





Introduction à l'apprentissage statistique

Cours de Mastère

Claude PETIT

14 octobre 2026

Table des matières

1 Une brève histoire de l'intelligence artificielle **7**
- 1.1 Les origines de l'intelligence artificielle . . . 7
- 1.2 Les débuts de l'IA symbolique : 1956-1970 . . . 8
- 1.3 Les désillusions et les hivers de l'IA : 1970-1990 . . . 9
- 1.4 Le tournant du machine learning statistique : 1990-2010 . . . 10
- 1.5 La révolution du deep learning : 2010-2016 . . . 11
- 1.6 L'ère des transformers et de l'IA générative : depuis 2017 . . . 11
- 1.7 Chronologie synthétique . . . 12

2 Introduction et problématique **15**
- 2.1 Définition de l'apprentissage statistique . . . 15
- 2.2 Le problème de la dimension . . . 16
  - 2.2.1 Le fléau de la dimension . . . 16
  - 2.2.2 La réduction de dimension . . . 20

3 Généralités sur l'apprentissage supervisé **21**
- 3.1 Théorie de la décision statistique . . . 22
  - 3.1.1 Classification et régression . . . 22
  - 3.1.2 Prédicteur de Bayes en classification binaire . . . 22
  - 3.1.3 Fonction oracle . . . 24
  - 3.1.4 Consistance d'un algorithme d'apprentissage . . . 25
- 3.2 Minimisation du risque empirique . . . 27
- 3.3 Sélection de modèles et validation croisée . . . 30

4 Régressions **33**
- 4.1 Rappels de base . . . 33
  - 4.1.1 Hypothèses du modèle . . . 33
  - 4.1.2 Le problème de la grande dimension . . . 35
  - 4.1.3 Validation du modèle et qualité de la régression . . . 36

5 Méthodes à base de partition **37**
- 5.1 Généralités sur les méthodes à partition . . . 37
- 5.2 La méthode des k plus proches voisins . . . 39
  - 5.2.1 Principe des k-ppv . . . 39
  - 5.2.2 Exemples en classification et régression . . . 40
  - 5.2.3 Remarques sur la méthode des k-ppv . . . 40
- 5.3 Les arbres de décision et de régression . . . 43
  - 5.3.1 La méthode CART : Classification And Regression Tree . . . 43
  - 5.3.2 Les différents critères de segmentation . . . 46
  - 5.3.3 Les différents critères d'élagage de CART. . . . 48
  - 5.3.4 Exemples en classification et régression. . . . 49
  - 5.3.5 Remarques finales. . . . 49

6 Méthodes à base de convexification **51**
- 6.1 Introduction et problématique . . . 51
  - 6.1.1 Convexification du dictionnaire . . . 51
  - 6.1.2 Convexification de la fonction perte . . . 52
  - 6.1.3 Consistance du minimiseur du ϕ-risque . . . 53

6.2 Espaces de Hilbert et méthodes à noyaux . . . 53
6.3 Machines à vecteurs de support (SVM) . . . 58
6.3.1 Machine à vecteurs de support : point de vue hilbertien . . . 58
6.3.2 Machine à vecteurs de supports : point de vue traditionnel . . . 58
6.3.3 Notion de marge souple de variables d'écart . . . 63
6.3.4 SVR : régression à vecteurs de support . . . 63
6.3.5 Conclusion et bibliographie sommaire . . . 65
6.4 Boosting . . . 69
6.4.1 Introduction et motivation . . . 69
6.4.2 Adaboost : Adaptive Boosting . . . 69
6.4.3 Gradient Boosting . . . 72
6.4.4 Boosting multi-classes : SAMME . . . 75
6.4.5 Extreme Gradient Boosting : XGBoost . . . 76
6.4.6 Boosting pour la régression : L^2-Boosting . . . 76
6.4.7 Conclusion et bibliographie sommaire . . . 77

7 Méthodes à base d'agrégation **79**
7.1 introduction et motivation . . . 79
7.2 Boostrap et rééchantillonnage . . . 79
7.3 Bagging : Bootstrap Agregating . . . 81
7.3.1 Principe du Bagging . . . 81
7.3.2 Forêts aléatoires . . . 82
7.3.3 Mesures de performances . . . 83
7.4 Le Boosting vu comme une méthode d'agrégation . . . 83
7.5 Stacking . . . 83
7.5.1 Principe du stacking . . . 83
7.6 Comparaison des méthodes . . . 84

Avant propos

Ce polycopié d'apprentissage statistique couvre un programme d'une trentaine d'heures. Le niveau requis est celui d'une licence ou d'une première année de master : les outils de base du calcul des probabilités sont supposés connus (théorie de la mesure, lois usuelles, fonctions caractéristiques, etc.), ainsi que les différentes notions de convergence des suites de variables aléatoires.

L'objectif du cours est de comprendre les aspects théoriques et pratiques des algorithmes de « Machine Learning » ; l'accent sera donc clairement mis sur la résolution d'exercices et sur l'implémentation des algorithmes en langage Python.

Le polycopié est un complément du cours, il ne le remplace absolument pas. Beaucoup d'exemples donnés durant les séances n'apparaissent pas dans ce document et à l'inverse beaucoup de démonstrations ou de paragraphes supplémentaires qui s'y trouvent ne seront pas traités en cours ; pour ces raisons, il est donc nécessaire de bien prendre le cours.

Le contenu du polycopié est très fortement inspiré des deux ouvrages suivants :

- Introduction à l'apprentissage statistique, de Frédéric Sur:
  https://members.loria.fr/FSur/enseignement/apprauto/poly_apprauto_FSur.pdf.
- Le polycopié du cours d'apprentissage et Data Mining d'Arnak Dalalyan:
  https://adalalyan.github.io/cours.html.

Il s'appuie également sur les ouvrages (classiques) suivants :

- Pattern Recognition, de C Bishop, éditions Springer, 2006:
  https://.
- The Elements of Statistical Learning, de Hastie, Tibshirani, Friedmann éditions Springer, 2008:
  https://adalalyan.github.io/cours.html.

G. Thomas,Mathematics for machine learning, Univ. of California at Berkeley, 2018. https://gwthomas.github.io/docs/math4ml.pdf

Les parties non traitées ou les démonstrations ne sont pas exigibles en examen, mais elles pourront intéresser ceux d'entre vous qui veulent aller plus loin.

Le site web du cours de maths contient des documents supplémentaires. La bibliothèque de l'école contient également un nombre important de livres de cours et d'exercices corrigés.

Chapitre 1

Une brève histoire de l'intelligence artificielle

 En cours de réalisation...

Introduction

Le terme même d'« intelligence artificielle » pose problème et suscite la controverse depuis sa première utilisation. Il n'est donc pas surprenant que sa définition soit elle aussi difficile à établir. De manière générale, l'intelligence artificielle, ou IA, désigne l'ensemble des techniques permettant à une machine d'accomplir des tâches habituellement associées à l'intelligence humaine : raisonner, reconnaître des images, comprendre le langage, jouer, prédire ou encore générer du contenu.

Nous reportons au chapitre suivant la définition du « machine learning » ou apprentissage automatique, qui est l'objet de ce cours.

On peut résumer l'histoire de l'IA en quatre grandes périodes :

- les années 1950-1970 : l'IA symbolique, fondée sur la logique et les règles ;
- les années 1980-2000 : le développement du machine learning statistique ;
- les années 2010 : la révolution du *deep learning* ;
- depuis 2017 : l'ère des *transformers*, des grands modèles de langage et de l'IA générative.

1.1 Les origines de l'intelligence artificielle

1940 : de la cybernétique de Wiener au neurone artificiel de McCulloch et Pitts

Avant même la naissance officielle de l'intelligence artificielle, plusieurs courants préparent le terrain. Dès les années 1940, Norbert Wiener, avec Julian Bigelow puis Arturo Rosenblueth, développe une réflexion sur les systèmes de contrôle, la rétroaction et le comportement finalisé des machines et des organismes. Leur article de 1943, *Behavior, Purpose and Teleology*, annonce la cybernétique, que Wiener formalisera en 1948 dans *Cybernetics: Or Control and Communication in the Animal and the Machine*.

En 1943, Warren McCulloch et Walter Pitts proposent un modèle mathématique simplifié du neurone biologique. Leur idée est de montrer qu'un neurone peut être représenté comme un dispositif logique capable de recevoir des signaux, de les combiner, puis de produire une sortie. Cette idée constitue l'une des premières bases théoriques des réseaux de neurones artificiels. Même si les ordinateurs de l'époque sont encore très limités, l'idée qu'une machine puisse imiter certains mécanismes du cerveau représente le fondement de cette science.

Quelques années après, Hebb propose une règle pour modifier les connections entre neurones et Minsky et Edmonds construisent le premier réseau de neurones.

1950 : Alan Turing et la question de l'intelligence des machines

En 1950, le mathématicien britannique Alan Turing publie un article célèbre intitulé *Computing Machinery and Intelligence*. Il y pose une question devenue fondamentale :

> Les machines peuvent-elles penser ?

Pour répondre à cette question, Turing propose le « jeu de l'imitation », aujourd'hui connu sous le nom de test de Turing (Turing's Option). Le principe est simple : si un humain dialogue avec une machine sans réussir à savoir s'il parle à une personne ou à un programme, alors la machine peut être considérée comme intelligente d'un certain point de vue. Le test de Turing ne définit pas parfaitement l'intelligence, mais il marque une étape importante : l'intelligence devient un objet que l'on peut tenter de mesurer à travers le comportement d'une machine.

1956 : la conférence de Dartmouth

L'année 1956 est souvent considérée comme la naissance officielle de l'intelligence artificielle comme domaine scientifique. Cette année-là, une conférence est organisée au Dartmouth College, aux États-Unis, par John McCarthy, Marvin Minsky, Nathaniel Rochester et Claude Shannon. C'est dans ce contexte que l'expression *artificial intelligence* est popularisée. Les chercheurs de Dartmouth pensent alors qu'il sera possible de décrire précisément les mécanismes de l'intelligence humaine pour les reproduire dans des machines. Cette ambition est très optimiste, mais elle lance un nouveau champ de recherche : Simon écrit « en moins de dix ans, un programme d'échecs arrivera au niveau d'un champion du monde et qu'un programme de démonstration automatique de théorème découvrira un théorème mathématique ».

C'est durant la conférence de Darmouth que le terme *intelligence artificielle* apparaît pour la première fois. La conférence dure deux mois.

1.2 Les débuts de l'IA symbolique : 1956-1970

Langage PL1 en 1956, Lisp en 1960 par MacCarthy et Prolog en 1971 par Alan Colmerauer. Eliza construit au MIT en 1965, système intelligent qui qui dialogue en anglais et joue au psychithérapeute. Simon prix Nobel d'économie en 1978. Logic Therorist (Newell Simon) résoud des puzzles simples.

Une IA fondée sur les règles

Après la conférence de Dartmouth, les premiers travaux en intelligence artificielle reposent principalement sur une approche symbolique.

L'idée est que l'intelligence peut être représentée par des symboles, des règles logiques et des raisonnements explicites. Un programme intelligent doit donc être capable de manipuler des symboles comme le ferait un humain lorsqu'il résout un problème.

Cette période voit apparaître des programmes capables de résoudre des problèmes mathématiques, de démontrer des théorèmes ou de jouer à certains jeux simples.

1958 : le perceptron de Frank Rosenblatt

En 1958, Frank Rosenblatt présente le **perceptron**. Il s'agit d'un modèle informatique inspiré du neurone biologique. Le perceptron peut apprendre à classer des données simples, par exemple distinguer deux catégories à partir d'exemples.

Le perceptron est important car il constitue l'un des premiers modèles d'apprentissage automatique. Cependant, ses capacités sont limitées. Il ne peut résoudre que des problèmes relativement simples, ce qui entraînera par la suite une perte d'intérêt temporaire pour les réseaux de neurones.

Les limites des premiers systèmes

Malgré l'enthousiasme des années 1950 et 1960, les premiers systèmes d'IA rencontrent rapidement plusieurs difficultés :

- les ordinateurs sont peu puissants ;
- les données disponibles sont limitées ;
- les programmes fonctionnent surtout dans des environnements très contrôlés ;

- il est difficile de représenter toute la complexité du monde réel avec des règles logiques.

Ces limites provoquent progressivement une baisse de confiance dans le domaine.

5 ans avaient été prévus pour un logiciel de traduction automatique, mais l'approche syntaxique était insuffisante (pour traduire un texte, il faut le comprendre). Annulation en 1966 de toute le financement américain pour les projet de TAL.

1.3 Les désillusions et les hivers de l'IA : 1970-1990

Les années 1970 : le premier hiver de l'IA

Dans les années 1970, l'intelligence artificielle traverse une période de déception. Les résultats ne correspondent pas aux promesses initiales. Les financements diminuent, notamment après le rapport Lighthill publié en 1973 au Royaume-Uni, qui critique les progrès réels de l'IA.

Cette période est souvent appelée le **premier hiver de l'IA**. Le terme désigne une baisse de l'intérêt scientifique, économique et politique pour l'intelligence artificielle.

Minsky et Papert en 1969 livre « perceptrons » les réseaux de neurones à une couche ne peuvent pas représenter certaines fonctions très simples.

Les années 1980 : les systèmes experts

Le premier système expert, appelé DENDRAL, fut créé en 1969 pour la tâche spécialisée consistant à déterminer la structure moléculaire d'une molécule étant données sa formule et les résultats de sa spectrométrie de masse. DENDRAL, comme toutes les systémes experts, est basé sur un grand nombre de règles heuristiques élaborées par des experts humains. Après le succès du DENDRAL, d'autres systèmes d'experts furent créés, notamment le système MYCIN, qui réalisait un diagnostic des infections sanguines. Avec 450 règles, MYCIN réussissait à diagnostiquer à un niveau proches des experts humains et considérablement meilleur que celui les jeunes médecins.

Dans les années 1980, l'IA connaît un regain d'intérêt grâce aux **systèmes experts**. Ces programmes sont conçus pour imiter le raisonnement d'un spécialiste humain dans un domaine précis.

Un système expert fonctionne généralement avec :

- une base de connaissances ;
- un ensemble de règles du type « si... alors... » ;
- un moteur d'inférence permettant d'appliquer ces règles.

Les systèmes experts sont utilisés dans des domaines comme la médecine, la chimie, l'industrie ou la finance. Ils montrent que l'IA peut avoir des applications pratiques, mais ils restent difficiles à maintenir. Dès que le domaine devient trop complexe ou change trop rapidement, il faut modifier manuellement les règles.

Les algorithmes génétiques simulent les processus de sélection naturelle des individus, les réseaux bayésiens se fondent sur la théorie des probabilités pour choisir, parmi plusieurs hypothèses, la plus vraisemblable. L'IA se focalise sur les théories et techniques permettant la réalisation d'intelligence individuelles ou collectives : IA distribuée, systèmes multiagents.

DEC configure des systèmes informatiques à l'aide d'un système expert.

1986 : la rétropropagation de l'erreur

En 1986, David Rumelhart, Geoffrey Hinton et Ronald Williams publient un article majeur sur la **rétropropagation de l'erreur**, ou *backpropagation*.

Cette méthode permet d'entraîner plus efficacement les réseaux de neurones. Le principe est de comparer la sortie produite par le réseau avec la sortie attendue, puis de corriger progressivement les paramètres internes du modèle pour réduire l'erreur.

La rétropropagation devient une technique centrale dans l'histoire des réseaux de neurones et prépare, plusieurs décennies plus tard, la révolution du deep learning.

Article ce qui est caché dans les couches cachées.

La fin des années 1980 : le deuxième hiver de l'IA

À la fin des années 1980 et au début des années 1990, les systèmes experts montrent leurs limites. Ils coûtent cher, sont fragiles et nécessitent beaucoup de maintenance humaine.

Cette nouvelle période de désillusion est appelée le **deuxième hiver de l'IA**. Pourtant, pendant que l'IA symbolique perd en popularité, une autre approche commence à se développer : le machine learning statistique.

1.4 Le tournant du machine learning statistique : 1990-2010

Changer d'approche : apprendre à partir des données

À partir des années 1990, les chercheurs s'intéressent davantage aux méthodes statistiques. Plutôt que d'essayer d'écrire toutes les règles à la main, on entraîne des modèles sur des données.

Cette évolution marque un tournant majeur. L'ordinateur ne se contente plus d'appliquer des règles : il apprend des régularités à partir d'exemples.

Le machine learning se développe alors dans plusieurs domaines :

- reconnaissance vocale ;
- classification d'images ;
- détection de fraude ;
- moteurs de recherche ;
- recommandations de produits ;
- analyse de données.

Les sous-domaines de l'IA : représentation des connaissances et raisonnement automatique, traitement du langage naturel, résolution de problèmes génériques, vision artificielle, robotique, apprentissage automatique.

1995 : les machines à vecteurs de support

En 1995, Corinna Cortes et Vladimir Vapnik publient un article important sur les **machines à vecteurs de support**, ou SVM.

Les SVM deviennent une méthode centrale du machine learning supervisé. Elles sont utilisées pour des tâches de classification, c'est-à-dire pour attribuer une catégorie à une donnée.

1997 : Deep Blue bat Garry Kasparov

En 1997, l'ordinateur Deep Blue, développé par IBM, bat le champion du monde d'échecs Garry Kasparov.

Cet événement marque fortement l'opinion publique. Pour la première fois, une machine bat le meilleur joueur humain dans un jeu considéré comme un symbole de l'intelligence.

Cependant, Deep Blue ne fonctionne pas comme les IA modernes fondées sur le deep learning. Il repose principalement sur la puissance de calcul, l'exploration de nombreuses positions possibles et des règles spécialisées écrites par des humains.

2001 : les forêts aléatoires

En 2001, Leo Breiman publie un article sur les **random forests**, ou forêts aléatoires. Cette méthode combine plusieurs arbres de décision pour améliorer la qualité des prédictions.

Les forêts aléatoires deviennent très populaires car elles sont efficaces, robustes et relativement faciles à utiliser. Elles illustrent bien la montée en puissance du machine learning statistique avant la domination du deep learning.

1.5 La révolution du deep learning : 2010-2016

Qu'est-ce que le deep learning ?

Le **deep learning**, ou apprentissage profond, est une branche du machine learning qui utilise des réseaux de neurones comportant de nombreuses couches.

Ces modèles peuvent apprendre automatiquement des représentations complexes. Par exemple, dans une image, les premières couches peuvent détecter des formes simples, tandis que les couches plus profondes peuvent reconnaître des objets, des visages ou des scènes.

Le succès du deep learning repose sur trois facteurs principaux :

- l'augmentation massive des données disponibles ;
- l'amélioration de la puissance de calcul, notamment grâce aux GPU ;
- le perfectionnement des algorithmes d'entraînement.

2012 : AlexNet et la compétition ImageNet

En 2012, Alex Krizhevsky, Ilya Sutskever et Geoffrey Hinton remportent la compétition ImageNet avec un réseau de neurones profond appelé **AlexNet**.

Cet événement est souvent considéré comme le début de la révolution moderne du deep learning. AlexNet obtient des résultats bien meilleurs que les méthodes précédentes en reconnaissance d'images.

À partir de ce moment, les réseaux de neurones profonds deviennent incontournables dans la vision par ordinateur.

2015 : reconnaissance du deep learning comme technologie centrale

En 2015, Yann LeCun, Yoshua Bengio et Geoffrey Hinton publient un article de synthèse montrant l'importance du deep learning dans plusieurs domaines :

- reconnaissance d'images ;
- reconnaissance vocale ;
- traduction automatique ;
- traitement du langage ;
- médecine ;
- génomique.

Le deep learning devient alors l'une des approches dominantes de l'intelligence artificielle.

2016 : AlphaGo bat Lee Sedol

En 2016, AlphaGo, développé par DeepMind, bat le champion sud-coréen Lee Sedol au jeu de go sur le score de quatre victoires à une.

