b)^2$
- ▶ l'ensemble \mathcal{G} des arbres à p nœuds

Gradient Boosting

Forward Stagewise Additive Models

$G_0(x) = 0$. On construit itérativement G_B avec, pour $b = 1, \dots, B$,

$$T_b \in \operatorname{argmin}_{T \in \mathcal{G}} \sum_{i=1}^n \ell(Y_i, G_{b-1}(X_i) + T(X_i))$$

$$G_b(x) = G_{b-1}(x) + \lambda T_b(x)$$

Observation: Pour une fonction de perte générale, le [calcul exact](#) de T_b n'est pas toujours possible !

Gradient Boosting

Forward Stagewise Additive Models

$G_0(x) = 0$. On construit itérativement G_B avec, pour $b = 1, \dots, B$,

$$T_b \in \operatorname{argmin}_{T \in \mathcal{G}} \sum_{i=1}^n \ell(Y_i, G_{b-1}(X_i) + T(X_i))$$

$$G_b(x) = G_{b-1}(x) + \lambda T_b(x)$$

Idée: Pour réduire le risque, on voudrait que le vecteur

$$(T_b(X_1), \dots, T_b(X_n))$$

soit proche de la direction de descente

$$\left(-\frac{\partial \ell}{\partial y} (Y_1, G_{b-1}(X_1)), \dots, -\frac{\partial \ell}{\partial y} (Y_n, G_{b-1}(X_n)) \right)$$

→ construire un arbre de régression sur ces valeurs

Gradient Boosting

Algorithm 10.3 Gradient Tree Boosting Algorithm.

1. Initialize $f_0(x) = \arg \min_{\gamma} \sum_{i=1}^N L(y_i, \gamma)$.

2. For $m = 1$ to M :

(a) For $i = 1, 2, \dots, N$ compute

$$r_{im} = - \left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)} \right]_{f=f_{m-1}}.$$

(b) Fit a regression tree to the targets r_{im} giving terminal regions R_{jm} , $j = 1, 2, \dots, J_m$.

(c) For $j = 1, 2, \dots, J_m$ compute

$$\gamma_{jm} = \arg \min_{\gamma} \sum_{x_i \in R_{jm}} L(y_i, f_{m-1}(x_i) + \gamma).$$

(d) Update $f_m(x) = f_{m-1}(x) + \sum_{j=1}^{J_m} \gamma_{jm} I(x \in R_{jm})$.

3. Output $\hat{f}(x) = f_M(x)$.

[Hastie et Tibshirani]

Python: `GradientBoostingClassifier` (package `sklearn.ensemble`)
ou `XGBoost` (implémentation plus rapide)

Bilan: Bagging versus Boosting

Bagging:

aggréger beaucoup de bons classificateurs (faible biais), faiblement correlés, dans le but de réduire la variance

Boosting:

aggréger beaucoup de classificateurs faibles (fort biais), fortement correlés, dans le but de réduire le biais

Outline

Théorie de l'apprentissage statistique

Algorithmes

Les k -plus proches voisins

Régression et régularisation

Arbres de décision

Les méthodes d'ensemble

Bagging et forêts aléatoires

Boosting: l'exemple des arbres de régression

Adaboost et Gradient Boosting

Retour sur l'évaluation d'un algorithme

Apprentissage séquentiel

Online Convex Optimization

Prediction de séquences individuelles

Prise de décision séquentielle

Modèles de bandit stochastiques

L'algorithme UCB

Cas général

A partir d'une base de test $\mathcal{D}_{n_t} = \{(X'_i, Y'_i)\}_{1 \leq i \leq n_t}$ on peut toujours calculer le risque empirique de test d'un classifieur f .

- ▶ **classification:** erreur de test

$$\frac{1}{n_t} \sum_{i=1}^{n_t} \mathbb{1}(Y'_i \neq f(X'_i))$$

(à valeurs dans $[0, 1]$)

- ▶ **régression:** erreur quadratique moyenne

$$\frac{1}{n_t} \sum_{i=1}^{n_t} (Y'_i - f(X'_i))^2$$

(peut prendre de grandes valeurs)

Classification binaire

Pour la classification binaire la [table de confusion](#) construite à partir de \mathcal{D}_{nt} donne plus d'informations sur le type d'erreurs commises par l'algorithme.

		Actual Value (as confirmed by experiment)	
		positives	negatives
Predicted Value (predicted by the test)	positives	TP True Positive	FP False Positive
	negatives	FN False Negative	TN True Negative

Pour un algorithme donné, on peut mesurer

- ▶ la sensibilité ou taux de [vrais positifs](#) (TPR) : $\frac{TP}{TP+FN}$
- ▶ la spécificité ou taux de [vrais négatifs](#) (TNR) : $\frac{TN}{TN+FP}$

Classification binaire basée sur un score

De nombreuses méthodes de classification binaires peuvent être modifiées pour produire un vecteur qui donne la probabilité d'appartenance à chacune des classes.

par exemple $\hat{p}_n(x) = (0.44, \mathbf{0.56})$

Ceci génère un score $\hat{s}_n(x)$ (la probabilité d'appartenir à la classe 1), habituellement converti en prédition de la manière suivante

- ▶ $\hat{s}_n(x) \geq 1/2$: on prédit 1
 - ▶ $\hat{s}_n(x) < 1/2$: on prédit 0
- On peut remplacer le seuil $1/2$ par un seuil arbitraire $t \in [0, 1]$.

Classification binaire basée sur un score

- Pour chaque valeur possible du seuil, on peut calculer les caractéristiques du classifieur obtenu sur la base de test:

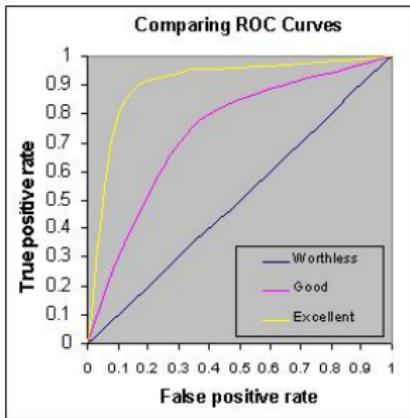
individu	score	vrai label
1	1	1
2	0.95	1
3	0.9	1
4	0.85	0
5	0.8	1
6	0.75	0
7	0.7	0
8	0.65	1
9	0.6	0
10	0.55	0

individu	score	vrai label
11	0.5	0
12	0.45	1
13	0.4	0
14	0.35	0
15	0.3	0
16	0.25	0
17	0.2	0
18	0.15	0
19	0.1	0
20	0.05	0

(**TPR** : 5/6, **FPR** : 6/14) → un point sur la courbe ROC

Classification binaire basée sur un score

La **courbe ROC** (pour Receiving Operator Characteristic) représente le Taux de Vrais Positifs en fonction du Taux de Faux Positifs pour tous les seuils possibles.