Le jeu de go était considéré comme très difficile pour les machines en raison du nombre immense de positions possibles. La victoire d'AlphaGo est donc perçue comme une avancée majeure.

AlphaGo combine plusieurs techniques : réseaux de neurones profonds, apprentissage par renforcement et recherche arborescente. Cette victoire montre que l'IA peut réussir dans des domaines où l'intuition humaine semblait difficile à reproduire.

1.6 L'ère des transformers et de l'IA générative : depuis 2017

2017 : l'architecture Transformer

En 2017, des chercheurs de Google publient l'article *Attention Is All You Need*. Ils y présentent une nouvelle architecture appelée **Transformer**.

Cette architecture repose sur un mécanisme d'**attention**, qui permet au modèle d'identifier les éléments importants dans une séquence de données, par exemple dans une phrase.

Les transformers deviennent rapidement la base des grands modèles de langage modernes.

2018-2020 : les grands modèles de langage

Entre 2018 et 2020, des modèles comme BERT, GPT et leurs successeurs montrent qu'un système entraîné sur de très grands corpus de textes peut accomplir de nombreuses tâches :

- compléter une phrase ;
- traduire un texte ;
- résumer un document ;
- répondre à des questions ;
- générer du texte cohérent.

Ces modèles ne sont pas programmés pour une seule tâche précise. Ils apprennent des structures générales du langage à partir d'énormes quantités de données.

2022 : lancement public de ChatGPT

Le 30 novembre 2022, OpenAI lance ChatGPT. Ce système popularise massivement l'IA générative auprès du grand public.

ChatGPT montre qu'un modèle de langage peut dialoguer, expliquer, écrire, reformuler, traduire, aider à programmer ou produire des idées. Cette date marque une étape importante dans la diffusion sociale de l'intelligence artificielle.

2023-2025 : généralisation de l'IA générative

À partir de 2023, l'IA générative devient un enjeu économique, culturel et politique majeur. Les systèmes d'IA ne produisent plus seulement du texte : ils peuvent aussi générer des images, du code, du son, de la vidéo ou assister des tâches professionnelles.

Cette période soulève également de nombreuses questions :

- la fiabilité des réponses ;
- les biais algorithmiques ;
- la protection des données ;
- le droit d'auteur ;
- l'impact sur le travail ;
- la régulation de l'intelligence artificielle.

1.7 Chronologie synthétique

Date	Événement	Importance
1943	Modèle de neurone artificiel de McCulloch et Pitts	Première base théorique des réseaux de neurones
1950	Article d'Alan Turing sur les machines pensantes	Naissance du test de Turing
1956	Conférence de Dartmouth	Naissance officielle de l'intelligence artificielle comme domaine scientifique
1958	Perceptron de Frank Rosenblatt	Premier grand modèle d'apprentissage neuronal
1973	Rapport Lighthill	Début d'une période de désillusion et baisse des financements
1986	Rétropropagation de l'erreur	Relance des réseaux de neurones
1995	Machines à vecteurs de support	Méthode majeure du machine learning supervisé

1997	Deep Blue bat Garry Kasparov	Victoire symbolique d'une machine aux échecs
2001	Random Forests de Leo Breiman	Méthode importante d'apprentissage statistique
2012	AlexNet remporte ImageNet	Début de la révolution du deep learning
2016	AlphaGo bat Lee Sedol	Percée majeure de l'apprentissage profond et par renforcement
2017	Publication de *Attention Is All You Need*	Introduction de l'architecture Transformer
2022	Lancement public de ChatGPT	Popularisation massive de l'IA générative
2023-2025	Explosion des usages de l'IA générative	L'IA devient un enjeu économique, social et politique majeur

Conclusion

L'histoire de l'intelligence artificielle n'est pas une progression linéaire. Elle alterne entre enthousiasme, déceptions, périodes de ralentissement et avancées majeures.

Dans les années 1950, les chercheurs imaginent des machines capables de raisonner grâce à des règles logiques. Dans les années 1990, l'IA évolue vers des méthodes statistiques capables d'apprendre à partir de données. Dans les années 2010, le deep learning permet des progrès spectaculaires en vision, en langage et dans les jeux. Depuis 2017, les transformers et l'IA générative transforment profondément notre rapport aux machines.

La grande question n'est donc plus seulement :

Une machine peut-elle être intelligente ?

Elle devient plutôt :

Comment utiliser une intelligence artificielle puissante de manière fiable, utile et responsable ?

Bibliographie indicative

- Alan Turing, *Computing Machinery and Intelligence*, 1950.
- John McCarthy, Marvin Minsky, Nathaniel Rochester, Claude Shannon, conférence de Dartmouth, 1956.
- Frank Rosenblatt, travaux sur le perceptron, 1958.
- David Rumelhart, Geoffrey Hinton, Ronald Williams, travaux sur la rétropropagation, 1986.
- Corinna Cortes et Vladimir Vapnik, travaux sur les machines à vecteurs de support, 1995.
- IBM, victoire de Deep Blue contre Garry Kasparov, 1997.
- Leo Breiman, *Random Forests*, 2001.
- Alex Krizhevsky, Ilya Sutskever, Geoffrey Hinton, *ImageNet Classification with Deep Convolutional Neural Networks*, 2012.
- Yann LeCun, Yoshua Bengio, Geoffrey Hinton, *Deep Learning*, 2015.
- DeepMind, victoire d'AlphaGo contre Lee Sedol, 2016.
- Vaswani et al., *Attention Is All You Need*, 2017.
- OpenAI, lancement public de ChatGPT, 2022.

Chapitre 2

Introduction et problématique

2.1 Définition de l'apprentissage statistique

Selon Wikipedia, l'apprentissage statistique (apprentissage automatique, Machine Learning (ML)) est le champ d'études de l'intelligence artificielle (IA) qui se fonde sur des approches mathématiques et statistiques de l'IA pour donner aux ordinateurs la capacité d'apprendre à partir des données.

Selon ChatGPT ou Mistral AI, l'apprentissage statistique est une sous-discipline de l'IA qui permet aux ordinateurs d'apprendre à partir de données et de prendre des décisions sans être explicitement programmés pour accomplir ces tâches.

On peut considérer l'apprentissage comme une modification d'un comportement sur la base d'une expérience. Il s'agit d'imiter (un peu) le fonctionnement inductif du cerveau dans le but de prendre des décisions de façon autonome, en fonction des données disponibles. Arthur Samuel, pionnier de l'intelligence artificielle et grand informaticien, caractérise le ML par le fait que l'algorithme doit avoir la capacité d'apprendre sans que sa réponse ne soit explicitement programmée. Généralement, les décisions sont prises à partir de résultats de tests statistiques et d'intervalles de confiance.

En bref, l'apprentissage statistique, ce sont les mathématiques de l'IA.

Le ML est un domaine à l'intersection des mathématiques et de l'informatique, sollicitant fortement les statistiques. En quoi se distingue-t-il des statistiques traditionnelles ? Le développement des moyens informatiques amène à gérer des données plus nombreuses et plus complexes, pour lesquelles les méthodes traditionnelles des statistiques se révèlent peu efficaces. L'outil de base en statistique est le modèle tandis qu'en ML, ce sera l'algorithme. Un algorithme en ML apprend un modèle à partir d'exemples, par le biais d'un problème d'optimisation. Il prédit et calibre ses paramètres à partir de l'erreur de prévision.

Le Machine Learning n'est qu'une branche de l'intelligence artificielle. Nous avons évoqué d'autres facettes de l'IA dans le chapitre précédent. Historiquement l'IA était souvent découpée en apprentissage automatique, traitement du langage naturel, vision par ordinateur et robotique. Aujourd'hui, un découpage plus pertinent semble être donné par la typologie suivante:

- Apprentissage statistique: c'est l'approche connexionniste qui comprend le ML, les réseaux de neurones, l'apprentissage par renforcement, le traitement du langage naturel.
- Les systèmes formels: c'est l'approche cognitiviste qui comprend la programmation logique, les machines de Turing, la théorie de la calculabilité et celle des langages formels (Chomsky). Cette approche se concentre sur la modélisation des processus humains mentaux tels que la pensée, le raisonnement, la mémoire, les langages.
- Les méthodes faibles: ce sont des approches pragmatiques comprenant les méthodes heuristiques, les problèmes de satisfaction de contraintes, les systèmes experts ou les méthodes de représentations de connaissances. Cette approche se concentre sur la résolution pratique de problèmes concrets liés à l'intelligence, par des algorithmes, mais sans chercher à imiter le cerveau humain.

En statistique et en économie, un problème majeur est celui de la causalité. En ML, se pose de même le problème de l'interprétabilité: peut-on juste bien prédire sans avoir à comprendre comment le modèle effectue ses prédictions ? La réponse est clairement non et les deux aspects (prédiction et interprétabilité) sont indissociables: bien prédire implique bien expliquer.

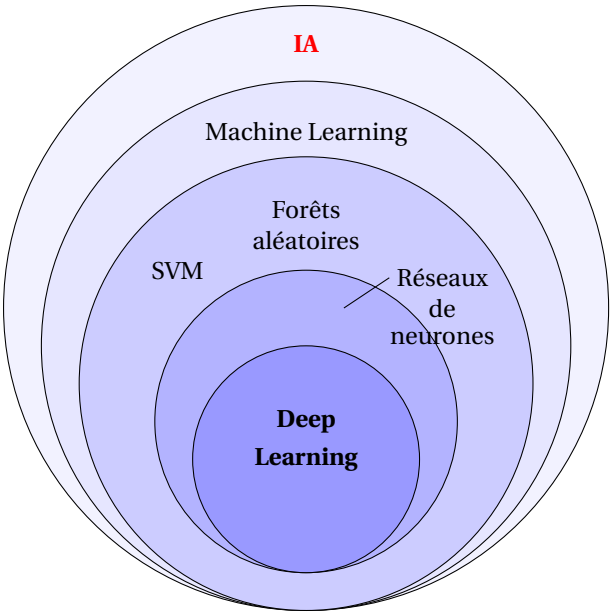


Figure 2.1 – Différents sous domaines de l'IA.

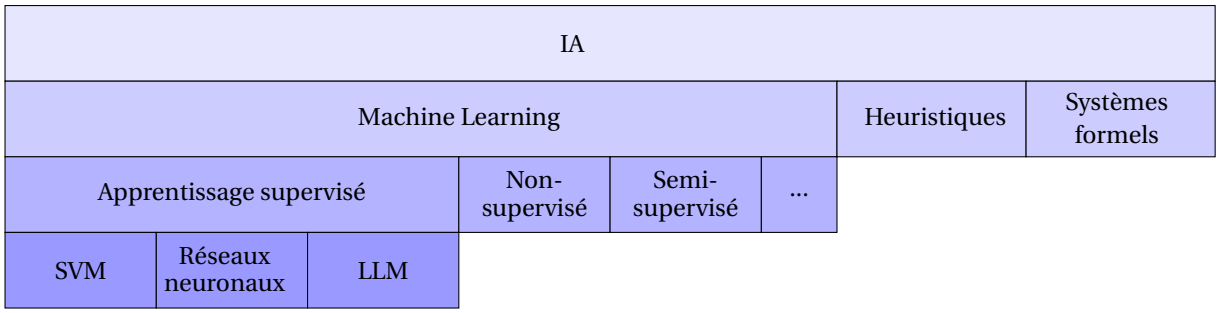


Figure 2.2 – Les différents sous domaines de l'IA.

Plusieurs approches sont possibles en ML : supervisée, non supervisée, semi-supervisée, en ligne, apprentissage par renforcement. L'apprentissage par renforcement fait l'objet d'un cours à part. Étant donné le volume horaire de 30 heures, nous aborderons uniquement les deux premières approches et ne donnerons que quelques exemples d'apprentissage non supervisé. L'essentiel du cours concerne donc l'apprentissage supervisé.

En apprentissage supervisé, les observations sont étiquetées et forment des couples (x_i, y_i) où les x_i sont les predicteurs ou « features » et les y_i les étiquettes ou labels. L'objectif est de prédire l'étiquette y pour un x donné. Ceci nécessite une phase d'apprentissage ou d'entrainement, puis une phase de test.

En apprentissage non supervisé, les données ne sont pas étiquetées. Lorsque la dimension d est assez grande, on peut chercher à caractériser la loi de probabilités ayant engendré ces observations en utilisant par exemple des méthodes de clustering ou d'estimation de densités de probabilités. Il existe aussi des méthodes de réduction de la dimension comme l'analyse en composantes principales (ACP), les méthodes de classifications hiérarchiques, les algorithmes de k-moyennes, l'estimation de densité non paramétrique, les mélanges de gaussiennes, etc.

En apprentissage semi-supervisé, seule une faible proportion des observations est étiquetée. Le but est le même qu'en apprentissage supervisé, mais les méthodes sont spécifiques.

2.2 Le problème de la dimension

2.2.1 Le fléau de la dimension

Ce terme (*curse of dimensionality*) recouvre différents problèmes relatifs aux propriérés des espaces de grande dimension, qui vont à l'encontre de l'intuition que l'on s'en fait en dimension 2 et 3. Il a été énoncé par Richard

Bellman dans les années 50-60 de la façon suivante : pour un choix de d décisions binaires, il existe 2^d configurations différentes, qui augmente de façon exponentielle avec d.

Voici une autre façon de le voir [Sur, 2024]: si le cube unité est discrétisé en petits cubes de côté $1/n$, il en faut n^d pour recouvrir le cube tout entier. Si l'on veut estimer une distribution de probabilités sur ce cube, en dimension $d = 1$, avec un échantillon de taille $n = 10$, on obtient une finesse de $1/10$ (en moyenne, un élément de l'échantillon se trouve dans chaque cube). En dimension $d = 10$, il faut un échantillon de taille 10^{10} pour couvrir en moyenne avec la même finesse, chaque partie.

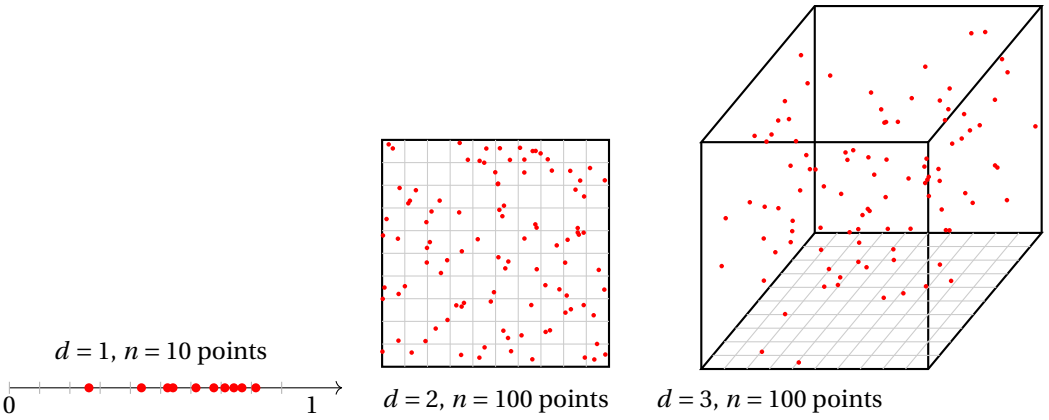


Figure 2.3 – Partition du cube unité et échantillonnage en dimension $d = 1, 2, 3$. Taille de grille = $1/10$. Le nombre de points nécessaires explose avec la dimension.

Le volume de la boule unité en dimension d est

$$V_d = \frac{\pi^{d/2}}{\Gamma(d/2+1)} \sim \frac{1}{\sqrt{\pi d}} \left(\frac{2\pi e}{d}\right)^{d/2} \tag{1}$$

qui tend vers 0 exponentiellement vite avec d. Γ est la fonction Gamma d'Euler définie pour $x > 0$ par:

$$\Gamma(x) = \int_0^{+\infty} t^{x-1} e^{-t}\, dt \tag{2}$$

et l'on a utilisé la formule de Stirling pour obtenir un équivalent en l'infini :

$$\Gamma(x) \sim \sqrt{2\pi} x^{x-1/2} e^{-x}. \tag{3}$$

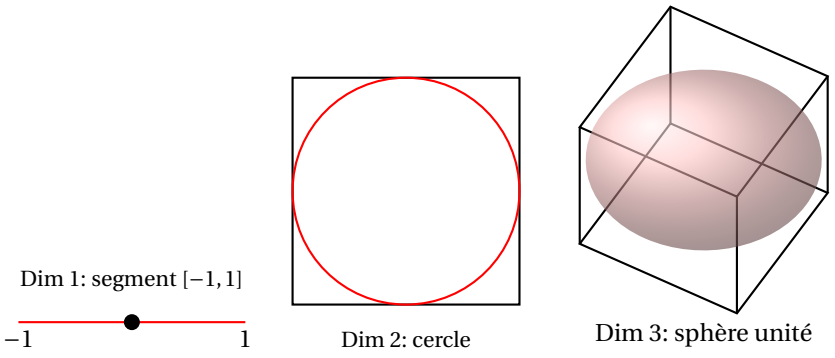


Figure 2.4 – Volume de la boule unité en dimension $d = 1, 2, 3$.

Considérons maintenant le cube de côté 2 circonscrit à la boule unité de $\mathbb{R}^d$. Son volume est 2^d. Le rapport du volume de la boule au cube est $V_d/2^d$ qui tend vers 0 encore plus vite que l'expression du volume. Des points répartis aléatoirement de manière uniforme sur le cube se retrouverons donc concentrés dans le volume extérieur à la boule unité (on parle de concentration dans les coins); surtout, si $0 < \epsilon < 1$, et $V_{d,1-\epsilon}$ est le volume de la boule de rayon $1-\epsilon$ en dimension d, alors on a

$$V_{d,1-\epsilon} = (1-\epsilon)^d V_d \tag{4}$$

et

$$\frac{V_d - V_{d,1-\epsilon}}{V_d} = 1 - (1-\epsilon)^d \tag{5}$$

de sorte que la proportion de la boule unité concentrée dans une couche d'épaisseur ϵ située à la surface de la boule, tend vers 1 quand d tend vers l'infini: le volume de la boule est concentré à sa surface.

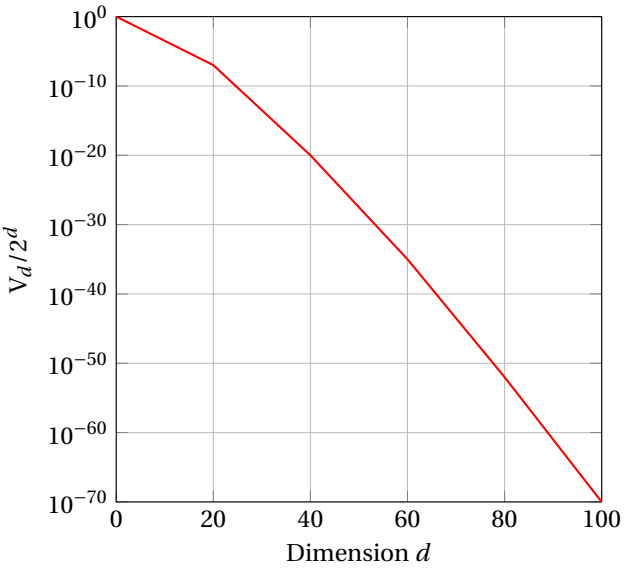


Figure 2.5 – Volume de la boule unité en dimension d rapporté au volume du cube de côté 2.

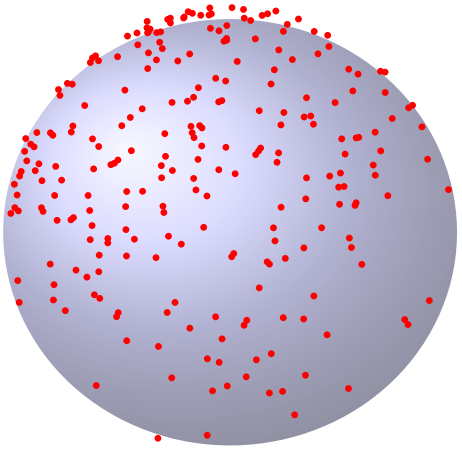


Figure 2.6 – Points aléatoires générés de façon uniforme dans l'espace, se concentrant à la surface de la sphère.

Ce phénomène peut également s'illustrer avec des distributions gaussiennes en grande dimension. En dimension $d = 1$, un échantillon de variables aléatoires gaussiennes standard se concentre autour de l'origine. C'est encore vrai pour les coordonnées d'un échantillon de vecteurs gaussiens standard pour une dimension d petite. Mais si d est grande, les points d'un échantillon gaussien standard vont se concentrer sur la surface de la sphère de rayon $\sqrt{d}$. En d'autres termes, la distribution normale standard en grande dimension est très proche de la distribution uniforme sur la sphère centrée en O et de rayon $\sqrt{d}$. C'est ce qu'illustre la figure 2.7.

Le phénomène de Hughes

Ce phénomène (heuristique) découvert et publié par Hugues en 1968 [Hughes, 1968], indique qu'il existe un nombre de variables caractéristiques (les « features ») optimal à ne pas dépasser. Il peut s'énoncer de la façon

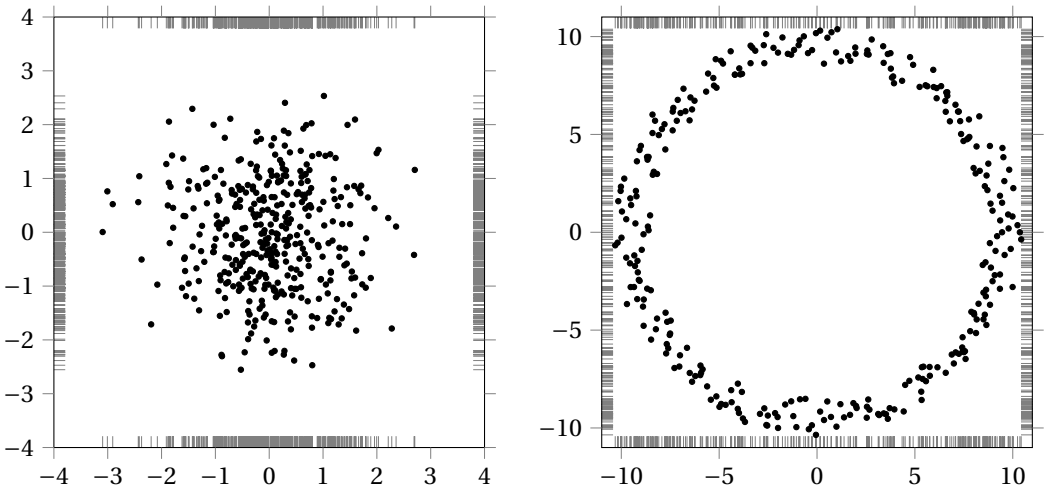


Figure 2.7 – Gauche : échantillon gaussien standard en dimension $d = 2$. Droit : Échantillon gaussien standard en dimension $d = 100$, projeté dans un sous-espace de dimension 2.

suivante : à base d'apprentissage de taille fixée, les performances d'un classifieur augmentent avec la dimension des observations, atteignent un maximum, puis diminuent. Autrement dit, le taux d'erreur d'un classifieur diminue avec la dimension des observations jusqu'à un minimum, puis augmente pour tendre vers 1/2 quand d tend vers l'infini : en très grande dimension, les classifieurs ne font donc pas mieux qu'un tirage à pile ou face.

Ce phénomène a intrigué les chercheurs pendant des années jusqu'à ce qu'en 1978, Van Campenhout ne montre que l'article contenait des défauts importants [Van Campenhout, 1978] : si l'on s'intéresse à l'erreur de classification en fonction de la complexité du problème, alors il faut considérer des ensembles de problèmes et donc définir une distribution de probabilités *a priori* sur cet ensemble. Hughes a choisi une distribution uniforme mais a utilisé des estimateurs non consistants sur son modèle. Avec un *a priori* uniforme, il aurait dû utiliser un estimateur de Bayes plutôt que l'estimateur du maximum de vraisemblance. Par ailleurs, l'hypothèse uniforme faite sur l'ensemble des problèmes n'est pas valable lorsque la dimension augmente. Ce phénomène, régulièrement observé dans les données réelles et lié à la malédiction de la dimension, demeure donc un paradoxe non expliqué. Nous reviendrons dessus lors de discussions concernant le dilemne biais-variance et le phénomène de double descente. On pourra se reporter au contenu du blog suivant pour plus de détails :
https://37steps.com/2322/hughes-phenomenon/.

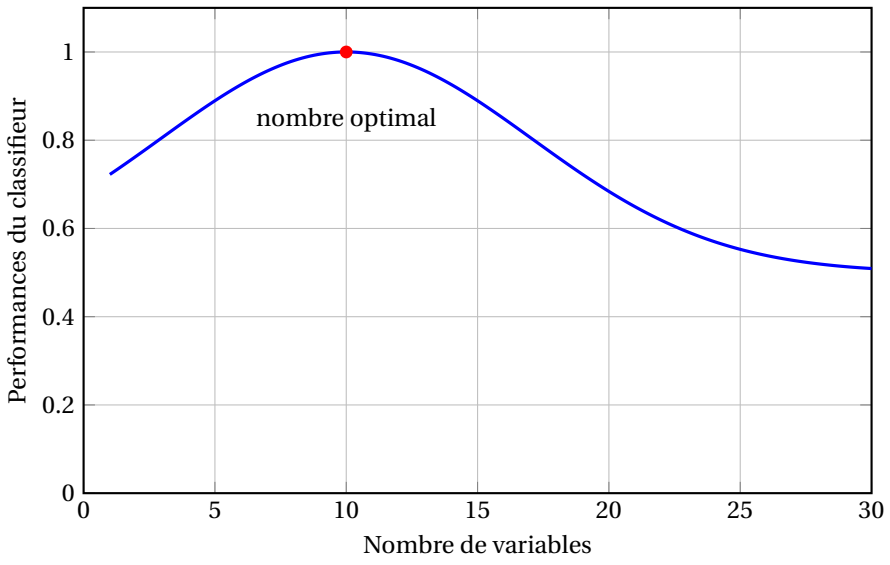


Figure 2.8 – Illustration du paradoxe de Hughes

2.2.2 La réduction de dimension

Comment faire pour contrer le fléau de la dimension ?

La réduction de dimension vise à transformer des données d'un espace de grande dimension en un espace de dimension inférieure tout en préservant des propriétés jugées essentielles des données d'origine. L'objectif est de rendre possible ou plus rapide le traitement de ces données, de réduire la complexité des processus les impliquant, d'économiser de l'espace de stockage, de l'énergie et de se prémunir contre le fléau de la dimension. Réduire la dimension peut également améliorer l'interprétabilité ou permettre la visualisation des données. On peut enfin considérer la réduction de dimension comme une forme de compression avec perte.

Les méthodes de réduction de dimension sont traditionnellement divisées en approches linéaires et non linéaires, mais d'autres axes de classification existent. On peut par exemple classer les méthodes selon qu'elles sont aléatoires ou déterministes, ou selon qu'elles s'appliquent à des modèles de taille finie ou asymptotiques, lorsque les valeurs des paramètres tendent vers l'infini sous des régimes spécifiques.

La classification qui nous intéresse ici découle du paradigme du Big Data (on dira données massives ou mégadonnées), où deux paramètres fondamentaux décrivent les dimensions des données : n, la taille de la population (nombre d'éléments de la base de données) et d, la dimension des variables statistiques attachées à ces éléments.

Trois situations sont possibles :

- n grand, d petit : c'est le domaine des statistiques multivariées traditionnelles (l'analyse de données « à la française »). Dans ce scénario, les outils d'inférence statistique classiques fonctionnent bien, en particulier les théorèmes limites classiques, lorsque n tend vers l'infini avec d fixé.
- n petit, d grand : c'est le domaine des statistiques en grande dimension, où les outils d'inférence statistique usuels ne fonctionnent plus, ni dans un cadre non asymptotique ni dans un cadre asymptotique. La matrice de covariance empirique est singulière, les estimateurs des moindres carrés ne sont pas consistants et peuvent donner de mauvais résultats, etc. Des hypothèses supplémentaires sont alors nécessaires pour traiter les données, comme une hypothèse de parcimonie, de structure sous-jacente cachée ayant une petite dimension, etc.
- n et d grands : c'est un autre aspect des statistiques en grande dimension, et typiquement le domaine de la théorie des matrices aléatoires. Dans un cadre asymptotique, aucun théorème limite classique ne s'applique, et des hypothèses sur la limite de n/d doivent être faites lorsque n et d tendent vers l'infini, pour appliquer des théorèmes spécifiques, comme la convergence vers la loi de Marchenko-Pastur.

À côté des deux paramètres fondamentaux, il peut exister d'autres paramètres pour décrire les données, par exemple si l'espace dans lequel elles vivent n'est pas euclidien. De la taille ou de la complexité de la structure hébergeant les données peuvent alors émerger un ou plusieurs paramètres décrivant cette structure. C'est le cas, par exemple, si les données se trouvent sur un graphe. Le nombre d'arêtes m est alors un troisième paramètre à prendre en compte. m peut être de l'ordre de n^2 si le graphe est dense, ou de l'ordre de n si le graphe est parcimonieux.

En résumé, pour réduire la dimension, on peut donc compresser les données, trouver un sous-espace de dimension plus petite qui contient (presque) toute l'information ou simplifier l'espace sous-jacent aux données.

Chapitre 3

Généralités sur l'apprentissage supervisé

Introduction

Comme nous l'avons dit dans l'avant-propos, l'apprentissage supervisé considère des données étiquetées et l'objectif est de prédire l'étiquette associée à une nouvelle observation à partir des observations étiquetées du jeu de données initial.

Toutes les méthodes considèrent une fonction de prédiction g_θ appartenant à une famille paramétrée par $\theta \in \Theta$, où Θ représente l'ensemble des paramètres possibles. La prédiction consiste alors à choisir au mieux la valeur du paramètre. Lorsqu'on se place dans un contexte probabiliste, l'apprentissage supervisé a donc le même objectif que la statistique mathématique : estimer au mieux la valeur du paramètre.

Dans une première phase d'apprentissage (ou d'entraînement), les paramètres sont estimés de manière à optimiser les performances de prédiction sur le jeu de données initial, appelé base d'apprentissage ou d'entraînement. L'estimation est basée sur la minimisation de l'écart entre les vraies étiquettes et les étiquettes prédites. Mais il ne suffit pas de choisir θ minimisant cet écart pour obtenir des performances de prédiction optimales.

Il y a donc deux composantes fondamentales dans toute méthode d'apprentissage supervisé:

- Les données d'apprentissage (qui comprennent les observations et leurs étiquettes) à partir duquel l'algorithme va « apprendre ».

- L'algorithme d'apprentissage qui va utiliser les données pour calibrer le modèle d'apprentissage statistique sous-jacent à la méthode.

Pourquoi un formalisme probabiliste ? Les données du monde réel sont souvent bruitées, incertaines et comportent des erreurs. Dans ce contexte, un formalisme probabiliste fournit un cadre naturel pour modéliser et quantifier les différents types d'incertitude qu'ils soient dus au bruit, aux erreurs ou aux facteurs non observables. Ce formalisme permet par ailleurs l'utilisation de modèles bayésiens qui rendent compte de l'information *a priori* et que l'on peut actualiser en fonction des connaissances acquises en cours de traitement.

Les phénomènes que l'on modélise sont influencés par de multiples facteurs, qui peuvent être corrélés ; un modèle probabiliste intégre ces dépendances entre variables. Il permet de prendre des décisions optimales en présence d'incertitude, sait gérer l'imputation de données manquantes ou aberrantes et fournit un cadre théorique unifié pour les différentes techniques d'apprentissage.

Mais la meilleure justification du formalisme probabiliste en IA est sans doute la suivante : les neuroscientifiques ont montré que le cerveau agissait de façon probabiliste en matière de perception du monde réel, de prise de décision ou d'apprentissage. Lorsque nous apprenons par l'exemple, dès la naissance, notre cerveau réagit en effectuant des inférences bayésiennes basées sur les informations sensorielles qu'il reçoit et sur les connaissances qu'il possède déjà [Rolls and Deco, 2010]. Il a été démontré que les neurones encodent des distributions de probabilités [Girardon et al., 2020].

Dans ce chapitre, nous définissons la notion de risque moyen et risque empirique. La minimisation du risque empirique est l'une des règles d'apprentissage les plus classiques. Dans ce cours, nous en étudierons essentiellement deux: minimisation du risque empirique et moyennage.

3.1 Théorie de la décision statistique

3.1.1 Classification et régression

On considère un n-échantillon $\mathcal{D}_n = \{Z_1, ..., Z_n\}$ de v.a.i.i.d. $Z_i = (X_i, Y_i)$. Les X_i sont des observations issues d'une v.a. X, ce sont les données que l'on souhaite classer et qui formeront les variables explicatives. Les Y_i sont issues d'une v.a. Y et sont les catégories auxquelles appartiennent les X_i (on dit également étiquettes ou labels). L'objectif de l'apprentissage supervisé est de déterminer au mieux la catégorie Y à laquelle appartient la donnée X correspondante, à partir des seules observations de l'échantillon $Z_1, ...Z_n$.

On suppose que les v.a. X sont issues d'un espace $\mathbb{X}$, que les v.a. Y sont issues d'un espace $\mathbb{Y}$ et l'on se donne une loi de probabilité (inconnue) $\mathbb{P}$ sur l'espace $\mathscr{E} = \mathbb{X} \times \mathbb{Y}$. $\mathbb{P}$ est la loi de (X,Y) et également la loi jointe commune des (X_i, Y_i).

Les étiquettes peuvent provenir d'un ensemble discret ou continu. Lorsque $\mathbb{Y} = \{0, 1\}$, on parle de classification binaire. Dans le cadre de la régression, $\mathbb{Y} = \mathbb{R}$ ou $\mathbb{R}^d$.

Une fonction de prédiction est un élément $g \in \mathcal{F} = \mathcal{F}(\mathbb{X}, \mathbb{Y})$ qui associe une étiquette à une observation. g s'appelle aussi prédicteur ou règle de prédiction. Pour mesurer la qualité de g, on définit différentes fonctions de perte $l : \mathbb{Y}^2 \longrightarrow \mathbb{R}_+$ telles que $l(Y, g(X))$ mesure l'écart entre la vraie valeur Y correspondant à X et la valeur $g(X)$ prédite à partir de la fonction g. La notation l vient de « loss ».

Lorsque la fonction perte utilisée est

$$l(Y, Y') = \mathbb{1}_{[Y \neq Y']}, \tag{1}$$

on parle de problème de classification et lorsque

$$l(Y, Y') = ||Y - Y'||^p, \tag{2}$$

on parle de problème de régression. Si $p = 2$, il s'agit de régression au sens des moindres carrés.

Le risque théorique $R(g)$ de g est la valeur moyenne des réalisations de toutes les pertes possibles. Autrement dit,

$$R(g) = R_{\mathbb{P}}(g) = \mathbb{E}\left[l(Y, g(X))\right] = \int_{\mathbb{X} \times \mathbb{Y}} l(y, f(x)) \mathbb{P}(dx, dy) \tag{3}$$

Étant donné que la loi de (X,Y) est inconnue (l'espérance est donnée sous cette loi), on ne peut pas calculer directement $R(g)$ et on ne peut que l'estimer à partir des observations de l'échantillon. $R(g)$ mesure la qualité de la prédiction de la fonction g. $R(g)$ s'appelle également erreur de prévision ou erreur de généralisation. Si g est une fonction déterministe, $R(g)$ est un nombre, mais si g est estimée à partir d'un échantillon de données, alors g est aléatoire et $R(g)$ est une variable aléatoire dépendant de l'échantillon.

3.1.2 Prédicteur de Bayes en classification binaire

Le prédicteur de Bayes est l'élément $g^\star$ de $\mathcal{F}$ qui minimise la perte $R(g)$ (on parle aussi de prédicteur oracle). C'est donc la fonction de prédiction optimale sachant les observations.

Dans ce paragraphe, nous nous limitons au problème de classification binaire, c'est à dire que Y ne peut prendre que deux valeurs: 0 ou 1. La fonction de perte naturelle associée est alors la fonction

$$l(Y, Y') = \mathbb{1}_{[Y \neq Y']} \tag{4}$$

Théorème 1

On note

$$\eta(x) = \mathbb{P}[Y = 1 | X = x] = \mathbb{E}[Y | X = x] \tag{5}$$

et

$$g^\star(x) = \mathbb{1}_{[\eta(x) > 1/2]} \tag{6}$$

Alors $g^\star$ minimise l'erreur de classification binaire.

Proof.

$$\begin{aligned}\mathbb{P}[\mathrm{Y}=1|\mathrm{X}=x] &= \mathbb{P}[\mathrm{Y}=g(\mathrm{X})|\mathrm{X}=x]\mathbb{1}_{[g(x)=1]} + \mathbb{P}[\mathrm{Y}=0|\mathrm{X}=x]\mathbb{1}_{[g(x)=0]} \\ &= \eta(x)\mathbb{1}_{[g(x)=1]} + (1-\eta(x))\mathbb{1}_{[g(x)=0]},\end{aligned}$$

donc

$$\mathbb{P}[\mathrm{Y}=g(\mathrm{X})|\mathrm{X}=x] = \eta(x)\mathbb{1}_{[g(x)=1]} + (1-\eta(x))\mathbb{1}_{[g(x)=0]}.$$

L'erreur de prédiction est:

$$\begin{aligned}\mathbb{P}[g(\mathrm{X})\neq\mathrm{Y}|\mathrm{X}=x] &= 1-\mathbb{P}[g(\mathrm{X})=\mathrm{Y}|\mathrm{X}=x] \\ &= 1-\mathbb{P}[\mathrm{Y}=1,g(\mathrm{X})=1|\mathrm{X}=x]-\mathbb{P}[\mathrm{Y}=0,g(\mathrm{X})=0|\mathrm{X}=x] \\ &= 1-\mathbb{1}_{[g(x)=1]}\mathbb{P}[\mathrm{Y}=1|\mathrm{X}=x]-\mathbb{1}_{[g(x)=0]}\mathbb{P}[\mathrm{Y}=0|\mathrm{X}=x] \\ &= 1-\mathbb{1}_{[g(x)=1]}\eta(x)-\mathbb{1}_{[g(x)=0]}(1-\eta(x)).\end{aligned}$$

Ainsi,

$$\begin{aligned}\mathbb{P}[g(\mathrm{X})\neq\mathrm{Y}|\mathrm{X}=x]-\mathbb{P}[g^\star(\mathrm{X})\neq\mathrm{Y}|\mathrm{X}=x] &= \eta(x)\left(\mathbb{1}_{[g^\star(x)=1]}-\mathbb{1}_{[g(x)=1]}\right)+(1-\eta(x))\left(\mathbb{1}_{[g^\star(x)=0]}-\mathbb{1}_{[g(x)=0]}\right) \\ &= (2\eta(x)-1)\left(\mathbb{1}_{[g^\star(x)=1]}-\mathbb{1}_{[g(x)=1]}\right) \\ &\geq 0,\end{aligned}$$

par définition de $g^\star(x)$ et parce que $2\eta(x)-1>0 \iff g^\star(x)=1$. On en déduit l'inégalité :

$$\mathbb{P}[\mathrm{Y}\neq g^\star(\mathrm{X})|\mathrm{X}=x] \leq \mathbb{P}[\mathrm{Y}\neq g(\mathrm{X})|\mathrm{X}=x], \tag{7}$$

licite pour toute fonction g. Le risque optimal est:

$$\mathrm{R}^\star = \mathrm{R}(g^\star) = \mathbb{E}[\mathbb{1}_{[g^\star(x)=1]}] = \mathbb{P}[g^\star(\mathrm{X})\neq\mathrm{Y}] = \mathbb{E}\left[\left(g^\star(\mathrm{X})-\mathrm{Y}\right)^2\right].$$