L'**aire sous la courbe ROC** fournit un indicateur numérique de la performance d'un algorithme (**AUC score**).

Scikit-learn: fonction `sklearn.metrics.roc_curve`

Outline

Théorie de l'apprentissage statistique

Algorithmes

Les k -plus proches voisins

Régression et régularisation

Arbres de décision

Les méthodes d'ensemble

Bagging et forêts aléatoires

Boosting: l'exemple des arbres de régression

Adaboost et Gradient Boosting

Retour sur l'évaluation d'un algorithme

Apprentissage séquentiel

Online Convex Optimization

Prediction de séquences individuelles

Prise de décision séquentielle

Modèles de bandit stochastiques

L'algorithme UCB

Batch versus Online

Apprentissage supervisé :

grand nombre de données étiquetées (**batch**),
prédire l'étiquette de nouvelles données

Apprentissage séquentiel :

données collectées séquentiellement, prédire à la volée (**online**)
l'étiquette de chaque nouvelle donnée

(prédition de l'évolution d'un marché, de la fiabilité d'un nouveau client, de la consommation électrique...)

Online Learning: cadre général

Apprentissage en ligne

A chaque instant $t = 1, \dots, n$,

1. observer $x_t \in \mathcal{X}$
2. prédire $\hat{y}_t \in \mathcal{Y}$
3. y_t est révélé et on reçoit la perte $\ell(y_t, \hat{y}_t)$.

But: Minimiser la perte cumulée (liée au nombre et à la gravité des erreurs de prédiction)

$$\sum_{t=1}^n \ell(y_t, \hat{y}_t)$$

A quoi peut-on se comparer ?

- au meilleur prédicteur dans une famille de fonctions \mathcal{G}
(ex: prédicteurs linéaires)
- à des experts ("boîte noire") proposant des prédictions

Outline

Théorie de l'apprentissage statistique

Algorithmes

Les k -plus proches voisins

Régression et régularisation

Arbres de décision

Les méthodes d'ensemble

Bagging et forêts aléatoires

Boosting: l'exemple des arbres de régression

Adaboost et Gradient Boosting

Retour sur l'évaluation d'un algorithme

Apprentissage séquentiel

Online Convex Optimization

Prediction de séquences individuelles

Prise de décision séquentielle

Modèles de bandit stochastiques

L'algorithme UCB

Online Learning: un cadre particulier

Une catégorie d'algorithmes

Soit \mathcal{G} une classe de prédicteurs. A chaque instant t ,

1. observer $x_t \in \mathcal{X}$
2. choisir $g_t \in \mathcal{G}$ et prédire $\hat{y}_t = g_t(x_t)$
3. y_t est révélé et on reçoit la perte $\ell(y_t, \hat{y}_t)$.

But: Minimiser le [regret](#), c'est-à-dire la différence de notre perte cumulée avec celle obtenue par le meilleur prédicteur de la classe:

$$R_n = \sum_{t=1}^n \ell(y_t, g_t(x_t)) - \min_{g \in \mathcal{G}} \sum_{t=1}^n \ell(y_t, g(x_t)).$$

- Sans hypothèse sur la fonction de perte, on ne peut pas toujours atteindre un regret faible

Optimisation Convexe Séquentielle

Online Convex Optimization

A chaque instant t ,

1. choisir $w_t \in \mathcal{K}$ (convexe)
2. l'environnement révèle une fonction de perte convexe
 $\ell_t : \mathcal{K} \rightarrow \mathbb{R}$
3. on reçoit la perte $\ell_t(w_t)$.

But: Minimiser le regret

$$R_n = \sum_{t=1}^n \ell_t(w_t) - \min_{w \in \mathcal{K}} \sum_{t=1}^n \ell_t(w).$$

Exemple: Régression linéaire / logistique en ligne

$$\ell_t(w) = (Y_t - \langle w | X_t \rangle)^2 \quad \ell_t(w) = \ln \left(1 + e^{-Y_t \langle w | X_t \rangle} \right)$$

Online Gradient Descent

Algorithme de descente de gradient **en ligne**

$$\begin{cases} w_0 &= 0 \\ w_{t+1} &= \Pi_{\mathcal{K}}(w_t - \eta \nabla \ell_t(w_t)) \end{cases}$$

où $\Pi_{\mathcal{K}}(x) = \operatorname{argmin}_{u \in \mathcal{K}} \|x - u\|$ est la projection sur le convexe \mathcal{K} .

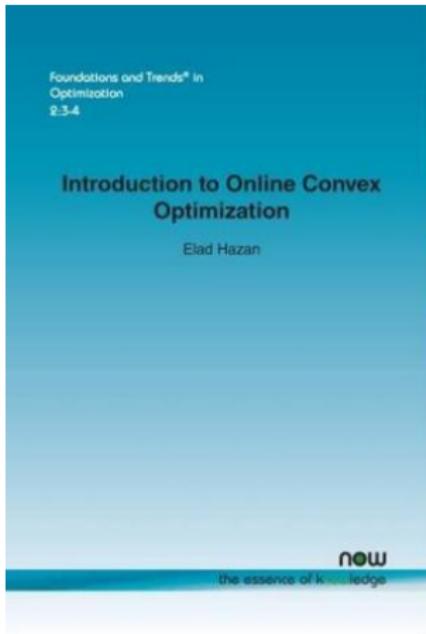
Théorème

Si $\|\nabla \ell_t(w)\| \leq L$ et $\|w\| \leq R$ pour tout $w \in \mathcal{K}$, alors

$$R_n = \max_{w \in \mathcal{K}} \sum_{t=1}^n (\ell_t(w_t) - \ell_t(w)) \leq \frac{R^2}{2\eta} + \frac{\eta L^2 n}{2}$$

Corollaire: pour le choix $\eta_n = \frac{R}{L\sqrt{n}}$, on obtient $R_n \leq RL\sqrt{n}$