En effet, comme Y et $g(\mathrm{X})$ valent 0 ou 1,

$$\mathbb{1}_{[g(x)\neq\mathrm{Y}]} = \left(g(\mathrm{X})-\mathrm{Y}\right)^2 = \mathrm{Y}^2+g(\mathrm{X})^2-2\mathrm{Y}g(\mathrm{X}) = \mathrm{Y}+g(\mathrm{X})(1-2\mathrm{Y}).$$

Ainsi,

$$\begin{aligned}\mathrm{R}(g) &= \mathbb{E}[\mathrm{Y}]+\mathbb{E}[g(\mathrm{X})(1-2\mathrm{Y})] = \mathbb{E}[\mathbb{E}[\mathrm{Y}|\mathrm{X}]]+\mathbb{E}[g(\mathrm{X})(1-2\mathbb{E}[\mathrm{Y}|\mathrm{X}])] \\ &= \mathbb{E}[\eta(\mathrm{X})]+\mathbb{E}[g(\mathrm{X})(1-2\eta(\mathrm{X}))]\end{aligned}$$

Maintenant, si $g=g^\star$,

$$\mathrm{R}^\star = \mathbb{E}[\eta(\mathrm{X})]+\mathbb{E}[\mathbb{1}_{[\eta(\mathrm{X})>1/2]}(1-2\eta(\mathrm{X}))]$$

Si $\eta(\mathrm{X})>1/2$ l'expression devient

$$\mathrm{R}^\star = \mathbb{E}\left[\eta(\mathrm{X})(1-2\mathbb{1}_{[\eta(\mathrm{X})>1/2]})+\mathbb{1}_{[\eta(\mathrm{X})>1/2]}\right] = \mathbb{E}[1-\eta(\mathrm{X})]$$

et de même, si $\eta(\mathrm{X})<1/2$,

$$\mathrm{R}^\star = \mathbb{E}\left[\eta(\mathrm{X})\right]$$

et donc

$$\mathrm{R}^\star = \mathbb{E}[\eta(\mathrm{X})\wedge(1-\eta(\mathrm{X}))].$$

on rappelle que

$$a\wedge b = (a+b-|a-b|)/2,$$

de sorte que

$$\mathrm{R}^\star = \mathbb{E}\left[\frac{1}{2}\left(1-|2\eta(\mathrm{X})-1|\right)\right] = \frac{1}{2}\left(1-\mathbb{E}\left[|2\eta(\mathrm{X})-1|\right]\right),$$

avec $x \wedge y = \inf(x, y)$. Ainsi, le risque de Bayes $R^\star = R(g^\star)$ vérifie

$$R^\star = \mathbb{E}[\eta(X) \wedge (1 - \eta(X))] = \frac{1}{2}\left(1 - \mathbb{E}[|2\eta(X) - 1|]\right), \tag{8}$$

De manière plus générale,

$$\begin{aligned}\mathbb{E}\left[(f(X) - Y)^2 | X = x\right] &= \mathbb{E}\left[(f(x) - \eta(x) + \eta(x) - Y)^2 | X = x\right] \\ &= (f(x) - \eta(x))^2 + 2(f(x) - \eta(x))\mathbb{E}[\eta(x) - Y | X = x] + \mathbb{E}[(\eta(X) - Y)^2 | X = x] \\ &= (f(x) - \eta(x))^2 + \mathbb{E}[(\eta(X) - Y)^2 | X = x]\end{aligned}$$

en passant à l'espérance. Cette expression est positive et l'on en déduit que

$$\mathbb{E}\left[(\eta(X) - Y)^2\right] \leq \mathbb{E}\left[(f(X) - Y)^2\right].$$

Quelque soit la fonction f de $\mathbb{X}$ dans $\mathbb{R}$, $\eta(X)$ minimise donc bien l'erreur quadratique lorsque $f(X)$ prédit Y. □

3.1.3 Fonction oracle

De façon plus générale, que ce soit en classification ou régression, toute fonction $g^\star$, si elle existe, minimisant le risque de prédiction est appelée fonction oracle. Le terme oracle provient du fait que la loi étant inconnue, $g^\star$ est également inconnue. Cette fonction se calcule à l'aide de l'espérance conditionnelle sachant les observsations.

Théorème 2

Si pour tout $x \in \mathbb{X}$ la borne inférieure sur $y \in \mathbb{Y}$ de $\mathbb{E}[l(Y, y)|X = x]$ est atteinte, alors toute fonction $g^\star : \mathbb{X} \longrightarrow \mathbb{Y}$ (avec $y = g^\star(x)$) minimisant cette espérance conditionnelle est une fonction oracle:

$$\forall x \in \mathbb{X}, g^\star(x) \in \underset{y \in \mathbb{Y}}{\arg\min}\, \mathbb{E}[l(Y, y)|X = x] \Rightarrow g^\star \in \underset{g \in \mathcal{F}}{\arg\min}\, R(g). \tag{9}$$

Proof. Par définition,

$$g^\star \in \underset{g \in \mathcal{F}}{\arg\min}\, R(g) \tag{10}$$

Soit $g_m \in \arg\min_{y \in \mathbb{Y}} \mathbb{E}[l(Y, y)|X = x]$. Pour toute fonction $g \in \mathcal{F}$,

$$\mathbb{E}[l(Y, g(x))|X = x] \geq \mathbb{E}[l(Y, g_m(x))|X = x]. \tag{11}$$

On a alors

$$\begin{aligned}R(g) &= \mathbb{E}[l(Y, g(X)] = \mathbb{E}[\mathbb{E}[l(Y, g(x))|X = x]] &(12)\\ &\geq \mathbb{E}[\mathbb{E}[l(Y, g_m(x))|X = x]] = \mathbb{E}[l(Y, g_m(X)] = R(g_m). &(13)\end{aligned}$$

Ainsi, $g_m \in \arg\min_g R(g)$ et l'on peut poser $g^\star = g_m$.

□

Théorème 3

En régression au sens des moindres carrées, la fonction oracle est donnée par

$$\eta^\star(x) = \mathbb{E}[Y|X = x] \tag{14}$$

et vérifie

$$\forall \eta : \mathbb{X} \longrightarrow \mathbb{R},\ R(\eta) = R(\eta^\star) + \mathbb{E}\left[(\eta(X) - \eta^\star(X))^2\right] \tag{15}$$

Proof. Soit $\mathbb{Y} = \mathbb{R}$ et $l(y, y') = (y - y')^2$ (cas de la régression au sens des moindres carrés). D'après le théorème précédent,

$$g^\star(x) \in \underset{y\in\mathbb{R}}{\arg\min}\, \mathbb{E}[(\mathrm{Y} - y)^2 | \mathrm{X} = x]. \tag{16}$$

Mais

$$\mathbb{E}[(\mathrm{Y} - y)^2 | \mathrm{X} = x] = \mathbb{E}[\mathrm{Y}^2 | \mathrm{X} = x] - 2y\mathbb{E}[\mathrm{Y} | \mathrm{X} = x] + y^2. \tag{17}$$

Il s'agit d'un polynôme du second degré en y dont le minimum est atteint en $y_0 = \mathbb{E}[\mathrm{Y}|\mathrm{X} = x] = \eta^\star(x)$.

□

Le prédicteur de Bayes n'existe pas toujours et lorsqu'il existe, il n'est pas forcément unique (c'est la raison pour laquelle nous avons utilisé la notation $\in$ au lieu de $=$ dans l'avant dernier théorème).

On notera l'excès de risque la quantité (positive) $\mathrm{R}(g) - \mathrm{R}(g^\star)$ qui mesure également la qualité du prédicteur et s'annule lorsque g est égal au prédicteur optimal.

3.1.4 Consistance d'un algorithme d'apprentissage

Un algorithme d'apprentissage est une fonction

$$g : \bigcup_{n=1}^{+\infty} (\mathbb{X} \times \mathbb{Y})^n \longrightarrow \mathcal{F}(\mathbb{X}, \mathbb{Y}) \tag{18}$$

qui a tout ensemble d'apprentissage $\mathscr{D}_n$ associe une fonction de prédiction g_n (n n'est pas fixé et peut varier). C'est en quelque sorte un estimateur de la meilleure fonction de prédiction. On écrira par abus de notation $g_n(x) = g_n(x, \mathscr{D}_n)$ sans toujours indiquer la dépendance à l'échantillon.

$$g_n = g_n(., \mathscr{D}_n) \in \mathcal{F}(\mathbb{X}, \mathbb{Y}) \tag{19}$$

Comme $\mathscr{D}_n$ est fonction des $(\mathrm{X}_i, \mathrm{Y}_i)$, il est aléatoire et par conséquent $g_n(x) = g_n(x, \mathscr{D}_n)$ est également aléatoire et il en est de même de

$$\mathrm{R}_{\mathbb{P}}\left(g_n\right) = \int_{\mathbb{X}\times\mathbb{Y}} l(y, g_n(x))\mathbb{P}(dx, dy) \tag{20}$$

En fait, l'expression $g_n(\mathrm{X}, \mathscr{D}_n)$ est doublement aléatoire. Elle l'est au travers de la loi de (X, Y) à cause de la variable X, mais également au travers de l'échantillon $\mathscr{D}_n$ qui intervient dans la construction de g_n. Pour Calculer $\mathrm{R}_{\mathbb{P}}\left(g_n\right)$ on intègre sur X et le résultat est donc une variable aléatoire fonction de $\mathscr{D}_n$.

Notons alors $\mathbb{E}_{\mathbb{P}}[\mathrm{R}_{\mathbb{P}}\left(g_n\right)]$ (ou $\mathbb{E}[\mathrm{R}\left(g_n\right)]$ pour simplifier) l'espérance de cette variable aléatoire. Cette espérance est calculée par rapport au couple de variables $(\mathrm{X}, \mathrm{Y}) = (x, y)$ présent dans l'expression de $l(x, y)$. C'est le risque de prédiction ou risque moyen, l'espérance étant prise par rapport à la loi de l'échantillon.

On dit qu'un algorithme est consistant par rapport à une loi $\mathbb{P}$ si l'estimateur du risque tend, lorsque n tend vers l'infini, vers le risque de Bayes associé :

$$\lim_{n\to+\infty} \mathbb{E}\left[\mathrm{R}\left(g_n\right)\right] = \mathrm{R}(g^\star) \tag{21}$$

On dit qu'un algorithme est universellement consistant s'il est consistant par rapport à toute loi de probabilité $\mathbb{P}$ de l'ensemble $\mathscr{P}$ des mesures de probabilités sur l'espace sous-jacent:

$$\sup_{\mathbb{P}\in\mathscr{P}} \lim_{n\to+\infty} \left(\mathbb{E}\left[\mathrm{R}\left(g_n\right)\right] - \mathrm{R}(g^\star)\right) = 0 \tag{22}$$

Il est universellement et uniformément consistant s'il est uniformément consistant par rapport à toute loi $\mathbb{P}$, c'est à dire si, et seulement si,

$$\lim_{n\to+\infty} \sup_{\mathbb{P}\in\mathscr{P}} \left(\mathbb{E}\left[\mathrm{R}\left(g_n\right)\right] - \mathrm{R}(g^\star)\right) = 0 \tag{23}$$

Un théorème important en apprentissage supervisé, appelé « no free lunch » [Wolpert and Macready, 1997], démontre que dès que $\mathrm{card}(\mathbb{X}) = +\infty$, il n'existe pas d'algorithme d'apprentissage uniformément et universellement consistant. L'objectif de l'apprentissage est alors de construire un algorithme permettant d'avoir une consistance universelle sur une classe de probabilités pertinente pour le problème posé et pour une famille de fonctions de prédiction $\mathcal{G} \subset \mathcal{F}$ suffisamment grande.

Le sous-ensemble $\mathcal{G}$ de fonctions auquel on se restreint s'appelle le modèle ou le dictionnaire associé à l'estimateur (prédicteur ou régresseur).

$\mathscr{P}$ étant maintenant un sous-ensemble de mesures de probabilités inclus dans l'espace de toutes les mesures de probabilités, et l'échantillon d'apprentissage étant donné, nous cherchons un algorithme d'apprentissage (sous la forme d'une fonction de prédiction g_n) de telle sorte que

$$\sup_{\mathbb{P}\in\mathscr{P}} \lim_{n\to+\infty} \left(\mathbb{E}\left[\mathrm{R}\left(g_n\right)\right] - \mathrm{R}(g^\star)\right) = 0 \tag{24}$$

et cette quantité doit décroître suffisamment vite vers 0 pour que peu de données soient nécessaires à l'algorithme pour prédire efficacement. L'ensemble $\mathscr{P}$ modélise alors notre *a priori* (comme dans un cadre bayésien) et entraîne un *a priori* sur la fonction cible.

Rappelons que la fonction de perte associée à une fonction de prédiction est $\mathrm{R}_{\mathbb{P}}[l(\mathrm{Y}, g(\mathrm{X})]$ et que, comme $\mathbb{P} = \mathbb{P}_{(\mathrm{X},\mathrm{Y})}$ est inconnue, la fonction de perte l'est également. L'algorithme d'apprentissage va tenter de trouver la fonction de prédiction dont le risque est aussi petit que possible (aussi proche que possible du risque des fonctions oracles).

Puisque la loi $\mathbb{P}$ de (X,Y) est inconnue, le risque $\mathrm{R}(g)$ d'un prédicteur est inconnu également. Mais on peut l'estimer à partir de l'estimateur plug-in de la moyenne empirique :

$$\mathrm{R}_n(g) = \widehat{\mathrm{R}}_n(g) = \frac{1}{n}\sum_{i=1}^{n} l(\mathrm{Y}_i, g(\mathrm{X}_i)) \tag{25}$$

On note souvent ce risque avec un chapeau pour indiquer le caractère empirique ; nous ne le ferons pas pour alléger les notations. Il est important de noter que le risque empirique est défini sur un dictionnaire (une famille donnée de fonctions de prédiction, sous-ensemble $\mathcal{G}$ de $\mathcal{F}$). Sous réserve que $\mathbb{E}[l(\mathrm{Y}, g(\mathrm{X}))^2] < \infty$, la loi des grands nombres et le théorème de la limite centrale assurent que

$$\lim_{n\to+\infty} \mathrm{R}_n(g) = \mathrm{R}(g) \text{ p.s.} \tag{26}$$

et

$$\sqrt{n}\left(\mathrm{R}_n(g) - \mathrm{R}(g)\right) \rightsquigarrow \mathcal{N}(0, \sigma^2) \tag{27}$$

avec $\sigma^2 = \mathbb{V}\left(l(\mathrm{Y}, g(\mathrm{X}))\right)$

D'après la loi forte des grands nombres, le risque empirique $\mathrm{R}_n(g)$ est une bonne approximation du risque de prédiction $\mathrm{R}(g)$ lorsque n est suffisamment grand. On peut donc choisir comme fonction de prédiction un minimiseur $\widehat{g}_n$ du risque empirique défini par:

$$\widehat{g}_n = \widehat{g}_{n,\mathcal{G}} = \widehat{g}_n(\mathscr{D}_n, \mathcal{G}) = \operatorname*{arg\,min}_{g\in\mathcal{G}} \mathrm{R}_n(g) \tag{28}$$

où $\mathcal{G} \subset \mathcal{F}$ est le dictionnaire associé au prédicteur.

Prendre $\mathcal{G} = \mathcal{F}$ n'est pas une bonne idée car il existe souvent une infinité de fonctions de prédiction minimisant le risque empirique. Enfin, si l'on prend l'algorithme du plus proche voisin comme minimiseur du risque empirique, on peut montrer qu'il est loin d'être universellement consistant.

$\mathcal{G} = \mathcal{F}$ entraîne souvent un phénomène de surapprentissage. En pratique, il faut prendre $\mathcal{G}$ suffisamment grand pour pouvoir approcher toute fonction, mais pas trop grand pour éviter le surapprentissage. La taille de $\mathcal{G}$ mesure une quantité appelée capacité ou complexité du modèle. On peut également ajouter à $\mathrm{R}_n(g)$ une pénalisation pour éviter une fonction de prédiction trop trop irrégulière.

Lorsque la complexité du modèle augmente (c'est à dire lorsque M augmente), le risque empirique diminue, tandis que - souvent - le risque théorique diminue, puis augmente à nouveau.

Mais le prédicteur de risque empirique minimal pour chaque n n'a aucune raison de converger vers le prédicteur de risque moyen de prédiction minimale (le risque optimal de Bayes). Intuitivement, à n fixé, sur le modèle $\mathcal{G}$ est suffisamment riche, on pourra toujours trouver un prédicteur de risque empirique très faible, même pour une grande base d'apprentisage, mais dont le risque moyen de prédiction est grand. Le cas extrême est le prédicteur g défini par $g(x_n) = y_n$ et qui prend une valeur quelconque ailleurs. (il apprend par coeur la base d'apprentissage). Son risque empirique est nul, mais son risque de prédiction très grand. On dit que sa capacité de généralisation est faible.

La non monotonie du risque théorique est un résultat... empirique, qui est remis en cause depuis quelques années suite à la découverte du phénomène de double-descente. La discussion précédente est valable en petite dimension ($d \leq n$) et l'on peut fréquemment observer qu'en grande dimension (on parle de zone de surparamétrage) le risque de généralisation décroit à nouveau et atteint parfois des valeurs inférieures à celles de la zone sous-paramétrée. Nous reviendrons sur ces aspects.

3.2 Minimisation du risque empirique

Soit $g^\star$ le prédicteur minimisant le risque de Bayes sur $\mathcal{F}$. $R(g^\star)$ mesure le coût moyen (idéal) minimum des erreurs de prédiction sur toutes les observations étiquetées possibles et sur l'ensemble de toutes les fonctions possibles. C'est la quantité que nous avons déjà appelé risque de Bayes optimal, risque théorique ou encore risque de prédiction. Mais $g^\star$ est inconnu (c'est une fonction oracle). On se contente donc de $\widehat{g}_{n,\mathcal{G}}$ qui minimise le risque empirique à partir de l'échantillon, en se limitant à des fonctions g appartenant au dictionnaire $\mathcal{G}$.

$\widehat{g}_{n,\mathcal{G}}$ minimise R_n, mais pas R. On cherche donc à quantifier à quel point le risque de prédiction $R(\widehat{g}_n)$ (qui mesure la capacité à généraliser) est éloigné du risque de prédiction minimal $R(g^\star)$. Pour cela, on va décomposer le l'excès de risque $R(\widehat{g}_{n,\mathcal{G}}) - R(g^\star)$ en deux termes positifs: l'erreur d'approximation (déterministe) et l'erreur d'estimation (stochastique). L'erreur d'approximation, qui s'appelle aussi erreur systématique et qui est homogène à un biais, est la quantité

$$R(g^\star_{\mathcal{G}}) - R(g^\star). \tag{29}$$

C'est l'écart, calculé sur le risque moyen, entre le prédicteur optimal de Bayes sélectionné dans le dictionnaire $\mathcal{G}$ (oracle associé à $\mathcal{G}$) et le prédicteur de Bayes sélectionné parmi toutes les fonctions possibles. Elle ne dépend que de la classe de fonctions paramétrée par θ et ne dépend donc que du dictionnaire.

L'erreur d'estimation (ou risque stochastique), qui est homogène à une variance (et qui est aléatoire car elle dépend de l'échantillon), est égale à

$$R(\widehat{g}_{n,\mathcal{G}}) - R(g^\star_{\mathcal{G}}). \tag{30}$$

C'est l'écart, en terme de risque moyen, entre le minimiseur du risque empirique calculé à partir de l'échantillon sur le dictionnaire $\mathcal{G}$ et le minimiseur idéal de Bayes sur le dictionnaire $\mathcal{G}$.

$$R(\widehat{g}_{n,\mathcal{G}}) - R(g^\star) = \left[R(\widehat{g}_{n,\mathcal{G}}) - R(g^\star_{\mathcal{G}})\right] + \left[R(g^\star_{\mathcal{G}}) - R(g^\star)\right] = \varepsilon_{\text{stochastique}} - \varepsilon_{\text{systématique}}. \tag{31}$$

Quand la taille du dictionnaire $\mathcal{G}$ augmente, l'erreur d'approximation diminue, et l'erreur stochastique devient souvent, en moyenne, plus grande. Le terme « souvent » indique que ce n'est pas un résultat mathématique rigoureux mais plutôt une observation empirique fréquente. Il y a donc un compromis de type biais/variance à trouver (c.f. fig. 3.1).

On peut borner l'erreur stochastique grâce au théorème suivant.

Propriété

Soient $\widehat{g}_n = \widehat{g}_{n,\mathcal{G}}$ et $g^\star = g^\star_{\mathcal{G}}$ respectivement le minimiseur du risque empirique et le prédicteur de Bayes sur le dictionnaire $\mathcal{G}$.

$$0 \leq R(\widehat{g}_n) - R(g^\star) \leq 2 \max_{g \in \mathcal{G}} |R(g) - R_n(g)| \tag{32}$$

Proof. Pour alléger les notations, dans cette démonstration, nous omettons d'indicer par $\mathcal{G}$ $g^\star$ et $\widehat{g}_n$, mais il est essentiel de se souvenir que les fonctions en jeu ici sont sélectionnées dans le dictionnaire $\mathcal{G}$.

Par définition de $g^\star$, $R(g^\star) \leq R(\widehat{g}_{n,\mathcal{G}})$ et par définition de $\widehat{g}_n$, $R_n(g^\star) \geq R_n(\widehat{g}_n)$. Par ailleurs,

$$R(\widehat{g}_n) - R(g^\star) = R(\widehat{g}_n) - R_n(\widehat{g}_n) + R_n(\widehat{g}_n) - R_n(g^\star) + R_n(g^\star) - R(g^\star). \tag{33}$$

En utilisant l'inégalité triangulaire, il vient

$$R(\widehat{g}_n) - R_n(\widehat{g}_n) + R(g^\star) - R_n(g^\star) \leq \left|R(\widehat{g}_n) - R_n(\widehat{g}_n)\right| + \left|R(g^\star) - R_n(g^\star)\right|. \tag{34}$$

et par suite

$$R(\widehat{g}_n) - R_n(\widehat{g}_n) + R(g^\star) - R_n(g^\star) \leq 2 \max_{g \in \mathcal{G}} |R(g) - R_n(g)|. \tag{35}$$

d'où l'inégalité recherchée, puisque $R_n(\widehat{g}_n) \leq R_n(g^\star)$. □

Le risque empirique est un estimateur sans biais et consistant de $R(g)$ on a donc dans tous les cas intérêt à utiliser une grande base d'apprentissage (n grand) pour diminuer la fluctuation sur $\mathcal{G}$.

Théorème 5 — Inégalité oracle pour un dictionnaire fini

Pour une fonction de perte l à valeurs dans $[0,1]$ et un dictionnaire fini $\mathcal{G}$ contenant M fonctions, alors pour tout $\delta \in]0,1[$, avec probabilité supérieure à $1-\delta$,

$$R(\widehat{g}_n) - R(g^\star) \leq \sqrt{\frac{2}{n}\ln\left(\frac{2M}{\delta}\right)} \tag{36}$$

Proof. Nous utilisons l'inégalité de Hoeffding: si $(V_i)_i$ est une suite de v.a.i.i.d. à valeurs presque sûrement dans $[a,b]$, alors $\forall t > 0$,

$$\mathbb{P}\left[\left|\overline{V}_n - \mathbb{E}[\overline{V}_n]\right| \geq t\right] \leq 2\exp\left(-\frac{2nt^2}{(b-a)^2}\right).$$

De

$$R(\widehat{g}_n) - R(g^\star) \leq 2\sup_{g\in\mathcal{G}} |R(g) - R_n(g)|$$

on tire

$$\begin{aligned}
&\mathbb{P}\left[R(\widehat{g}_n) - R(g^\star) \geq 2t\right] \leq \mathbb{P}\left[2\sup_{g\in\mathcal{G}} |R(g) - R_n(g)| \geq 2t\right] \\
&= \mathbb{P}\left[2\max_{m=1}^{M} |R(g_m) - R_n(g_m)| \geq 2t\right] \\
&= \mathbb{P}\left(\bigcup_{m=1}^{M} \left[|R(g_m) - R_n(g_m)| \geq t\right]\right) \\
&\leq \sum_{m=1}^{M} \mathbb{P}\left[|R(g_m) - R_n(g_m)| \geq t\right] = \bullet
\end{aligned}$$

Mais

$$R_n(g_m) = \frac{1}{n}\sum_{i=1}^{n} l(Y_i, g_m(X_i))$$

Posons alors $V_i = h_m(X_i,Y_i) = l(Y_i, g_m(Y_i)) \in [0,1]$. Ces va sont i.i.d. car les (X_i,Y_i) sont i.i.d. (et g_m est une fonction quelconque de $\mathcal{G}$ qui ne dépend pas de l'échantillon). Par ailleurs, l'espérance de $R_n(g_m)$ est égale à $R(g_m)$. On peut alors poser $a=0$ et $b=1$ et appliquer l'inégalité de Hoeffding :

$$\begin{aligned}
\bullet &\leq \sum_{m=1}^{M} 2\exp\left(-\frac{2nt^2}{(1-0)^2}\right) \\
&= 2Me^{-2nt^2}
\end{aligned}$$

Pour avoir un majorant égal à ε, il suffit alors de poser

$$\begin{aligned}
&\varepsilon = 2Me^{-2nt^2} \\
&\iff t = \frac{1}{2}\sqrt{\frac{2}{n}\ln\left(\frac{2M}{\varepsilon}\right)}.
\end{aligned}$$

Finalement, pour obtenir l'inégalité demandée, il suffit de passer à l'évènement complémentaire.

□

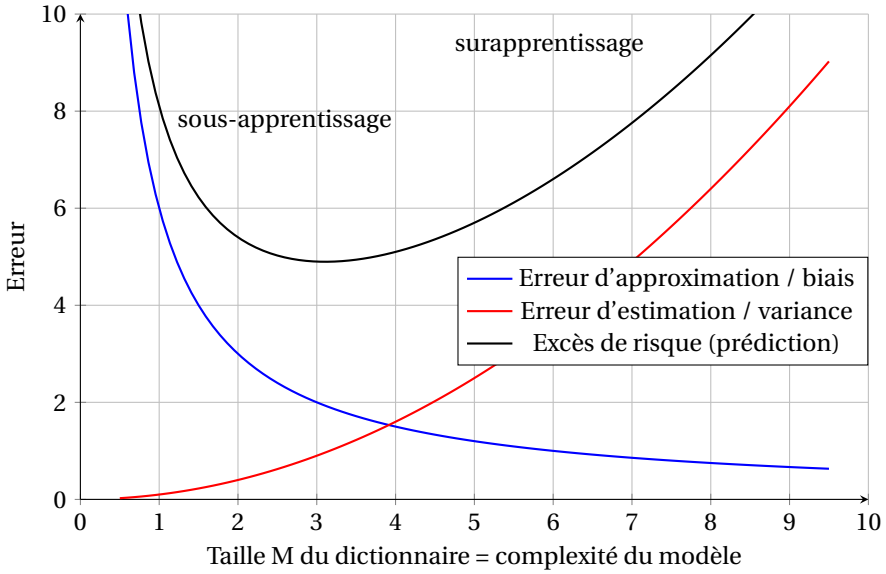


Figure 3.1 – Un exemple de dilemne biais / variance. En bleu, l'erreur d'approximation pour un dictionnaire de taille M. En rouge, l'erreur d'estimation, en noir le risque théorique ou excès de risque. La zone de sous-apprentissage caractérise un biais fort et une variance faible, tandis que la zone de surapprentissage caractérise un biais faible et une variance forte. La courbe bleue est typiquement obtenue avec l'ensemble d'apprentissage et la courbe noir avec l'ensemble de test.

Quand n tend vers l'infini, la différence tend vers 0: le risque du minimiseur empirique converge donc vers le risque théorique. Ainsi, augmenter la taille de l'échantillon et donc la quantité de données, améliore la qualité du prédicteur empirique. Mais dans le même temps, quand la taille du dictionnaire augmente, le terme de droite augmente... Plus M est petit, plus le risque théorique est grand.

Cette situation est à rapprocher du dilemne biais variance: quand la taille de $\mathcal{G}$ augmente, le biais diminue (c'est à dire l'erreur de modélisation, erreur due au modèle) mais la variance augmente également (erreur statistique due à l'aléa des données). Augmenter M augmente le risque de surapprentissage.

$$\mathrm{R}_n(\widehat{g}_n(\mathrm{D}_n, \mathcal{G}_\mathrm{M})) \leq \mathrm{R}_n(\widehat{g}_n(\mathrm{D}_n, \mathcal{G}_{\mathrm{M}'})) \text{ si } \mathrm{M} < \mathrm{M}.' \tag{37}$$

Redécouvert aux alentours de 2018, le phénomène de double-descente (c.f. fig. 3.2) s'intéresse aux modèles sur-paramétrés, lorsque le nombre de paramètres et/ou la complexité du modèle est plus grand que la taille de l'échantillon d'entraînement. C'est typiquement le cas dans les méthodes de réseaux de neurones, dont les instances modernes possèdent facilement plusieurs dizaines voire centaines de milliards de paramètres. En ce cas, le dilemne biais–variance laisse place à une seconde descente qui permet parfois d'atteindre des niveaux d'erreur inférieurs à ceux de la zone sous-paramétrée. Nous ne détaillerons pas plus ce phénomène qui dépasse le cadre du cours et renvoyons le lecteur à [Bach, 2024].

La complexité des modèles, la présentation du dilemme biais-variance et du phénomène de double-descente est exposée par Stéphane Mallat dans son cours du Collège de France de 2018, dont les vidéos sont disponibles en ligne en suivant ce lien :
https://www.college-de-france.fr/site/stephane-mallat/course-2017-2018.htm

Nous terminons cette section par quelques exemples de dictionnaires sur lesquels nous reviendrons en détail dans les chapitres suivants.

- Régression linéaire : $\mathcal{G} = \{h : x \mapsto h(x) = b + w^\top x\}$.
- Régression logistique : $\mathcal{G} = \{h : x \mapsto h(x) = \sigma(b + w^\top x)\}$.
- Classifieur bayésien naïf binaire : $\mathcal{G} = \{h : x \mapsto h(x) = \mathbb{1}_{[\mathbb{P}[\mathrm{Y}=1|\mathrm{X}=x]>\frac{1}{2}]}\}$.
- Classifieur bayésien naïf multinomial : $\mathcal{G} = \{h : x \mapsto h(x) = \mathbb{1}_{[\arg\max_k \mathbb{P}[\mathrm{Y}=y_k|\mathrm{X}=x]]}\}$.

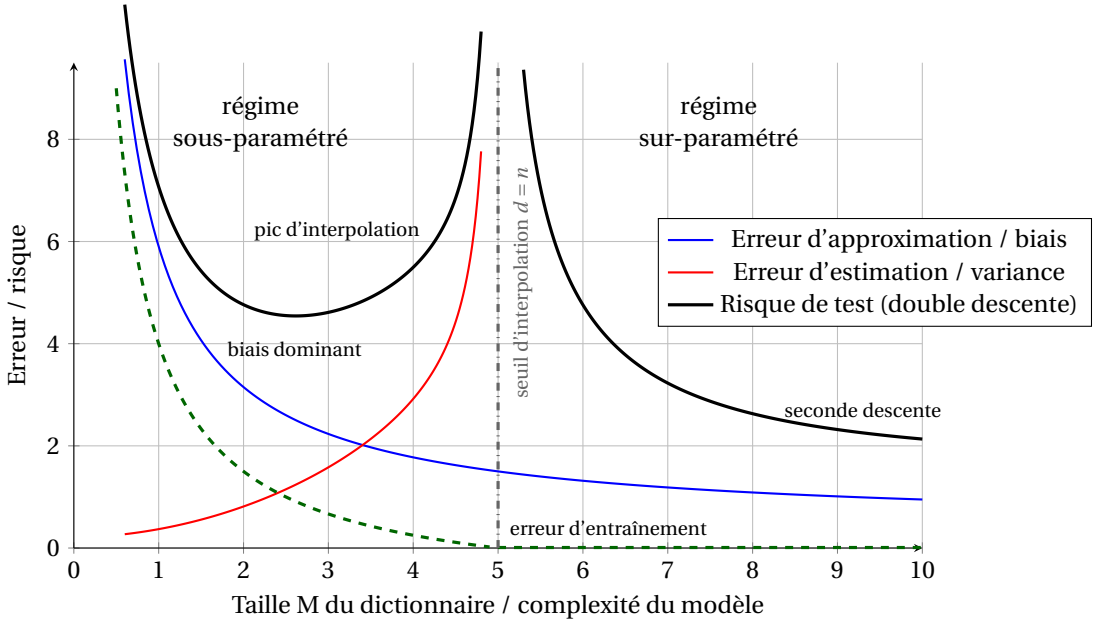


Figure 3.2 – Illustration schématique du phénomène de double descente. Dans le régime sous-paramétré, le risque de test suit le compromis biais–variance classique. Au voisinage du seuil d'interpolation (où le nombre de paramètres $d = \mathrm{M}$ est égal à la taille de l'échantillon d'entraînement n), il présente un pic, puis redescend dans le régime sur-paramétré. La courbe verte en pointillés représente l'erreur d'entraînement, qui devient nulle à partir du seuil d'interpolation.

- Réseaux de neurones : $\mathcal{G} = \{h : x \mapsto h(x)\}$ avec h tel que

$$h(x) = \bigcirc_{i=1}^{\mathrm{L}} \sigma(w_i^{\mathrm{T}} x + b_i) = \sigma\left(w_{\mathrm{L}} \sigma\left(\ldots . \sigma\left(w_1^{\mathrm{T}} x + b_1\right) \ldots\right)\right)$$

où σ fonction logistique.

3.3 Sélection de modèles et validation croisée

Nous profitons de ce paragraphe pour revoir les définitions introduites dans les paragraphes précédents.

$\mathrm{R}(g^\star)$ est le risque de Bayes, que nous avons aussi appelé risque optimal, risque théorique ou encore risque de prédiction. $\mathrm{R}_n = \mathrm{R}_n(g)$ est le risque empirique calculé uniquement à partir de l'échantillon et défini par l'estimateur plug-in du risque de Bayes. $\widehat{g}_n$ minimise le risque empirique R_n, mais pas le risque théorique R; c'est pourtant le seul prédicteur que l'on puisse calculer et utiliser en pratique. Mais cela explique pourquoi le risque empirique n'est pas forcément une bonne approximation du risque théorique.

Sélectionner un modèle signifie sélectionner le dictionnaire $\mathcal{G}$ dans lequel les prédicteurs seront choisis. Un modèle est donc simplement un sous-ensemble de $\mathcal{F}$, ensemble de toutes les fonctions (prédicteurs) possibles.

L'estimateur minimisant le risque empirique sur $\mathcal{G}$ est $\widehat{g}_n$ (comme dans le paragraphe précédent, on omet $\mathcal{G}$ pour alléger la notation mais $\widehat{g}_n$ dépend évidemment de $\mathcal{G}$). Nous avons vu que $\mathrm{R}(\widehat{g}_n)$ est possiblement éloigné de $\mathrm{R}_n(\widehat{g}_n)$. Ces deux quantités dépendent de l'échantillon $\mathcal{D}_n$ au travers de $\widehat{g}_n$, ce sont donc des variables aléatoires fonctions de $(\mathrm{X}_i, \mathrm{Y}_i)$: R_n est calculé à partir de l'échantillon $\mathcal{D}_n$. Lorsqu'on applique R_n à son minimiseur $\widehat{g}_n$ on obtient une moyenne empirique de termes de la forme $l(\mathrm{Y}_i, \widehat{g}_n(\mathrm{X}_i))$ qui ne sont pas indépendants les uns des autres, car $\widehat{g}_n$ dépend déjà de $\mathcal{D}_n$. On ne peut donc pas utiliser R_n pour évaluer un estimateur qui a été construit à partir du même échantillon.

Pour contourner ce problème, trois méthodes sont possibles : le découpage simple des données en un jeu d'entraînement et un jeu de test, la validation "Hold-out" ou la méthode de validation croisée.

Le principe du découpage basique de $\mathcal{D}_n$ en un jeu d'entraînement et de test est le suivant : on entraîne le modèle sur le sous-échantillon d'entraînement et on évalue ses performances sur l'échantillon de test, qui est alors indépendant de l'estimateur.

La méthode de validation "Hold-out" est utile lorsqu'on effectue de la sélection de modèles, c'est à dire qu'on souhaite tester plusieurs jeux d'hyperparamètres afin de déterminer ceux qui maximisent les performances. Le

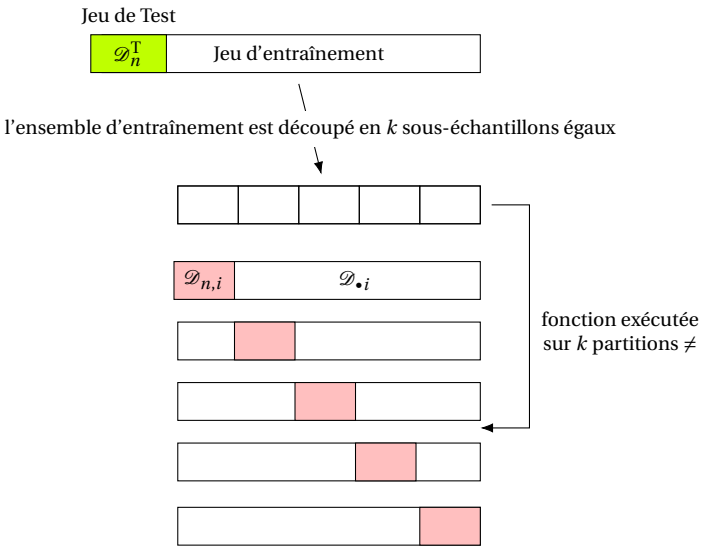


Figure 3.3 – Illustration de la validation croisée k-Fold pour $k = 5$. La base initiale est $\mathscr{D}_n$, partitionnée en trois sous-échantillon : $\mathscr{D}_n^{\mathrm{T}}$ (en vert, fixe durant tout le processus), $\mathscr{D}_{n,i}$ (en rose) et $\mathscr{D}_{\bullet i}$ qui varient à chaque itération. Souvent, $k = 5$ ou $k = 10$. $k = n$ est la méthode "leave-one-out".

découpage en deux ne suffit plus, car les deux sous-échantillons deviennent alors dépendants et il est nécessaire de conserver une troisième partie des données indépendante pour valider l'ensemble des modèles (correspondant chacun à un jeu d'hyperparamètres). En notant $\mathscr{D}_n^{\mathrm{E}}$, $\mathscr{D}_n^{\mathrm{T}}$, $\mathscr{D}_n^{\mathrm{V}}$ respectivement le sous-échantillon d'entraînement, de test et de validation, on procède comme suit :

1 • Pour chaque jeu d'hyperparamètres, on entraîne un estimateur $g_1, \ldots, g_m$ sur $\mathscr{D}_n^{\mathrm{E}}$ avec $g_i = g_i(., \mathscr{D}_n^{\mathrm{E}})$.

2 • Pour chaque estimateur, on évalue le risque empirique $\mathrm{R}_{n,\mathscr{D}_n^{\mathrm{V}}}(g_1), \ldots, \mathrm{R}_{n,\mathscr{D}_n^{\mathrm{V}}}(g_m)$ sur le sous-échantillon $\mathscr{D}_n^{\mathrm{V}}$.