Pour aller plus loin



[The OCO book]

Outline

Théorie de l'apprentissage statistique

Algorithmes

Les k -plus proches voisins

Régression et régularisation

Arbres de décision

Les méthodes d'ensemble

Bagging et forêts aléatoires

Boosting: l'exemple des arbres de régression

Adaboost et Gradient Boosting

Retour sur l'évaluation d'un algorithme

Apprentissage séquentiel

Online Convex Optimization

Prediction de séquences individuelles

Prise de décision séquentielle

Modèles de bandit stochastiques

L'algorithme UCB

Prediction with expert advice

- ▶ on cherche à prédire séquentiellement un phénomène (bourse, météo, consommation...)
- ▶ on ne fait aucune hypothèse probabiliste sur ce phénomène
- ▶ on s'appuie sur des experts (boîtes noires) ± fiables
- ▶ on cherche à faire au moins aussi bien que le meilleur expert



Le jeu de prédition

Prediction with Expert Advice

A chaque instant t ,

1. chaque expert k fait une prédition $z_{k,t}$ que l'on observe
2. on fait une prédition \hat{y}_t
3. y_t est révélé et on subit une perte $\ell(\hat{y}_t, y_t)$.

Chaque expert subit la perte $\ell(z_{k,t}, y_t)$.

Hypothèses:

- ▶ K experts
- ▶ \mathcal{Y} espace des prédictions supposé convexe
- ▶ $z \mapsto \ell(z, y)$ convexe pour tout y

Remarque: pas de variables explicatives $x_t \in \mathcal{X}$

(mais on peut imaginer que x_t est pris en compte par les experts)

Le jeu de prédition

Prediction with Expert Advice

A chaque instant t ,

1. chaque expert k fait une prédition $z_{k,t}$ que l'on observe
2. on fait une prédition \hat{y}_t
3. y_t est révélé et on subit une perte $\ell(\hat{y}_t, y_t)$.
Chaque expert subit la perte $\ell(z_{k,t}, y_t)$.

Objectif: étant donné

- ▶ $\hat{L}_n = \sum_{t=1}^n \ell(\hat{y}_t, y_t)$ la perte cumulée de notre stratégie
- ▶ $L_{k,n} = \sum_{t=1}^n \ell(z_{k,t}, y_t)$ la perte cumulée de l'expert k

on veut un faible **regret** :

$$R_n = \hat{L}_n - \underbrace{\min_{k \in K} L_{k,n}}_{\text{perte cumulée du meilleur expert}}$$

Weighted Average Prediction

Idée: prédire une moyenne pondérée des prédictions faites par les experts, $(z_{k,t})_{1 \leq k \leq K}$:

$$\hat{y}_t = \frac{\sum_{k=1}^K w_{k,t} z_{k,t}}{\sum_{k=1}^K w_{k,t}} = \sum_{k=1}^K \left(\frac{w_{k,t}}{\sum_{i=1}^K w_{i,t}} \right) z_{k,t}.$$

$(\hat{y}_t \in \mathcal{Y} \text{ car } \mathcal{Y} \text{ est convexe})$

$w_{k,t}$ est le poids associé à l'expert k à l'instant t .

→ poids dépendant de la qualité des experts jusqu'à maintenant:

$$w_{k,t} = F(L_{k,t-1}) \quad \text{avec } F \text{ une fonction décroissante}$$

Weighted Average Prediction

Idée: prédire une moyenne pondérée des prédictions faites par les experts, $(z_{k,t})_{1 \leq k \leq K}$:

$$\hat{y}_t = \frac{\sum_{k=1}^K w_{k,t} z_{k,t}}{\sum_{k=1}^K w_{k,t}} = \sum_{k=1}^K \left(\frac{w_{k,t}}{\sum_{i=1}^K w_{i,t}} \right) z_{k,t}.$$

$(\hat{y}_t \in \mathcal{Y} \text{ car } \mathcal{Y} \text{ est convexe})$

$w_{k,t}$ est le poids associé à l'expert k à l'instant t .

→ poids dépendant de la qualité des experts jusqu'à maintenant:

$$w_{k,t} = F(L_{k,t-1}) \quad \text{avec } F \text{ une fonction décroissante}$$

Choix typique: $F(x) = \exp(-\eta x)$.

Exponentially Weighted Average

Algorithme EWA(η)

Paramètre: $\eta > 0$.

Initialisation: pour tout $k \in \{1, \dots, K\}$, $w_{k,1} = \frac{1}{K}$.

Pour $t = 1, \dots, n$

1. Observer les prédictions des experts: $(z_{k,t})_{1 \leq k \leq K}$
2. Calculer la moyenne pondérée

$$\hat{y}_t = \frac{\sum_{k=1}^K w_{k,t} z_{k,t}}{\sum_{k=1}^K w_{k,t}}$$

3. Prédire \hat{y}_t et calculer les pertes

$$\ell(\hat{y}_t, y_t) \quad \text{et} \quad \ell_{k,t} = \ell(z_{k,t}, y_t)$$

4. Calculer les poids pour l'étape suivante

$$w_{k,t+1} = w_{k,t} \exp(-\eta \ell_{k,t})$$

Regret de l'algorithme EWA

Théorème

Supposons que pour tout $(a, b) \in \mathcal{Y}$, $\ell(a, b) \leq 1$. Alors pour tout $\eta > 0$ et pour tout $n \geq 0$, le regret de l'algorithme EWA(η) vérifie

$$\hat{L}_n - \min_{k=1,\dots,K} L_{k,n} \leq \frac{\ln(K)}{\eta} + \frac{n\eta}{8}$$

- ▶ En choisissant $\eta = \sqrt{8 \ln(K)/n}$, on obtient $R_n \leq \sqrt{\frac{n \ln(K)}{2}}$.
- ▶ η peut aussi être choisi sans connaître l'“horizon” n : pour un choix qui dépend du temps, $\eta_t = \sqrt{8 \ln(K)/t}$ on a

$$R_n \leq 2\sqrt{\frac{n \ln(K)}{2}} + \sqrt{\frac{\ln(K)}{8}}$$

Et sans la convexité ?

L'algorithme EWA suppose que l'espace \mathcal{Y} des prédictions est **convexe**, ainsi que la fonction de perte $\ell(\cdot, y)$.
C'est loin d'être toujours le cas !