3 • On sélectionne celui qui minimise le risque empirique :

$$\widehat{g}_n = \underset{i=1,\ldots,m}{\arg\min}\, \mathrm{R}_{n,\mathscr{D}_n^{\mathrm{V}}}\left(g_i(\mathscr{D}_n^{\mathrm{E}}, .)\right) \tag{38}$$

4 • On évalue les performances finales du modèle avec le risque calculé à l'aide du sous-échantillon de test :

$$\widehat{\mathrm{R}}_n = \mathrm{R}_{n,\mathscr{D}_n^{\mathrm{T}}}\left(\widehat{g}_n\right). \tag{39}$$

Il y a deux inconvénients à cette méthode : ce découpage réduit les données disponibles pour l'entraînement et les performances dépendent fortement du choix du sous-échantillon (une partition mal choisie pourrait donner de mauvais résultats et).

La méthode de validation croisée corrige ce problème et peut être utilisée dans tous les cas de figure, que ce soit pour une estimation unique ou pour une sélection de modèles. C'est donc la méthode à privilégier.

On découpe en trois l'échantillon initial $\mathscr{D}_n$ en construisant un sous-échantillon de test, un autre de validation et un troisième d'entraînement. Le sous-échantillon de test sera noté $\mathscr{D}_n^{\mathrm{T}}$ et représentera traditionnellement 20% des données, même si en pratique on ne respectera pas ce pourcentage. Il restera constant tout au long du traitement des données. Les deux autres sous-échantillons forment une partition du reste de l'échantillon initial et vont varier durant le processus de validation croisée.

Le principe de la validation croisée (c.f. Fig. 3.3) est de diviser les données (hors échantillon de test) en plusieurs sous-ensembles (les plis, ou "folds") et de permuter leurs rôles entre données d'entraînement et données de validation. Voici les différentes étapes d'une validation k-Fold :

1 • Après avoir sélectionné le sous-échantillon de test, on divise les données restantes en k sous-échantillons disjoints de taille égale : $\mathscr{D}_{n,1}, \ldots, \mathscr{D}_{n,k}$.

2 • Pour chaque itération $i = 1, ..., k$, on utilise $\mathscr{D}_{n,i}$ comme ensemble de validation pour évaluer les performances avec un jeu d'hyperparamètres donné, et les $k-1$ autres sous-échantillons $\mathscr{D}_{\bullet i} = \cup_{j \neq i} \mathscr{D}_{n,j}$ comme base d'entraînement. Après k itérations, chaque sous-échantillon a servi une seule fois de données de validation.

3 • On sélectionne l'estimateur qui minimise le risque empirique (calculé à chaque fois sur son échantillon de validation) :

$$\widehat{g}_n = \underset{i=1,\ldots,k}{\arg\min} \mathrm{R}_{n,\mathscr{D}_{\bullet i}} \left(g_i(\mathscr{D}_{n,i}, .) \right) \tag{40}$$

• On évalue la performance globale en calculant la moyenne des scores obtenus à chaque itération. Ces scores sont les risques empiriques de chaque estimateurs, calculés avec le sous-échantillon de test.

$$\widehat{\mathrm{R}}_n = \frac{1}{k} \sum_{i=1}^{k} \mathrm{R}_{n,\mathscr{D}_n^{\mathrm{T}}} \left(g_i(\mathscr{D}_{n,i}, .) \right) \tag{41}$$

En pratique, on effectue souvent une re-calibration en ré-entraînant le modèle correspondant au meilleur jeu d'hyperparamètres sur la totalité des données (hors données de test). L'analyse théorique de la méthode de validation croisée est un champ de recherche très actif [Bach, 2024, Arlot and Celisse, 2010].

Chapitre 4

Régressions

⚠ En cours de réalisation...

4.1 Rappels de base

Une régression linéaire est de la forme $Y = X\beta + \varepsilon = f(X,\beta) + \varepsilon$ avec f fonction linéaire. Une régression non linéaire est de la forme $Y = f(X,\beta) + \varepsilon$ avec f fonction non-linéaire. Une régression logistique : $Y = \Lambda(X.\beta) + \varepsilon$ avec Λ fonction logistique.

Dans une régression, on cherche à expliquer une variable aléatoire Y à partir de variables aléatoires (explicatives) X_i, qualitatives ou quantitatives.

Il peut s'agir de régression simple (une seule variable explicative quantitative), multiple (plusieurs variables explicatives quantitatives), d'analyse de la variance (une ou plusieurs variables explicatives, quantitatives) ou d'analyse de la covariance (variables explicatives quantitatives ou qualitatives).

Lorsque Y est une variable qualitative, le modèle linéaire n'est pas adapté. Par exemple, pour une variable binaire (on supposera qu'elle peut prendre comme valeurs 0 et 1) il est difficile d'approcher par une droite deux nuages de points ne prenant comme ordonnées que 0 ou 1. Pour expliquer une variable qualitative, on peut alors effectuer une régression logistique, une analyse discriminante, ou bien encore des techniques d'apprentissage supervisé type k-plus proches voisins.

4.1.1 Hypothèses du modèle

Dans le cas d'une régression simple, on doit supposer que le lien entre Y et X est donné par une relation linéaire, perturbée par un bruit aléatoire :

$$Y = \beta_0 + \beta_1 X + \varepsilon \tag{1}$$

ε est un bruit aléatoire, β_0 et β_1 les coefficients (inconnus) de la régression. Pour les déterminer, on les estime à partir d'un échantillon de n mesures observées (x_i, y_i), déterministes. Les n équations obtenues :

$$y_i = \beta_0 + \beta_1 x_i + \varepsilon_i \; ; i = 1, .., n \tag{2}$$

permettent le calcul de β_0 et β_1 qui sont solutions de l'équation des moindres carrés

$$\left(\widehat{\beta}_0, \widehat{\beta}_1\right) = \min_{\beta_0,\beta_1} \sum_{i=1}^{n} \left(y_i - \beta_0 - \beta_1 x_i\right)^2 \tag{3}$$

La droite de régression est alors donnée par

$$\widehat{f}(x) = \widehat{y}_i = \widehat{\beta}_0 + \widehat{\beta}_1 x \tag{4}$$

et l'erreur est donnée par les résidus

$$\widehat{\varepsilon}_i = y_i - \widehat{y}_i \tag{5}$$

Parfois, la régression s'écrit avec un Y_i majuscule ($Y_i = \beta_0 + \beta_1 x_i + \varepsilon_i$), pour indiquer que Y_i est une variable aléatoire dépendant de la variable aléatoire ε_i. Les x_i minuscules indiquent que l'on travaille sachant $[X_i = x_i]$,

mais que ε_i est toujours considéré en tant que variable aléatoire. Dans la suite, nous utiliserons toujours y_i minuscule et c'est en fonction du contexte qu'il faudra savoir si y_i est une observation (lorsque ε_i est une observation déterministe du bruit) ou bien une variable aléatoire (lorsque ε_i est une variable aléatoire).

Dans une régression linéaire multiple, la relation liant Y aux variables $X_1, ..., X_p$ est de la forme

$$Y = \beta_0 + \beta_1 X_1 + ... + \beta_p X_p + \varepsilon \tag{6}$$

Considérons un échantillon $(x_i, y_i)_{i=1,..,n}$ de données issues de (X,Y). On obtient ainsi n équations liant les y_i observés aux x_i et aux β_i.

$$\begin{cases} y_1 = \beta_0 + \beta_1 x_{11} + ... + \beta_p x_{1p} + \varepsilon_1 \\ \vdots \\ y_i = \beta_0 + \beta_1 x_{i1} + ... + \beta_p x_{ip} + \varepsilon_i \\ \vdots \\ y_n = \beta_0 + \beta_1 x_{n1} + ... + \beta_p x_{np} + \varepsilon_n \end{cases} \tag{7}$$

x_{ij} représente la $i$ème observation du vecteur aléatoire (en majuscule) X_j dans l'échantillon. Chaque vecteur ligne (en minuscule) $x_i = (x_{i1}, ..., x_{ip})$ représente une observation des p vecteurs X_j, pour $j = 1, .., p$.

Il est plus pratique d'utiliser une notation matricielle : en notant $\mathbb{X} = \left(x_{ij}\right)$ on a :

$$\mathbb{X} = \begin{pmatrix} 1 & x_{11} & ... & x_{1p} \\ \vdots & \vdots & & \vdots \\ 1 & x_{n1} & ... & x_{np} \end{pmatrix} \tag{8}$$

Dans la matrice $\mathbb{X} \in \mathbb{R}^{n\times(p+1)}$, la ligne i $(x_{i1}, ...x_{ip})$ représente un individu, c'est à dire la $i$ème observation de toutes ses variables et la colonne j (sauf la première colonne formée de 1 uniquement) représente les différentes observations relatives à une variable X_j. $y \in \mathbb{R}^n$, $\beta \in \mathbb{R}^{p+1}$ et $\varepsilon = (\varepsilon_1, ..., \varepsilon_n) \in \mathbb{R}^n$ sont maintenant des vecteurs et les coordonnées de ε sont des copies i.i.d. du bruit défini dans (6). L'équation de régression devient :

$$y = \mathbb{X}\beta + \varepsilon \tag{9}$$

dont la solution au sens des moindres carrés est donnée par

$$\widehat{\beta} = \underset{\beta\in\mathbb{R}^{p+1}}{\arg\min} \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 \tag{10}$$

$$= \underset{\beta\in\mathbb{R}^{p+1}}{\arg\min} \left((y - \mathbb{X}\beta)^{\mathrm{T}} \times (y - \mathbb{X}\beta) \right) \tag{11}$$

$$= \underset{\beta\in\mathbb{R}^{p+1}}{\arg\min} ||y - \mathbb{X}\beta||_2^2 \tag{12}$$

la solution est le vecteur de coefficients le plus proche - au sens des moindres carrés - des valeurs observées. L'idée de base est un peu d'inverser la relation $y = \mathbb{X}\beta$ et d'écrire $\beta = \mathbb{X}^{-1} y$. Oui mais voilà, la matrice n'est pas inversible (elle n'est pas même pas carrée). Il existe une notion de pseudo-inverse au sens de Moore Penrose qui s'écrit $\mathbb{X}^{\dagger} \in \mathbb{R}^{p\times p}$ et qui coïncide exactement avec la solution au sens des moindres carrés (si cette matrice est de rang plein),

$$\widehat{\beta} = \mathbb{X}^{\dagger} y = \left(\mathbb{X}^{\mathrm{T}}\mathbb{X}\right)^{-1} \mathbb{X}^{\mathrm{T}} y \tag{13}$$

Si $n \geq p$ et si $\mathbb{X}$ est de rang plein, alors $\mathbb{X}^{\mathrm{T}}\mathbb{X}$ est inversible et l'équation précédente est licite.

Certaines hypothèses du modèle linéaire concerne le bruit ε (hypothèses de Gauss-Markov) :

$$\varepsilon \sim \mathcal{N}(0, \sigma^2 \mathrm{I}_n) \tag{14}$$

Autrement dit, ε est un vecteur aléatoire gaussien, centré, dont les coordonnées sont i.i.d. de variance commune σ^2 (homoscédasticité et non corrélation des résidus). On suppose également que le bruit n'est pas corrélé aux variables explicatives $\mathbb{E}[\varepsilon|X] = 0$ (exogénéïté des variables explicatives: le bruit ne doit pas être corrélé aux variables explicatives). On a alors (c'est le théorème de Gauss Markov) :

$$y \sim \mathcal{N}(\mathbb{X}\beta, \sigma^2 \mathrm{I}_n) \tag{15}$$

On peut prouver que sous les hypothèses précédentes $\widehat{\beta}$ est un estimateur sans biais de β de variance minimale égale à $\mathbb{V}(\widehat{\beta}) = \sigma^2(\mathbb{X}^T\mathbb{X})^{-1}$ qui coïncide avec l'estimateur du maximum de vraisemblance du modèle et qu'il est VUMSB.

La matrice $\Pi = \mathbb{X}(\mathbb{X}^T\mathbb{X})^{-1}\mathbb{X}^T = \mathbb{X}\mathbb{X}^\dagger$ est la matrice du projecteur orthogonal sur l'espace vectoriel engendré par les colonnes de $\mathbb{X}$ et le vecteur des valeurs ajustées est simplement le projeté orthogonal de y sur l'espace engendré par les colonnes de $\mathbb{X}$.

$$\widehat{y} = \Pi y \tag{16}$$

Le vecteur des résidus est le projeté de y sur l'orthogonal de cet espace, soit:

$$\widehat{\varepsilon} = (\mathrm{I} - \Pi)\, y = y - \widehat{y} \tag{17}$$

La somme des carrés résiduelle est

$$\mathrm{SCR} = ||\widehat{\varepsilon}||^2 = \sum_{i=1}^{n} (y_i - \widehat{y}_i)^2 \tag{18}$$

La somme des carrés totale (recentrée car la colonne de 1 est présente dans la matrice $\mathbb{X}$) est

$$\mathrm{SCT} = ||y - \overline{y}||^2 \tag{19}$$

et la somme des carrés estimée (c'est la variance expliquée par le modèle) est

$$\mathrm{SCE} = ||\widehat{y} - \overline{y}||^2 \tag{20}$$

Le théorème de Pythagore donne l'équation d'analyse de la variance:

$$\mathrm{SCT} = \mathrm{SCE} + \mathrm{SCR} \tag{21}$$

et le coefficient de détermination R^2 est défini par

$$\mathrm{R}^2 = \frac{\mathrm{SCE}}{\mathrm{SCT}} \tag{22}$$

La séquence d'équations suivante :

$$\mathbb{E}[||\widehat{\varepsilon}||^2] = \mathbb{E}[\widehat{\varepsilon}^T.\widehat{\varepsilon}] = \mathbb{E}[\mathrm{Tr}(\widehat{\varepsilon}^T.\widehat{\varepsilon})] = \mathbb{E}[\mathrm{Tr}(\widehat{\varepsilon}.\widehat{\varepsilon}^T)] = \mathrm{Tr}\left(\mathbb{E}[\widehat{\varepsilon}^T.\widehat{\varepsilon}]\right) = \mathrm{Tr}\left(\mathbb{V}(\widehat{\varepsilon})\right) \tag{23}$$

$$= \sigma^2 \mathrm{Tr}(\mathrm{I}_p - \Pi) = \sigma^2(n - \mathrm{Tr}(\mathbb{X}(\mathbb{X}^T\mathbb{X})^{-1}\mathbb{X}^T)) = \sigma^2(n - \mathrm{Tr}(\mathrm{I}_p)) = \sigma^2(n-p) \tag{24}$$

permet de définir un estimateur sans biais de la variance $v = \sigma^2$:

$$\widehat{v} = \frac{||\widehat{\varepsilon}||^2}{n-p} \tag{25}$$

4.1.2 Le problème de la grande dimension

Lorsque $n = p$, si la matrice $\mathbb{X}$ est inversible, la résolution de la régression montre que $\widehat{y}_i = y_i$ pour tout i :

$$\widehat{\beta} = \mathbb{X}^{-1} y. \tag{26}$$

Il s'agit du point d'interpolation : le modèle passe par tous les points de l'échantillon de test et l'erreur d'entraînement est nulle. Mais dans le même temps,

$$\widehat{\beta} = \beta + \mathbb{X}^{-1}\varepsilon. \tag{27}$$

Si $\mathbb{X}$ est mal conditionnée, $\mathbb{X}^T\mathbb{X}$ est presque singulière et l'équation précédente amplifie le bruit : la variance de $\widehat{\beta}$ est très grande et les coefficients sont instables; par ailleurs, l'estimateur de la variance est infini (à cause du $n-p$ au dénominateur).

Lorsque $p > n$, la matrice de covariance $\mathbb{X}^T\mathbb{X}$ est singulière et les estimateurs des moindres carrés ne sont plus consistants. On ne peut plus utiliser la méthode précédente.

À la place on peut utiliser une régression pénalisée (Ridge, Lasso, Elastic Net) qui ajoute un terme diagonal à $\mathbb{X}^T\mathbb{X}$ et rend cette matrice inversible, ou bien utiliser la matrice de covariance $\mathbb{X}\mathbb{X}^T$ à la place. Si $\mathbb{X}$ est de rang plein $n < p$ alors la pseudo-inverse de Moore Penrose est toujours définie et détermine toujours l'estimateur au sens des moindres carrés. On a :

$$\widehat{\beta} = \mathbb{X}^{\dagger} y = \mathbb{X}^{\mathrm{T}} (\mathbb{X}\mathbb{X}^{\mathrm{T}})^{-1} y. \tag{28}$$

avec maintenant $\mathbb{X}^{\dagger} \in \mathbb{R}^{n \times n}$ et parmi tous les vecteurs β possibles (il n'y a plus unicité), la pseudo-inverse sélectionne celui de norme 2 minimale :

$$\widehat{\beta} = \underset{\beta : \mathbb{X}\beta = y}{\arg\min} ||\beta||_2. \tag{29}$$

Par contre, l'estimateur de la variance n'est plus défini et il faut utiliser un estimateur issu (par exemple) d'une régression ridge.

4.1.3 Validation du modèle et qualité de la régression

Cette seconde partie est au moins aussi importante que la première et elle est pourtant souvent négligée. Après la modélisation et l'estimation des paramètres, il faut en effet valider le modèle en évaluant la qualité de la régression :

- Vérifier la pertinence du caractère linéaire.
- Faire le choix d'inclure ou non une variable expliquée.
- Analyser les résidus.
- Regarder les valeurs aberrantes.
- Analyser la normalité.
- Vérifier l'homoscédasticité.
- Regarder l'impact d'une variable sur les résidus partiels.
- Etc.

Si l'hypothèse linéaire n'est pas vérifiée ou bien si la matrice $\mathbb{X}$ n'est pas de rang plein, on peut effectuer une régression avec un terme de régularisation (régression «Ridge», LASSO ou autre).