Cas particulier:

$$\mathcal{Y} = \{-1, 1\} \quad \ell(a, b) = \mathbb{1}_{(a \neq b)}$$

Pour toute stratégie **déterministe**, il existe une séquence (y_1, \dots, y_n) telle que $\hat{L}_n = n$! $(y_t = -\hat{y}_t)$

Mais, on peut avoir de meilleurs experts, par exemple $K = 2$ et

$$\begin{cases} z_{1,t} = 1 & \text{pour tout } t \\ z_{2,t} = -1 & \text{pour tout } t \end{cases} \Rightarrow \min_{k \in \{1,2\}} L_{k,n} \leq \frac{n}{2} \quad \text{donc } R_n \geq \frac{n}{2}$$

Et sans la convexité ?

L'algorithme EWA suppose que l'espace \mathcal{Y} des prédictions est **convexe**, ainsi que la fonction de perte $\ell(\cdot, y)$.
C'est loin d'être toujours le cas !

Cas particulier:

$$\mathcal{Y} = \{-1, 1\} \quad \ell(a, b) = \mathbb{1}_{(a \neq b)}$$

Pour toute stratégie **déterministe**, il existe une séquence (y_1, \dots, y_n) telle que $\hat{L}_n = n$! $(y_t = -\hat{y}_t)$

Mais, on peut avoir de meilleurs experts, par exemple $K = 2$ et

$$\begin{cases} z_{1,t} = 1 & \text{pour tout } t \\ z_{2,t} = -1 & \text{pour tout } t \end{cases} \Rightarrow \min_{k \in \{1,2\}} L_{k,n} \leq \frac{n}{2} \quad \text{donc } R_n \geq \frac{n}{2}$$

→ Il faut **randomiser** !

Un jeu de prédiction plus général

Prediction avec avis d'expert dans un environnement général

A chaque instant t ,

1. l'environnement choisit un vecteur $\ell_t = (\ell_{1,t}, \dots, \ell_{K,t}) \in \mathbb{R}^K$ (représentant la perte de chaque expert)
2. on choisit un expert $k_t \in \{1, \dots, K\}$ et on prédit ce que l'expert suggère ($z_{k_t,t} \in \mathcal{Y}$)
3. on subit la perte associée $\ell_{k_t,t}$ et on observe le vecteur ℓ_t

Objectif: Minimiser le **regret**, défini par

$$R_n = \underbrace{\sum_{t=1}^n \ell_{k_t,t}}_{\text{notre perte cumulée}} - \underbrace{\min_{k \in \{1, \dots, K\}} \sum_{t=1}^n \ell_{k,t}}_{\text{celle du meilleur expert}}.$$

→ le choix d'expert doit être **randomisé**

Exponentially Weighted Forecaster

Algorithme EWF(η) (ou HEDGE)

Paramètre: $\eta > 0$.

Initialisation: pour tout $k \in \{1, \dots, K\}$, $w_{k,1} = \frac{1}{K}$.

Pour $t = 1, \dots, n$

1. Observer les prédictions des experts: $(z_{k,t})_{1 \leq k \leq K}$
2. Calculer le vecteur de probabilités $p_t = (p_{1,t}, \dots, p_{K,t})$ où

$$p_{k,t} = \frac{w_{k,t}}{\sum_{i=1}^K w_{i,t}}$$

3. Tirer un expert $k_t \sim p_t$, càd $\mathbb{P}(k_t = k) = p_{k,t}$
4. Prédire $\hat{y}_t = z_{k_t,t}$ et observer les pertes

$$\ell_{k,t} \quad \text{pour tout } k \in \{1, \dots, K\}$$

5. Calculer les poids pour l'étape suivante

$$w_{k,t+1} = w_{k,t} \exp(-\eta \ell_{k,t})$$

Résultats pour EWF

L'algorithme comprenant un aléa, on s'intéresse au [regret moyen](#)

$$\mathbb{E}[R_n] = \mathbb{E} \left[\sum_{t=1}^n \ell_{k_t, t} - \min_{k \in \{1, \dots, K\}} \sum_{t=1}^n \ell_{k, t} \right].$$

Théorème

Supposons que pour tout $k, t, \ell_{k,t} \leq 1$. Alors, pour tout $\eta > 0$ et pour tout $n \geq 0$, l'algorithme EWF(η) vérifie

$$\sum_{t=1}^n p_t^T \ell_t - \min_{k \in \{1, \dots, K\}} \sum_{t=1}^T \ell_{k,t} \leq \frac{\ln(K)}{\eta} + \frac{n\eta}{2}$$

(analyse similaire à celle d'EWA)

Remarque: $\mathbb{E} [\ell_{k_t, t}] = \mathbb{E} \left[\sum_{k=1}^K p_{k,t} \ell_{k,t} \right] = \mathbb{E} [p_t^T \ell_t]$.

Résultats pour EWF

Théorème

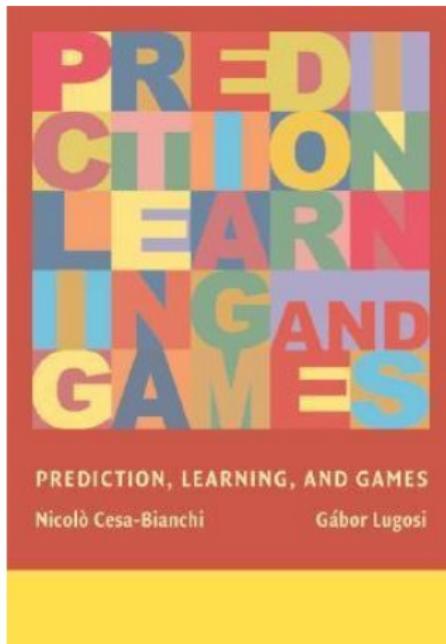
Pour le choix $\eta = \sqrt{\frac{8 \ln(K)}{n}}$, l'algorithme EWF(η) vérifie

$$\mathbb{E}[R_n] \leq \sqrt{\frac{n \ln(K)}{2}}$$

Extensions:

- ▶ d'autres notions de regret: se comparer à la meilleure combinaison d'experts, traquer le meilleur expert
- ▶ jeu à information partielle: on observe *uniquement* la perte de l'expert choisi (bandit)

Pour aller plus loin...



[Prediction, Learning and Games]

Outline

Théorie de l'apprentissage statistique

Algorithmes

Les k -plus proches voisins

Régression et régularisation

Arbres de décision

Les méthodes d'ensemble

Bagging et forêts aléatoires

Boosting: l'exemple des arbres de régression

Adaboost et Gradient Boosting

Retour sur l'évaluation d'un algorithme

Apprentissage séquentiel

Online Convex Optimization

Prediction de séquences individuelles

Prise de décision séquentielle

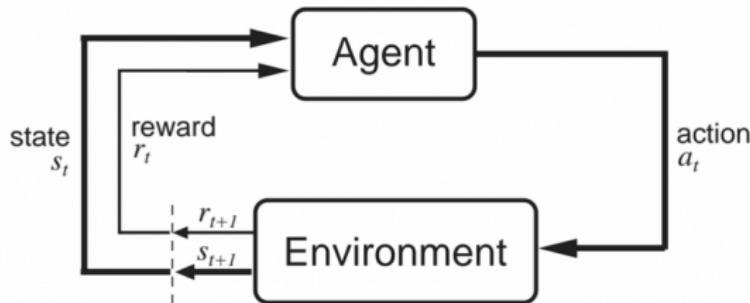
Modèles de bandit stochastiques

L'algorithme UCB

Apprentissage par renforcement

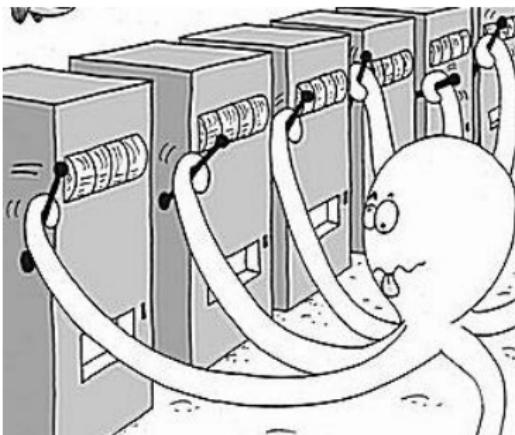
Cadre où on est amené à prendre des décisions qui conduisent à des récompenses et permettent d'acquérir de l'information.

→ ex. recommander un produit à un client



Formalisation: un agent dans un état s_t choisit une action a_t , puis reçoit une récompense r_{t+1} et se retrouve dans un nouvel état s_{t+1} .

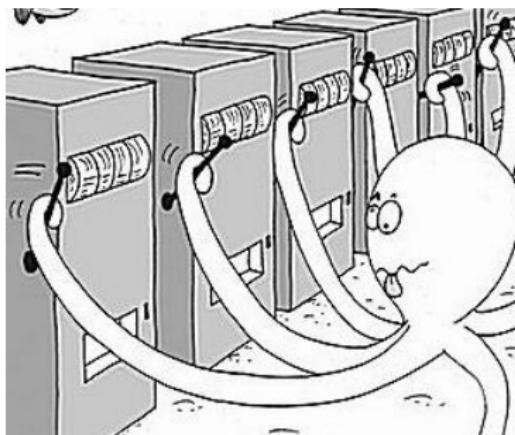
Un cadre simplifié: le bandit à plusieurs bras



- ▶ un agent
- ▶ K actions possibles, qui donnent des récompenses
- ▶ pas d'état

Objectif: maximiser la somme des récompenses obtenues.

Un cadre simplifié: le bandit à plusieurs bras



A chaque instant $t = 1, \dots, T$, l'agent

- ▶ choisit une action $A_t \in \{1, \dots, K\}$
- ▶ observe un récompense R_t liée à l'action choisie.

Stratégie séquentielle: A_t dépend de $A_1, R_1, \dots, A_{t-1}, R_{t-1}$.

Exemples d'applications

Essais cliniques

- ▶ K traitements possibles (d'effet inconnu)



- ▶ Quel traitement allouer à chaque patient en fonction des effets observés sur les patients précédents?

Publicité en ligne

- ▶ K publicités pouvant être affichées



- ▶ Quelle publicité montrer à un nouvel utilisateur en fonction des clics des utilisateurs précédents?

Quelle hypothèse sur les récompenses ?

Comme dans le cadre de la prédiction de séquences individuelles, on peut essayer de faire aussi peu d'hypothèses que possible.

- ▶ action k : suite (arbitraire) de récompenses

$$r_{k,1}, r_{k,2} \dots, r_{k,T}$$

- ▶ récompense à l'instant t : celle de l'action choisie $R_t = r_{A_t,t}$.
- ⚠ on n'observe pas $r_{k,t}$ pour $k \neq A_t$!

Objectif: maximiser la somme des récompenses

↔ minimiser le regret

$$\mathcal{R}_T = \underbrace{\max_{k \in \{1, \dots, K\}} \sum_{t=1}^T r_{k,t}}_{\text{récompenses de la meilleure action}} - \underbrace{\sum_{t=1}^T R_t}_{\text{notre récompense cumulée}}$$

Une variante de EWF: l'algorithme EXP3

Hypothèse: $r_{k,t} \in [0, 1]$ pour tout t

Paramètre: $\eta > 0$.

Initialisation: pour tout $k \in \{1, \dots, K\}$, $w_{k,1} = \frac{1}{K}$.

Pour $t = 1, \dots, n$

1. Calculer le vecteur de probabilités $p_t = (p_{1,t}, \dots, p_{K,t})$ où

$$p_{k,t} = \frac{w_{k,t}}{\sum_{i=1}^K w_{i,t}}$$

2. Choisir une action $A_t \sim p_t$, càd $\mathbb{P}(A_t = k) = p_{k,t}$
3. Observer la récompense $R_t = r_{A_t,t}$.
4. Calculer la perte corrigée associé à l'action choisie:

$$\tilde{L}_t = \frac{1 - R_t}{p_{A_t,t}}$$

5. Mettre à jour le poids de l'action choisie

$$w_{A_t,t+1} = w_{A_t,t} \exp(-\eta \tilde{L}_t)$$

L'algorithme EXP3

Théorème

Pour le choix

$$\eta = \sqrt{\frac{\log(K)}{KT}}$$

l'algorithme EXP3(η) vérifie

$$\mathbb{E}[\mathcal{R}_T] \leq \sqrt{2 \ln(K) \sqrt{KT}}$$

- on peut obtenir de meilleurs résultats en **exploitant des hypothèses stochastiques sur la génération des récompenses.**