Chapitre 5

Méthodes à base de partition

5.1 Généralités sur les méthodes à partition

Nous avons vu au chapitre précédent que le choix de la méthode d'apprentissage supervisé se résume à choisir un sous ensemble de fonctions de $\mathbb{X}$ dans $\mathbb{Y}$ pour former le dictionnaire $\mathcal{G}$ et le modèle d'apprentissage. Un choix populaire est de considérer les fonctions constantes par morceaux sur une partition de $\mathbb{X}$. Cela conduit à l'algorithme des plus proches voisins k-ppv ou k-NN (pour k Nearest Neighbors) et aux arbres de décision et de régression avec l'algorithme CART (classification and regression tree)

Soit $\mathcal{F} = \mathcal{F}(\mathbb{X}, \mathbb{Y})$ l'ensemble des fonctions de $\mathbb{X}$ dans $\mathbb{Y}$. Considérons une partition $\mathscr{A} = (\mathrm{A}_1, ..., \mathrm{A}_\mathrm{M})$ de l'espace $\mathbb{X}$ (la réunion des A_i est égale à $\mathbb{X}$ et ils sont 2 à 2 disjoints)

$$\mathbb{X} = \bigcup_{m=1}^{\mathrm{M}} \mathrm{A}_m \tag{1}$$

et soit $\mathcal{G} = \mathcal{G}(\mathscr{A}) \subset \mathcal{F}$ le dictionnaire formé de l'ensemble des fonctions de $\mathbb{X}$ dans $\mathbb{Y}$ qui sont constantes sur chaque élément A_m de la partition $\mathscr{A}$. La partition peut être aléatoire, déterminée en fonction de l'échantillon $\mathscr{D}_n$.

$$\mathcal{G} = \left\{ g : \mathbb{X} \longrightarrow \mathbb{Y} : \ \forall m = 1, ..., \mathrm{M}, \ g \text{ constante sur } \mathrm{A}_m \right\} \tag{2}$$

Le minimiseur du risque empirique associé à cette partition est la fonction

$$\widehat{g}_n = \widehat{g}_{n,\,\mathscr{A}} = \underset{g \in \mathcal{G}}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} l(\mathrm{Y}_i, g(\mathrm{X}_i)) \tag{3}$$

Notons N_m le nombre d'observations X_i de l'échantillon qui se trouvent dans l'élément A_m de la partition :

$$\mathrm{N}_m = \sum_{i=1}^{n} \mathbb{1}_{\mathrm{A}_m}(\mathrm{X}_i) \tag{4}$$

et notons $\overline{\mathrm{Y}}_m$ la moyenne des étiquettes observées dans l'élément A_m :

$$\overline{\mathrm{Y}}_m = \sum_{i=1}^{n} \mathrm{Y}_m \mathbb{1}_{\mathrm{A}_m}(\mathrm{X}_i) \tag{5}$$

On se place dans un cadre de classification binaire, avec $\mathbb{Y} = \{0, 1\}$

Théorème 6 — Classifieur binaire optimal

$$\forall m = 1, ..., \mathrm{M}, \ \forall x \in \mathrm{A}_m, \ \widehat{g}_n(x) = \begin{cases} 1 & \text{si } \overline{\mathrm{Y}}_{\mathrm{A}_m} > 1/2 \\ a_m & \text{si } \overline{\mathrm{Y}}_{\mathrm{A}_m} = 1/2 \\ 0 & \text{si } \overline{\mathrm{Y}}_{\mathrm{A}_m} < 1/2 \end{cases} \tag{6}$$

ce que l'on peut également écrire sous la forme

$$\forall x \in \mathbb{X}, \ \widehat{g}_n(x) = \sum_{m=1}^{\mathrm{M}} \mathbb{1}_{[\overline{\mathrm{Y}}_{\mathrm{A}_m} > 1/2]} \mathbb{1}_{\mathrm{A}_m}(x). \tag{7}$$

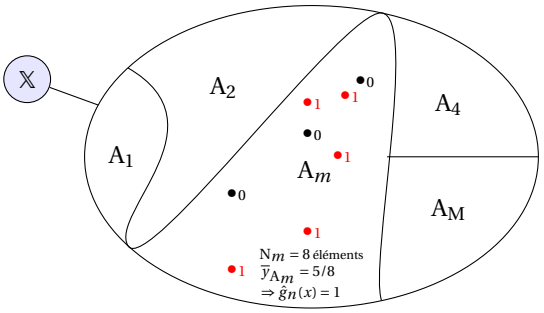


Figure 5.1 – Dans une cellule de la partition de l'espace des « features », l'estimateur prendra comme valeur (constante) la valeur majoritaire des points de l'échantillon d'entraînement présents dans la cellule.

On se place maintenant dans un cadre de régression au sens des moindres carrés.

Théorème 7 — Minimiseur du risque empirique pour la régression MC

$$\widehat{g}_n(x) = \sum_{m=1}^{M} \overline{Y}_{A_m} \mathbb{1}_{A_m}(x) = \sum_{m=1}^{M} \left(\frac{1}{N_m} \sum_{i=1}^{n} Y_i \mathbb{1}_{A_m}(X_i) \right) \mathbb{1}_{A_m}(x) \tag{8}$$

Proof. Nous commençons par la preuve du minimiseur pour la régression au sens des moindres carrés.

$$\frac{1}{n} \sum_{i=1}^{n} l(Y_i, g(X_i)) = \frac{1}{n} \sum_{i=1}^{n} (Y_i - g(X_i))^2. \tag{9}$$

Soit $g \in \mathcal{G}$ telle que $g(x) = a_m$, $\forall x \in A_m$. g est identifiée de façon unique comme vecteur $(a_i, ..., a_M)^T \in \mathbb{R}^M$.

$$R_n(g) = \frac{1}{n} \sum_{i=1}^{n} \sum_{m=1}^{M} (Y_i - g(X_i))^2 \mathbb{1}_{A_m}(X_i) \tag{10}$$

$$= \frac{1}{n} \sum_{m=1}^{M} \sum_{i=1}^{n} (Y_i - a_m)^2 \mathbb{1}_{A_m}(X_i) \tag{11}$$

$$\min_{g \in \mathcal{G}} R_n(g) = \min_{a \in \mathbb{R}^M} \left(\frac{1}{n} \sum_{i=1}^{n} \sum_{m=1}^{M} (Y_i - g(X_i))^2 \mathbb{1}_{A_m}(X_i) \right) \tag{12}$$

$$= \frac{1}{n} \sum_{m=1}^{M} \min_{a \in \mathbb{R}^M} \left(\sum_{i=1}^{n} (Y_i - a_m)^2 \mathbb{1}_{A_m}(X_i) \right) \tag{13}$$

$$\sum_{i=1}^{n} (Y_i - a_m)^2 \mathbb{1}_{A_m}(X_i) = \sum_{i=1}^{n} Y^2 \mathbb{1}_{A_m}(X_i) - 2a_m \sum_{i=1}^{n} Y_i \mathbb{1}_{A_m}(X_i) + a_m^2 N_m. \tag{14}$$

Il s'agit d'une fonction quadratique en a_m^2 dont le minimum est atteint en

$$a_m = \frac{1}{N_m} \sum_{i=1}^{n} Y_i \mathbb{1}_{A_m}(X_i). \tag{15}$$

Les mêmes arguments sont valables pour le minimiseur de la classification binaire :

$$\min_{g \in \mathcal{G}} R_n(g) = \frac{1}{n} \sum_{m=1}^{M} \min_{a_m = 0,1} \left(\sum_{i=1}^{n} \mathbb{1}_{[Y_i \neq a_m]} \mathbb{1}_{A_m}(X_i) \right) \tag{16}$$

$$\widehat{a}_m \in \arg\min_{a=0,1} \sum_{i=1}^{n} \mathbb{1}_{[Y_i \neq a]} \mathbb{1}_{A_m}(X_i) \tag{17}$$

et cette expression vaut 0 ou 1 selon que

$$\sum_{i=1}^{n} \mathbb{1}_{[Y_i \neq 0]} \mathbb{1}_{A_m}(X_i) < \sum_{i=1}^{n} \mathbb{1}_{[Y_i \neq 1]} \mathbb{1}_{A_m}(X_i) \tag{18}$$

ou pas. Si $Y \in \{0,1\}$, $\mathbb{1}_{[Y \neq 0]} = Y$ et $\mathbb{1}_{[Y \neq 1]} = 1 - Y$ et par suite,

$$\sum_{i=1}^{n} \left(\mathbb{1}_{[Y_i \neq 0]} - \mathbb{1}_{[Y_i \neq 0]} \right) \mathbb{1}_{A_m}(X_i) = \sum_{i=1}^{n} (2Y_i - 1) \mathbb{1}_{A_m}(X_i) = 2N_m(\overline{Y}_{A_m} - 1/2). \tag{19}$$

On a donc $\widehat{a}_m = 0$ si, et seulement si, $\overline{Y}_m < 1/2$. □

On peut résumer le résultat des deux théorèmes de la façon suivante : pour estimer $\mathbb{E}[Y|X = x]$ on effectue un moyennage local des Y_i par les X_i les plus proches de x.

Il est facile de généraliser la formule du classifieur binaire à une situation multi-classes dans laquelle Y peut prendre K valeurs différentes. Si $\mathbb{Y} = \{1, ..., K\}$,

$$\forall x \in \mathbb{X},\ \widehat{g}_n(x) = \underset{k=1,...,K}{\arg\max} \sum_{m=1}^{M} \left(\frac{1}{N_m} \sum_{i=1}^{n} \mathbb{1}_{[Y_i=k]} \mathbb{1}_{A_m}(X_i) \right) \mathbb{1}_{A_m}(x) \tag{20}$$

$$= \underset{k=1,...,K}{\arg\max} \sum_{m=1}^{M} w_{m,k} \mathbb{1}_{A_m}(x). \tag{21}$$

où $w_{m,k} = N_m(k)/N_m$ est la proportion des observations dans A_m pour lesquelles $Y = k$.

La démonstration qui précède et les résultats des deux théorèmes sont valables pour toutes les fonctions constantes sur chaque élément d'une partition. Elles ne dépendent pas de la façon dont sont construites les partitions. Nous examinons maintenant deux méthodes différentes de constructions de partitions qui mènent d'une part à la méthode des k plus proches voisins et d'autre part aux arbres de décision.

5.2 La méthode des k plus proches voisins

5.2.1 Principe des k-ppv

Soit $k \leq n$. Pour chaque $x \in \mathbb{R}^d$ et chaque $i = 1, ..., n$, on note $d_i(x) = ||X_i - x||$ la distance entre x et chacune des observations X_i. On définit également les statistiques de rang $r_i(x)$ comme étant l'indice du $i$ème plus proche voisin de x parmi les $X_1, ..., X_n$. Mathématiquement, les rangs sont définis comme suit :

$$r_1(x) = j \iff \begin{cases} d_j(x) = \min\limits_{i=1,...,n} d_i(x) \\ d_j(x) < \min\limits_{1 \leq i < j} d_i(x) \end{cases} \tag{22}$$

$$\tag{23}$$

et par récurrence sur $k \geq 1$,

$$r_k(x) = j \iff \begin{cases} d_j(x) = \min\limits_{i=1..n; i \neq r_1, .., r_{k-1}} d_i(x) \\ d_j(x) < \min\limits_{1 \leq i < j; i \neq r_1, .., r_{k-1}} d_i(x) \end{cases} \tag{24}$$

$$\tag{25}$$

Les inégalités sont nécessaires pour prouver l'existence du rang de façon unique. Pour un entier $k \in [\![1..n]\!]$, les statistiques de rang permettent de définir une partition de l'espace $\mathscr{A}_k$ dont les éléments A_m sont donnés, pour tout $m = 1, ..., \begin{pmatrix} n \\ k \end{pmatrix}$, par :

$$A_m = \{x \in \mathbb{X} = \mathbb{R}^d :\ C_m = (r_1(x), ..., r_k(x))\} \tag{26}$$

C_m étant une combinaison donnée de k éléments choisis parmi n. Les ensembles A_m sont donc les parties de $\mathbb{X}$ pour lesquelles l'application $x \mapsto (r_1(x), ..., r_k(x))$ est constante. En d'autres termes, la partition est constituée de zones caractérisées par un choix de k observations parmi n de telle sorte que les points de cette zone soient les plus près des k observations caractérisant la zone. Pour deux points $x, x' \in A_m$, les k plus proches voisins de x et x' parmi les $X_1, ..., X_n$ sont les mêmes. Si $x \in A_m$ et $x' \in A_{m'}$ avec $m \neq m'$,les k plus proches voisins de x sont différents des k plus proches voisins de x'.

De telles partitions s'appellent des diagrammes de Voronoi et chaque zone forme une cellule de Voronoi. Les deux figures suivantes (Fig. 5.2 et Fig. 5.3) illustrent des partitions au sens du plus proche voisin pour $k = 1$. Voir également l'animation suivante en ligne : https://strongriley.github.io/d3/ex/voronoi.html.

On peut démontrer que si k dépend de n ($k = k_n$) et tend vers l'infini moins vite que n, c'est à dire si $k/n \longrightarrow 0$ quand n tend vers l'infini, alors le prédicteur k-NN est consistant.

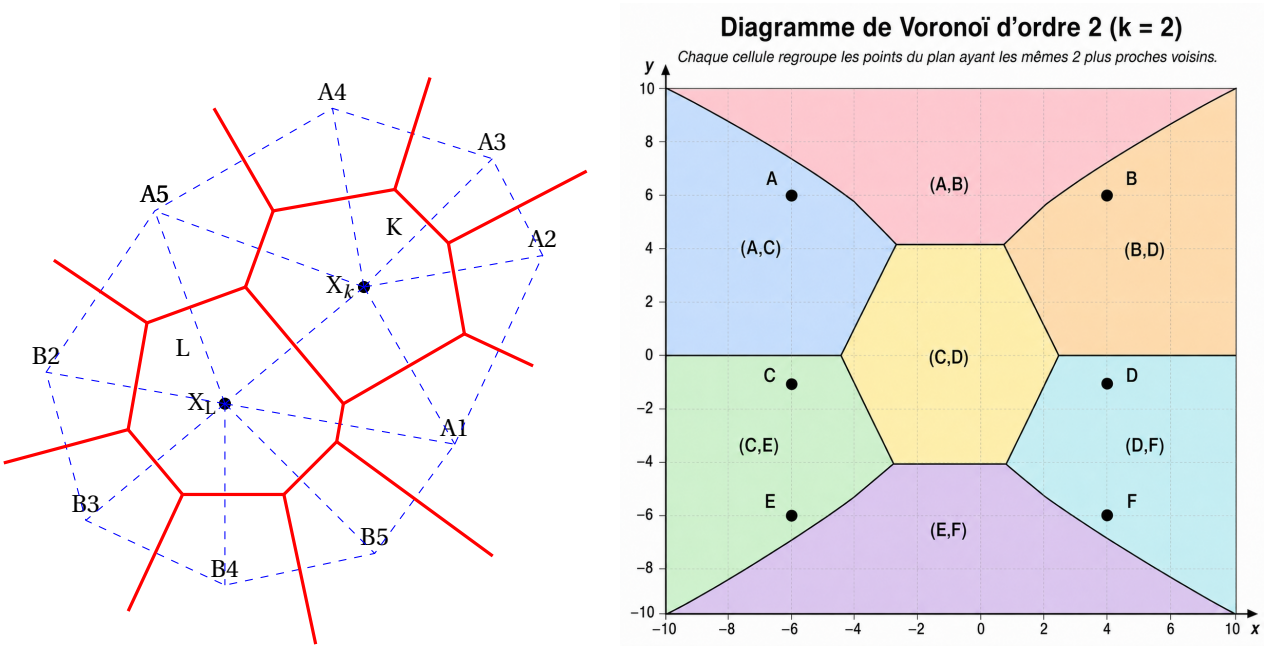


Figure 5.2 – k-ppv et cellule de Voronoï.

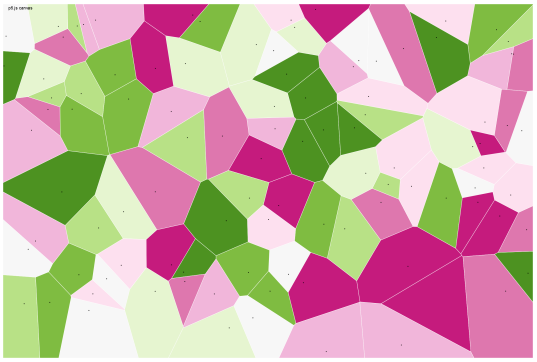


Figure 5.3 – Cellules de Voronoi pour $k = 1$.

5.2.2 Exemples en classification et régression

Un exemple incontournable en apprentissage statistique est la classification des iris de Fisher (nous le traiterons en TP). Le jeu de données initial a été utilisé par Ronald Fisher en 1936 à partir de données collectées par Edgar Anderson, pour illustrer la méthode d'analyse discriminante linéaire. Il comprend 50 échantillons de chacune des 3 espèces d'iris (setosa, virginica et versicolor). 4 variables statistiques ont été mesurées pour chaque échantillon : la longueur et la largeur des sépales et des pétales, en centimètres. Les iris forment donc un échantillon avec $n = 50$ et $d = 4$. La classification n'est pas binaire (ici $k = 3$), mais on regroupe souvent 2 des 3 espèces pour illustrer cette méthode.

La figure Fig. 5.5 illustre ce jeu de données et le résultat d'une classification des 3 espèces.

La figure Fig. 5.6 illustre une tâche de régression avec la méthode des k-ppv. Un nuage de points est créé à partir d'une sinusoïde perturbée par un bruit aléatoire et il s'agit, avec le régresseur k-ppv, de reconstituer au mieux la courbe initiale à partir du nuage.

5.2.3 Remarques sur la méthode des k-ppv

Une méthode non paramétrique

La méthode des k-pvv est dite non paramétrique parce qu'elle ne suppose pas une forme fonctionnelle fixe pour la relation entre X et Y. Pour les k-ppv, on ne cherche pas à estimer un vecteur fini de paramètres θ. On



Figure 5.4 – Les trois espèces d'Iris (de gauche à droite versicolor, setosa, virginica), pour le plaisir de mettre des photos de fleurs dans un polycopié aussi aride. Crédit photo : Frank Mayfield et Kosaciec Szczecinkowaty, Wikipédia, page des iris.

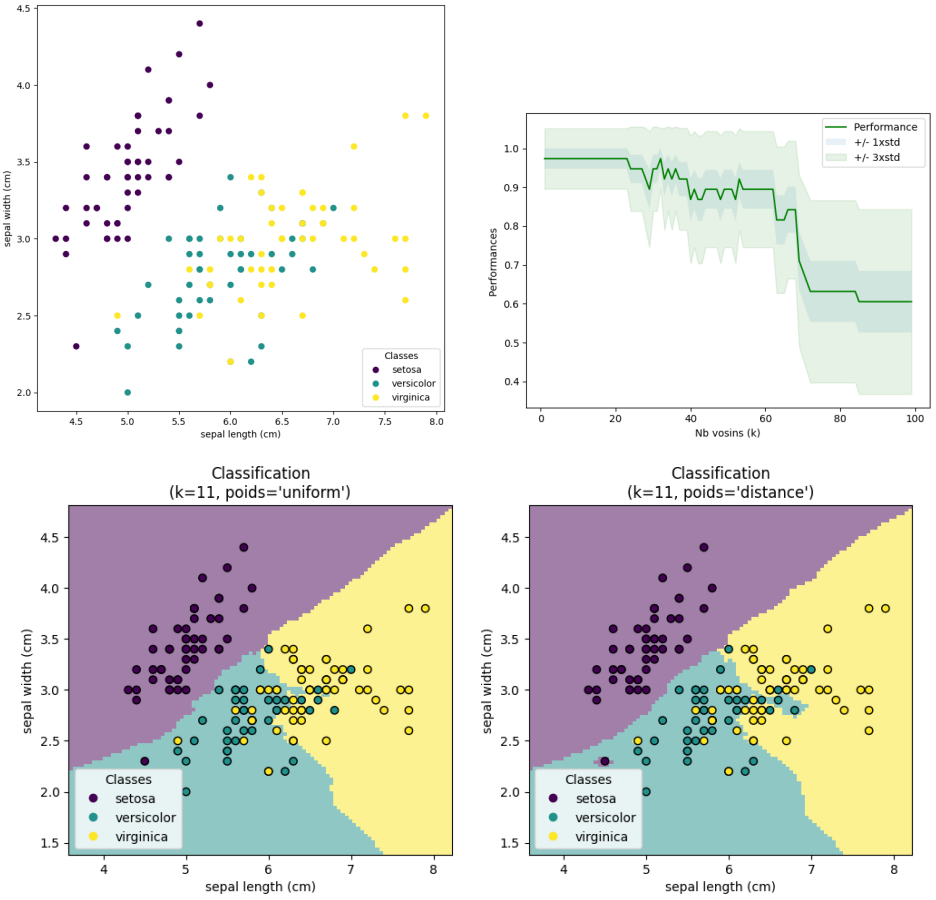


Figure 5.5 – Haut : à gauche, nuage de points représentant les iris de Fisher (une espèce par couleur) en fonction de la longueur et de la largeur de leur sépale. À droite, une courbe indiquant les performances de k-ppv sur la tâche de classification des iris, en fonction du nombre k. Bas : illustration d'un classifieur k-ppv pour $k = 11$, avec deux types de poids différents. Les zones de couleur représentent l'espèce qui sera affectée à une nouvelle observation selon la longueur et la largeur de son sépale. Simulations effectuées sous Python.