Outline

Théorie de l'apprentissage statistique

Algorithmes

Les k -plus proches voisins

Régression et régularisation

Arbres de décision

Les méthodes d'ensemble

Bagging et forêts aléatoires

Boosting: l'exemple des arbres de régression

Adaboost et Gradient Boosting

Retour sur l'évaluation d'un algorithme

Apprentissage séquentiel

Online Convex Optimization

Prediction de séquences individuelles

Prise de décision séquentielle

Modèles de bandit stochastiques

L'algorithme UCB

Le modèle de bandit stochastique

K bras (actions) = K lois de probabilités



ν_1



ν_2



ν_3



ν_4



ν_5

À l'instant t , un agent

- ▶ sélectionne le bras A_t
- ▶ observe une récompense tirée sous la distribution correspondant au bras : $R_t \sim \nu_{A_t}$

La stratégie d'échantillonnage (A_t) est séquentielle :

$$A_{t+1} = F_t(A_1, X_1, \dots, A_t, X_t).$$

Objectif: maximiser $\mathbb{E} \left[\sum_{t=1}^T R_t \right]$.

Exemple: récompenses binaires

K bras (actions) = K lois de probabilités

 $\mathcal{B}(\mu_1)$  $\mathcal{B}(\mu_2)$  $\mathcal{B}(\mu_3)$  $\mathcal{B}(\mu_4)$  $\mathcal{B}(\mu_5)$

À l'instant t , un agent

- ▶ sélectionne le bras A_t
- ▶ observe une récompense $R_t \sim \mathcal{B}(\mu_{A_t})$

$$\mathbb{P}(R_t = 1 | A_t = a) = \mu_a \text{ et } \mathbb{P}(R_t = 0 | A_t = a) = 1 - \mu_a.$$

“patient guéri”

“patient non guéri”

La stratégie d'échantillonnage (A_t) est séquentielle :

$$A_{t+1} = F_t(A_1, X_1, \dots, A_t, X_t).$$

Objectif: maximiser le nombre moyen de patients guéris.

Regret pour les bandits stochastiques

$$\mu^* = \max_a \mu_a \quad a^* = \operatorname{argmax}_a \mu_a$$

L'agent cherche une stratégie qui maximise, *en moyenne*,

$$\mathbb{E} \left[\sum_{t=1}^T R_t \right]$$

Il voudrait

$$\mathbb{E} \left[\sum_{t=1}^T R_t \right] \simeq \underbrace{T\mu^*}_{\text{gains moyen d'une stratégie ne tirant que le bras } a^*}$$

et cherche donc à minimiser le **regret (moyen)** :

$$\mathcal{R}(T) = \underbrace{T\mu^*}_{\text{récompense cumulée d'une stratégie oracle}} - \underbrace{\mathbb{E} \left[\sum_{t=1}^T R_t \right]}_{\text{récompense cumulée de la stratégie } (A_t)} .$$

Une réécriture du regret

Pour simplifier les notations, on supposera dans la suite

$$\mu^* = \mu_1 > \mu_2 \geq \dots \geq \mu_K.$$

Le regret peut se réécrire

$$\mathcal{R}(T) = \sum_{a=2}^K (\mu_1 - \mu_a) \times \mathbb{E}[N_a(T)]$$

où $N_a(T)$ est le nombre de tirages du bras a jusqu'à l'instant T

Pour minimiser le regret:

- Tirer aussi peu que possible les bras sous-optimaux !
- Réaliser un compromis entre exploration et exploitation

Premiers exemples de stratégies

- ▶ **Idée 1 :** Tirer chaque bras T/K fois

⇒ EXPLORATION

$$\mathcal{R}(T) = \left(\frac{1}{K} \sum_{a=2}^K (\mu_1 - \mu_a) \right) T$$

Premiers exemples de stratégies

- ▶ **Idée 1 :** Tirer chaque bras T/K fois

⇒ EXPLORATION

$$\mathcal{R}(T) = \left(\frac{1}{K} \sum_{a=2}^K (\mu_1 - \mu_a) \right) T$$

- ▶ **Idée 2 :** Faire confiance au meilleur empirique jusqu'ici

$$A_{t+1} = \operatorname{argmax}_a \hat{\mu}_a(t)$$

où

$$\hat{\mu}_a(t) = \frac{\text{somme des récompenses issues du bras } a}{\text{nombre de sélections du bras } a}$$

est un estimateur de la moyenne inconnue μ_a .

⇒ EXPLOITATION

$$\mathcal{R}(T) \geq (1 - \mu_1) \times \mu_2 \times (\mu_1 - \mu_2) T$$

Premiers exemples de stratégies

- ▶ **Idée 1 :** Tirer chaque bras T/K fois

⇒ EXPLORATION

$$\mathcal{R}(T) = \left(\frac{1}{K} \sum_{a=2}^K (\mu_1 - \mu_a) \right) T$$

- ▶ **Idée 2 :** Faire confiance au meilleur empirique jusqu'ici

$$A_{t+1} = \operatorname{argmax}_a \hat{\mu}_a(t)$$

où

$$\hat{\mu}_a(t) = \frac{Y_{a,1} + \cdots + Y_{a,N_a(t)}}{N_a(t)}$$

est un **estimateur** de la moyenne inconnue μ_a .

⇒ EXPLOITATION

$$\mathcal{R}(T) \geq (1 - \mu_1) \times \mu_2 \times (\mu_1 - \mu_2) T$$

Une meilleure idée: explorer puis exploiter

Etant donné $m \in \{1, \dots, T/K\}$, on

- ▶ tire chaque bras m fois
- ▶ détermine le meilleur empirique $\hat{a} = \operatorname{argmax}_a \hat{\mu}_a(Km)$
- ▶ on sélectionne ce bras jusqu'à la fin

$$A_{t+1} = \hat{a} \text{ pour } t \geq Km$$

⇒ EXPLORATION puis EXPLOITATION

Une meilleure idée: explorer puis exploiter

Etant donné $m \in \{1, \dots, T/K\}$, on

- ▶ tire chaque bras m fois
- ▶ détermine le meilleur empirique $\hat{a} = \operatorname{argmax}_a \hat{\mu}_a(Km)$
- ▶ on sélectionne ce bras jusqu'à la fin

$$A_{t+1} = \hat{a} \text{ pour } t \geq Km$$

⇒ EXPLORATION puis EXPLOITATION

Analyse pour deux bras. $\Delta = \mu_1 - \mu_2$.

$$\mathcal{R}(T) \leq \Delta m + T\Delta \exp\left(-\frac{m\Delta^2}{2}\right)$$

→ Comment choisir m ?

Une meilleure idée: explorer puis exploiter

Etant donné $m \in \{1, \dots, T/K\}$, on

- ▶ tire chaque bras m fois
- ▶ détermine le meilleur empirique $\hat{a} = \operatorname{argmax}_a \hat{\mu}_a(Km)$
- ▶ on sélectionne ce bras jusqu'à la fin

$$A_{t+1} = \hat{a} \text{ pour } t \geq Km$$

⇒ EXPLORATION puis EXPLOITATION

Analyse pour deux bras. $\Delta = \mu_1 - \mu_2$.

$$\mathcal{R}(T) \leq \frac{2}{\Delta} \left[\log \left(\frac{T\Delta^2}{2} \right) + 1 \right]$$

→ en prenant $m = \frac{2}{\Delta^2} \log \left(\frac{T\Delta^2}{2} \right)$

Une meilleure idée: explorer puis exploiter

Etant donné $m \in \{1, \dots, T/K\}$, on

- ▶ tire chaque bras m fois
- ▶ détermine le meilleur empirique $\hat{a} = \operatorname{argmax}_a \hat{\mu}_a(Km)$
- ▶ on sélectionne ce bras jusqu'à la fin

$$A_{t+1} = \hat{a} \text{ pour } t \geq Km$$

⇒ EXPLORATION puis EXPLOITATION

Analyse pour deux bras. $\Delta = \mu_1 - \mu_2$.

$$\mathcal{R}(T) \leq \frac{2}{\Delta} \log \left(\frac{T\Delta^2}{2} \right) + \frac{2}{\Delta}$$

→ en prenant $m = \frac{2}{\Delta^2} \log \left(\frac{T\Delta^2}{2} \right)$

Problème: nécessite de connaître Δ ...

Plan

Théorie de l'apprentissage statistique

Algorithmes

Les k -plus proches voisins

Régression et régularisation

Arbres de décision

Les méthodes d'ensemble

Bagging et forêts aléatoires

Boosting: l'exemple des arbres de régression

Adaboost et Gradient Boosting

Retour sur l'évaluation d'un algorithme

Apprentissage séquentiel

Online Convex Optimization

Prediction de séquences individuelles

Prise de décision séquentielle

Modèles de bandit stochastiques

L'algorithme UCB

Le principe d'optimisme

- ▶ Pour chaque bras a , on suppose construit un **intervalle de confiance** sur la moyenne inconnue μ_a :

$$\mathcal{I}_a(t) = [\text{LCB}_a(t), \text{UCB}_a(t)]$$

LCB = **L**ower **C**onfidence **B**ound

UCB = **U**pper **C**onfidence **B**ound

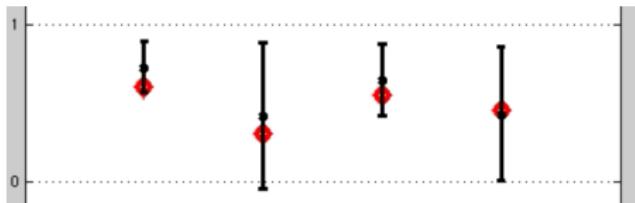


Figure: Intervalles de confiance sur les bras après t instants

Le principe d'optimisme

- ▶ On applique le principe suivant :

«agir comme si on se trouvait dans le meilleur des modèles possible»

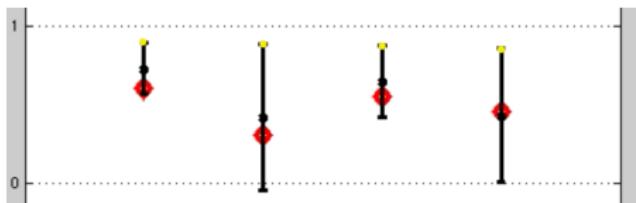


Figure: Intervalles de confiance sur les bras après t instants

- ▶ Ceci revient à choisir à l'instant $t + 1$

$$A_{t+1} = \operatorname{argmax}_a \text{UCB}_a(t)$$

Comment construire des intervalles de confiance?

On cherche $\text{UCB}_a(t)$ tel que

$$\mathbb{P}(\mu_a \leq \text{UCB}_a(t)) \simeq 1 - \frac{1}{t}.$$

→ utiliser une inégalité de concentration

Inégalité de Hoeffding

Z_i i.i.d. de moyenne μ avec $Z_i \in [0, 1]$. Pour tout entier $s \geq 1$

$$\mathbb{P}\left(\frac{Z_1 + \cdots + Z_s}{s} \geq \mu + x\right) \leq e^{-2x^2 s}$$

On peut ainsi montrer que

$$\mathbb{P}\left(\mu_a \leq \hat{\mu}_a(t) + \sqrt{\frac{\alpha \log(t)}{N_a(t)}}\right) \geq 1 - \frac{1}{t^{2\alpha-1}}.$$

Comment construire des intervalles de confiance?

On cherche $\text{UCB}_a(t)$ tel que

$$\mathbb{P}(\mu_a \leq \text{UCB}_a(t)) \simeq 1 - \frac{1}{t}.$$

→ utiliser une inégalité de concentration

Inégalité de Hoeffding

Z_i i.i.d. de moyenne μ avec $Z_i \in [0, 1]$. Pour tout entier $s \geq 1$

$$\mathbb{P}\left(\frac{Z_1 + \cdots + Z_s}{s} \leq \mu - x\right) \leq e^{-2x^2 s}$$

On peut ainsi montrer que

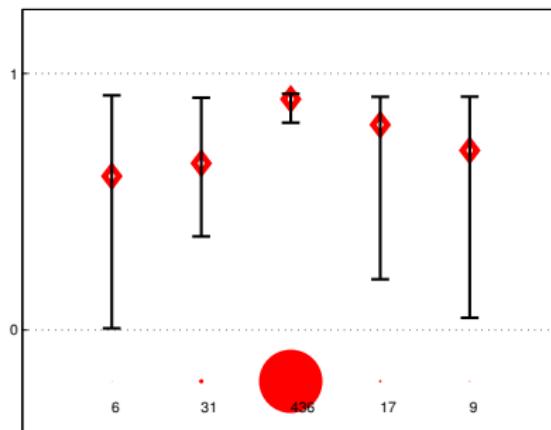
$$\mathbb{P}\left(\mu_a \leq \hat{\mu}_a(t) + \sqrt{\frac{\alpha \log(t)}{N_a(t)}}\right) \geq 1 - \frac{1}{t^{2\alpha-1}}.$$

L'algorithme UCB

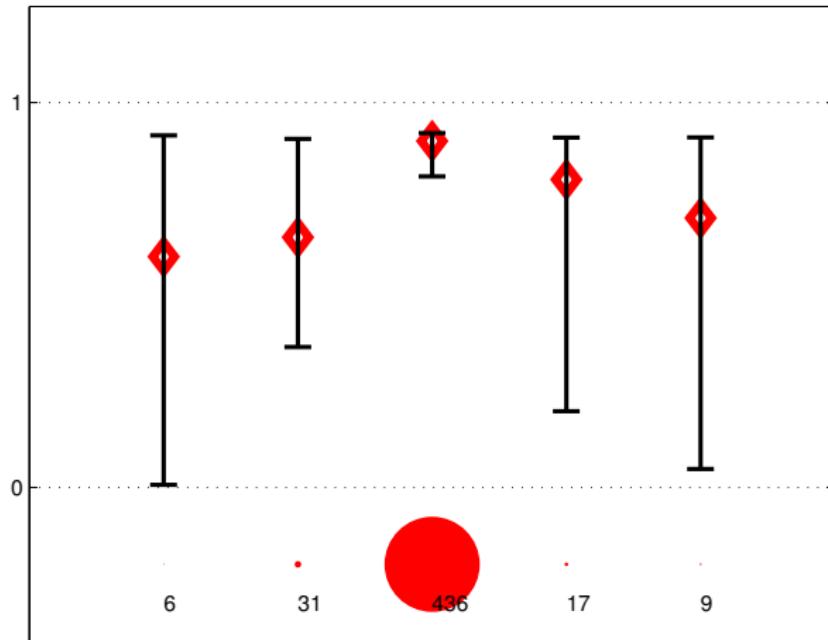
L'algorithme UCB(α) choisit $A_{t+1} = \operatorname{argmax}_a \text{UCB}_a(t)$ avec

$$\text{UCB}_a(t) = \underbrace{\hat{\mu}_a(t)}_{\text{terme d'exploitation}} + \underbrace{\sqrt{\frac{\alpha \log(t)}{N_a(t)}}}_{\text{bonus d'exploration}}.$$

popularisé par [Auer et al. 02] : UCB1, $\alpha = 2$



UCB en action !



Borne de regret pour UCB1

Théorème [Auer et al. 02]

Pour l'algorithme UCB(α) avec $\alpha = 3/2$,

$$\mathbb{E}[N_a(T)] \leq \frac{6}{(\mu_1 - \mu_a)^2} \log(T) + 1 + \frac{\pi^2}{3}.$$

2 bras : $\mathcal{R}(T) \leq \frac{6}{\Delta} \log(T) + C.$

Borne de regret pour UCB1

Théorème [Auer et al. 02]

Pour l'algorithme UCB(α) avec $\alpha = 3/2$,

$$\mathbb{E}[N_a(T)] \leq \frac{6}{(\mu_1 - \mu_a)^2} \log(T) + 1 + \frac{\pi^2}{3}.$$

2 bras : $\mathcal{R}(T) \leq \frac{6}{\Delta} \log(T) + C.$

- ▶ L'analyse peut être raffinée pour $\alpha \simeq 1/2$:

2 bras : $\mathcal{R}(T) \leq \frac{1}{2\Delta} \log(T) + C \sqrt{\log(T)}.$

Bilan: L'algorithme UCB combine exploration et exploitation et a un regret plus faible que celui des stratégies précédentes.

Pour aller plus loin

The Bandit Book

(suivez le lien)

Bandits pour les systèmes de recommandation

Example: recommandation de film



Quel film Netflix va recommander à un utilisateur, en se basant sur les notes données par les utilisateurs précédents?

- On a besoin d'un modèle prenant en compte les caractéristiques des films présentés

Exemple: le modèle logistique

$$\mathbb{P}(X_t = + | A_t = a) = \frac{1}{1 + e^{-x_a^T \theta}}$$

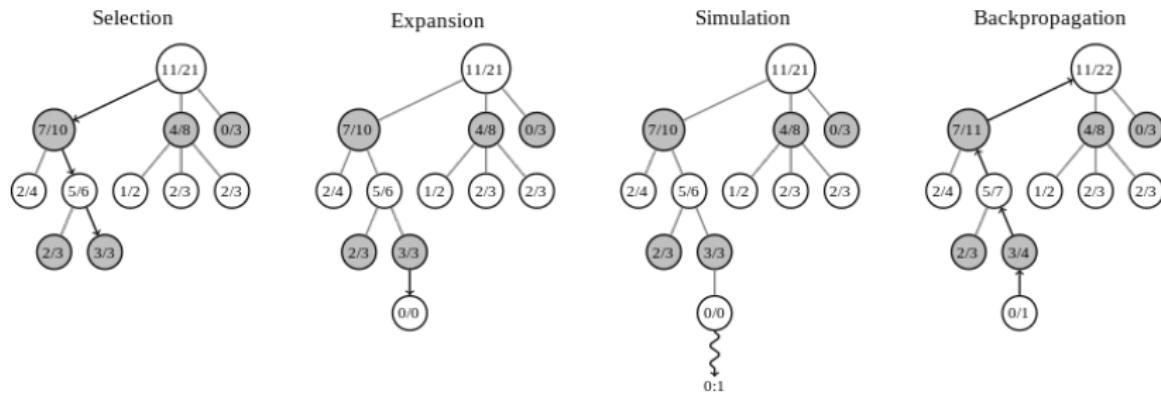
où

- ▶ $x_a \in \mathbb{R}^d$ est un vecteur (connu) de caractéristique du film
- ▶ $\theta \in \mathbb{R}^d$ (inconnu) indique les préférences de l'utilisateur

Bandits et intelligences artificielles pour les jeux

Pour décider du prochain coup à jouer:

- ▶ choisir successivement des trajectoires dans l'arbre de jeu
- ▶ effectuer des évaluations (aléatoires) de certains positions
- Comment sélectionner séquentiellement les trajectoires ?



Algorithme UCT [Kocsis & Szepesvari 06]: **UCB for Trees !**