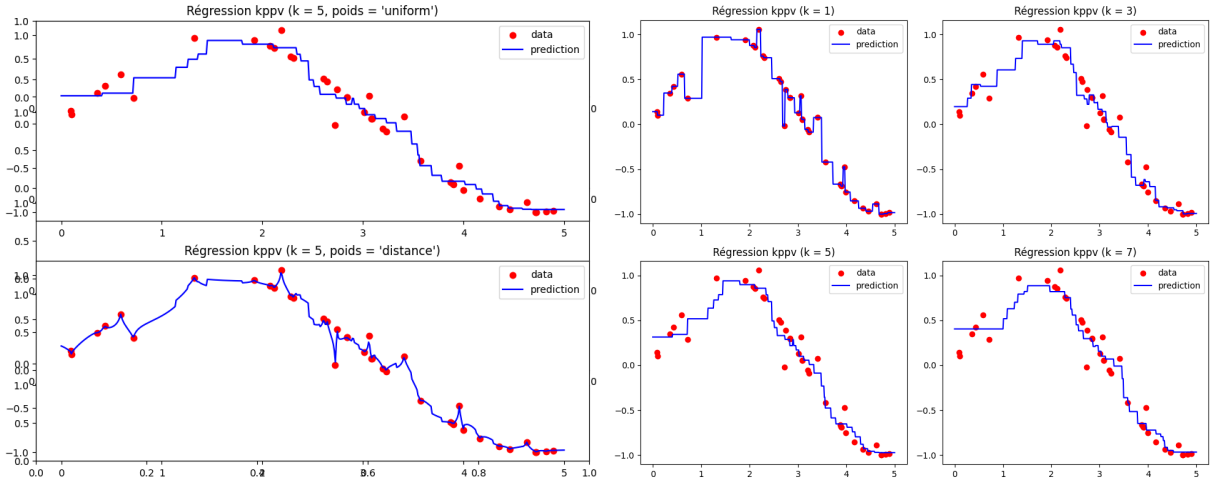


Figure 5.6 – Régression à l'aide de la méthode k-ppv. Un nuage de points (rouges) bruité est généré en perturbant aléatoirement les points d'une sinusoïde. La courbe bleue, issue de la méthode des k-ppv, doit reconstituer au mieux cette sinusoïde. Les deux courbes à gauche correspondent à $k = 5$ pour deux types de poids différents. Les 4 courbes de droite représentent la régression pour des valeurs différentes de $k = 1, 3, 5, 7$. Simulations effectuées sous Python.

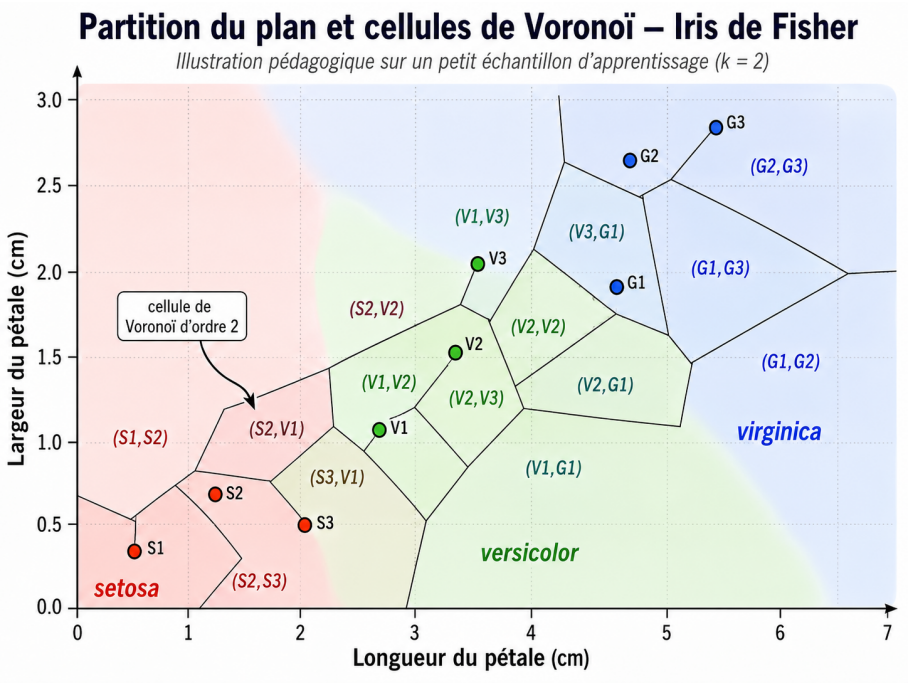


Figure 5.7 – ..

garde tout l'échantillon d'apprentissage : la fonction apprise dépend directement des données stockées. Quand n augmente, le modèle devient plus riche, car il dispose de plus de points de référence. Le terme non paramétrique ne veut pas dire « sans aucun paramètre ». Il y a des choix à faire et des paramètres à calibrer, mais qui ne définissent pas un modèle de dimension fixe :

- le nombre de voisins k,
- la distance utilisée (euclidienne, Manhattan, etc.)
- une pondération des voisins.
- la normalisation des variables.

Non paramétrique : la complexité du modèle croit avec n.

Le choix de k

k est l'hyperparamètre le plus important du modèle. Le choix peut s'effectuer par validation croisée pour obtenir un compromis entre variabilité et lissage, il peut aussi d'effectuer par la méthode du coude («elbow») comme pour la méthode k-means.

- k petit : modèle flexible, mais plus instable (overfitting).
- k grand : modèle plus stable, mais biaisé (underfitting) et plus complexe.

k-ppv fonctionne souvent très bien avec de petites valeurs de k ! Parfois, on conseille $k \approx \sqrt{n}$...

le choix de la distance

Il existe de nombreuses distances : eculidienne, Manhattan, Mahalanobis, Hamming, Jaccard, Levenshtein (pour un texte). Il n'y a pas de choix universel car la bonne distance dépend du problème. On peut pondérer les données ou pas (choix par défaut). Si aucun *a priori* n'est connu, on peut choisir distance et pondération par validation croisée.

Complexité de k-ppv

La méthode a un gros avantage : il n'y a presque pas d'apprentissage («lazy training»), mais la prédiction peut devenir coûteuse. Si n est la taille de l'échantillon, d la dimension des données et k le nombre de voisins, on a :

• Coût de l'apprentissage : $\mathcal{O}(nd)$.
• Calcul des distances : $\mathcal{O}(nd)$.
• Trouver les k-ppv: $\mathcal{O}(n \ln n)$.
• Pour chaque nouveau point prédire : $\mathcal{O}(nd)$.

La principale limite de la méthode est qu'en grande dimension d, tous les points deviennent équidistants (malédiction de la dimension). Vous pouvez consulter l'excellent cours de Ricco Rakotomalala :
https://cours-machine-learning.blogspot.com/.

5.3 Les arbres de décision et de régression

5.3.1 La méthode CART : Classification And Regression Tree

Le principe des arbres de décision est le même que l'algorithme des k-ppv : diviser pour régner. La différence entre les deux algorithmes vient de la façon de construire la partition ; la partition engendrée par un arbre de décision est basée non seulement sur les variables explicatives X_i mais également sur les étiquettes observées Y_i. Il existe plusieurs façons de construire un arbre à partir d'un échantillon, les plus connues sont les algorithmes [Breiman et al., 1984] CART et C4.5 (proposé par Quinlan en 1993).

Un graphe est un ensemble de nœuds reliés par des arêtes. Un arbre est un graphe sans cycle.

La méthode CART produit des arbres qui peuvent être utilisés en classification comme en régression, l'objectif étant d'expliquer une réponse (variable qualitative ou quantitative) à l'aide d'autres variables. On construit un arbre à l'aide de divisions successives des individus de la base d'apprentissage en deux sous-ensembles homogènes (les nœuds) par rapport à une variable à expliquer. On peut voir la segmentation par arbre comme une approche non-paramétrique et non linéaire de l'analyse discriminante.

Dans un arbre de décision, chaque nœud correspond à un sous-ensemble $A \subset \mathbb{X}$ et un test statistique T qui donne le critère de segmentation (ou d'impureté) et est appliqué sur les variables explicatives $x \in \mathbb{X}$. Si le test peut donner K résultats $1, ..., K$, alors le nœud (A, T) donne naissance à k nœuds fils, tel que l'ensemble A_k associé au k^e fils est $A_k = \{x \in A : T(x) = k\}$. Très souvent, on se limite à des tests binaires ($k = 2$) et donc à des arbres binaires (Fig. 5.8).

L'algorithme est initialisé à la racine correspondant à $A = \mathbb{X}$, puis on itère le procédé de segmentation jusqu'à ce qu'un critère d'arrêt se produise. Les feuilles de l'arbre (nœuds terminaux) forment alors une partition de $\mathbb{X}$ en classes homogènes et distinctes, relativement à la variable Y à expliquer. Finalement, des branches sont élaguées si elles ne dégradent pas trop le taux d'erreur de l'arbre.

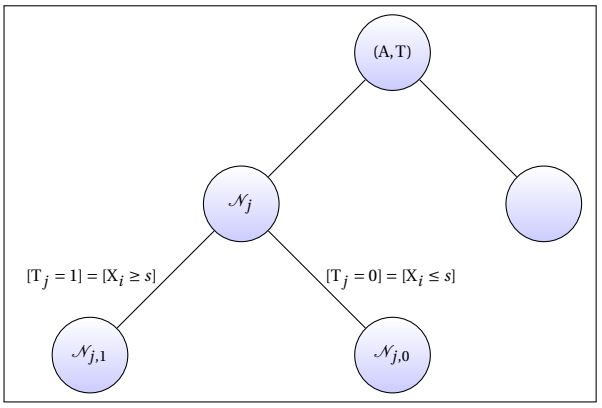


Figure 5.8 – Exemple d'arbre avec 5 nœuds et 4 arêtes. La racine A correspond à l'ensemble $\mathbb{X}$. À un nœud donné $\mathcal{N}_j$ de profondeur j est affecté un sous-ensemble de $\mathbb{X}$. Le test courant T_j est effectué sur les observations contenues dans $\mathcal{N}_j$, qui sont réparties dans les deux nœuds fils $\mathcal{N}_{j,0}$ et $\mathcal{N}_{j,0}$ en fonction du résultat de T_j.

L'algorithme CART de Breiman [Breiman et al., 1984] fournit des solutions sous forme graphique, faciles à construire, à interpréter et qui traitent les variables quantitatives ou qualitatives. Le déroulement de l'algorithme est le suivant :

Algorithm 1: CART algorithm

```
Input: Arbre = nœud racine
// Expansion
for chaque nœud n de Arbre do
    if n ≠ condition d'arrêt then
        Choisir critère de segmentation T
        Créer les nœuds fils
        Maj : Arbre = Arbre ∪ nœuds fils
    end
end
// Élagage
for chaque nœud n de Arbre do
    if n = condition d'élagage then
        Maj : Arbre = Arbre − nœuds fils et descendants
    end
end
```

Différentes implémentations sont possibles selon les critères de segmentation choisis, le critère d'arrêt ou le critère d'élagage.

Une fois l'arbre construit, la règle de classification consiste simplement à parcourir l'arbre depuis la racine :

- Pour un $x \in \mathbb{X}$, on détermine la feuille (nœud terminal) qui le contient en parcourant l'arbre de haut en bas.
- En classification, on affecte à x l'étiquette y correspondant à la classe majoritairement représentée par les exemples x_i de cette feuille.

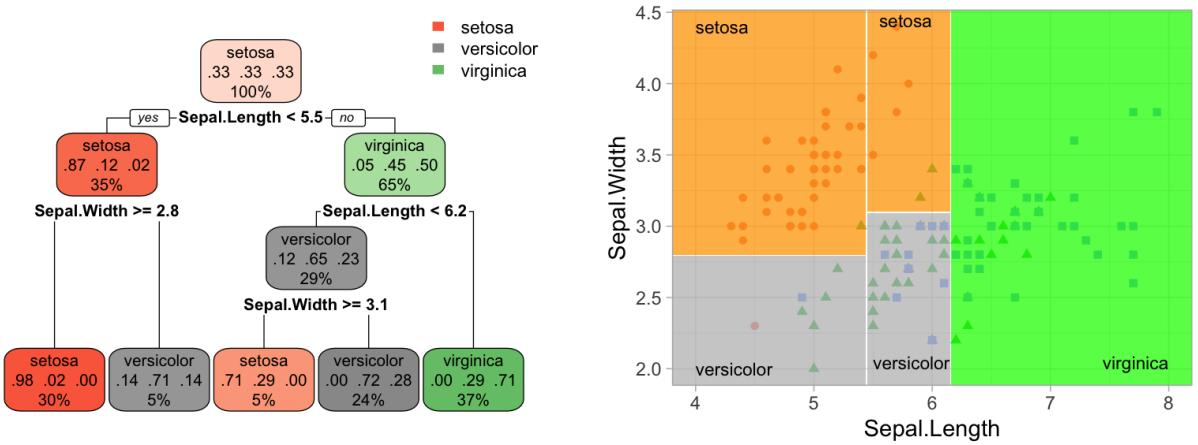


Figure 5.9 – Algorithme CART appliqué à la base de données des iris de Fisher. À gauche, l'arbre de décision construit sur les variables longueur et largeur de la sépale. À droite, la partition de l'ensemble $\mathbb{X} = \mathbb{R}^2$ en fonction des critères. Les frontières des classes sont nécessairement parallèles aux axes du repère.

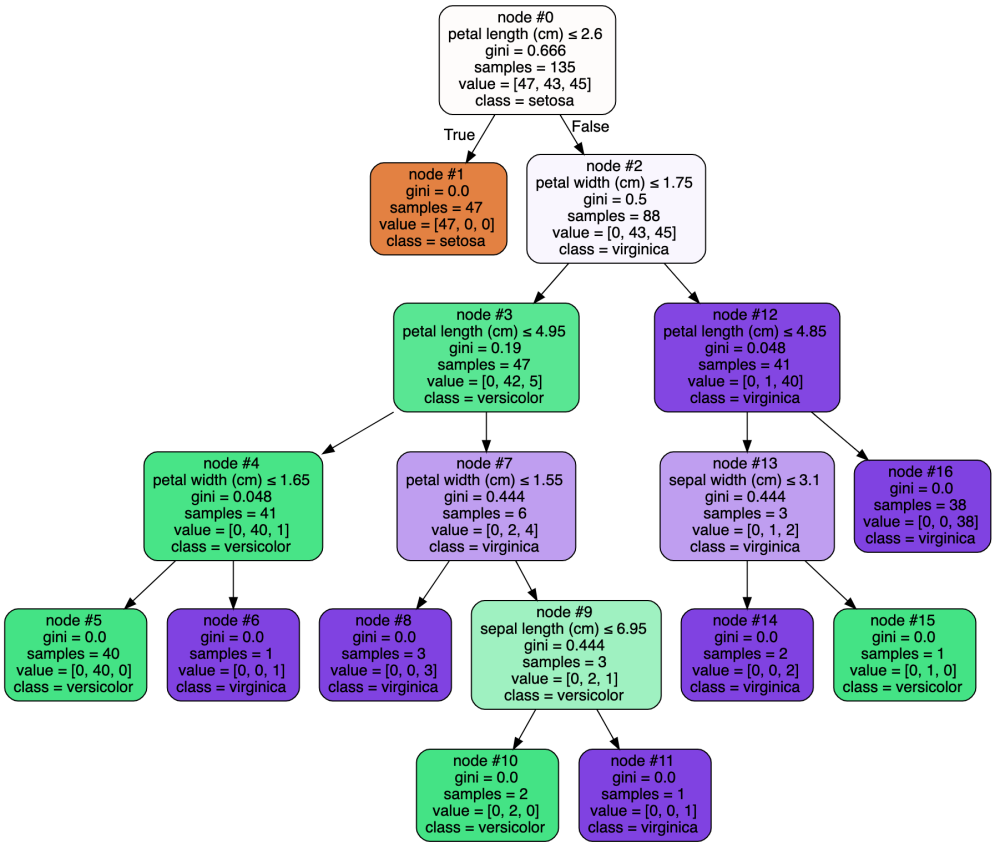


Figure 5.10 – Arbre de décision issu de la base des Iris de Fisher avec des critères de segmentation portant sur les 4 variables.

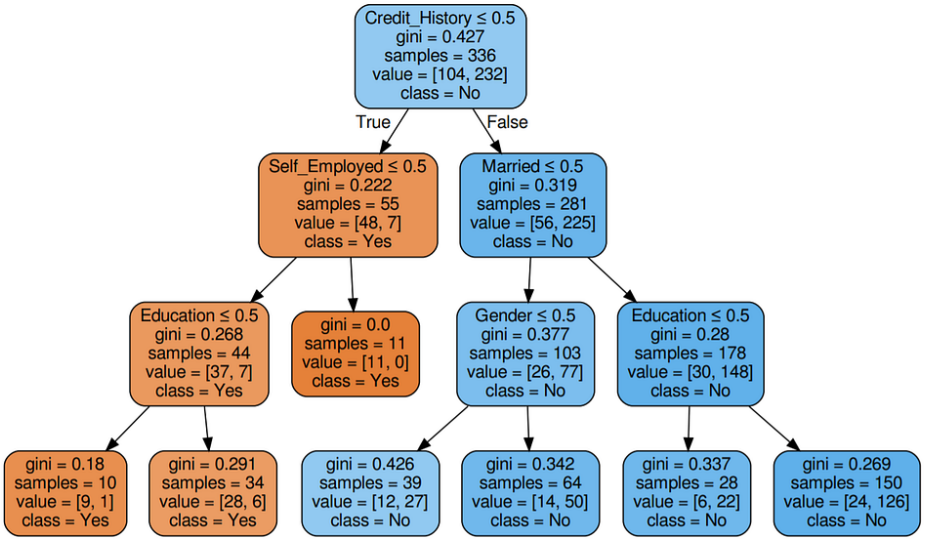


Figure 5.11 – Arbre de décision issu d'une base de données de défauts de paiement contenant 336 observations. Les variables utiliseées pour la segmentation sont : "Credit History", "Self Employed", "Married", "Education", "Gender".

- En régression, on affecte à x la moyenne des étiquettes y_i correspondant aux exemples x_i de la feuille.

Les figures ci-après (Fig. 5.10 et Fig. 5.11) illustrent des arbres CART pour différents jeux de données traditionnels.

Le critère d'arrêt vérifie l'une des conditions suivantes sur un seuil donné :

- Chaque feuille satisfait un seuil d'homogénéïté.
- La profondeur de l'arbre dépasse le seuil.
- Le nombre de feuilles dépasse le seuil.
- L'effectif du nœud est inférieur au seuil.

La segmentation peut produire de bonnes performances sur l'ensemble des données d'entrainement, mais est susceptible de provoquer du sur-apprentissage si l'arbre est trop complexe. Un arbre avec moins de branches aura une variance plus faible en contrepartie d'un biais un peu plus fort. Une stratégie pour simplifier l'arbre est d'élaguer des branches qui ne contribuent pas trop à augmenter le risque estimé, c'est le processus d'élagage (« pruning »).

5.3.2 Les différents critères de segmentation

La segmentation a pour objectif de partager les individus en classes homogènes relativement à la variable à expliquer. On quantifie l'homogénéïté par l'indice de Gini ou l'entropie pour les tâches de classification, la variance pour les tâches de régression. Ces grandeurs doivent être nulles si le segment (le découpage en deux sous-classes) est homogène et maximales lorsque les valeurs de Y sont très dispersées. On arrête de segmenter quand toutes les classes sont homogènes ou bien quand elles contiennent moins d'observations qu'un seuil fixé.

Pour la classification

Définition 8 — Indice de Gini

L'indice de Gini d'un sous-ensemble A de $\mathbb{X}$ est défini par :

$$\forall A \subset \mathbb{X},\ G(A) = 1 - \overline{Y}_A^2 - (1 - \overline{Y}_A)^2 \tag{27}$$

Remarques.

• $G(A) = 0 \iff \overline{Y}_A = 0$ ou 1.

• Un nœud sera de bonne qualité (on dit homogène, pur) si une très grande majorité des étiquettes des exemples associés à ce nœud sont identiques. Il est alors très discriminant et G(A) est presque nul.

• Pour évaluer la qualité d'un critère de segmentation, on calcule le gain d'homogénéité lorsque A est segmenté en A_1 et A_2 :

$$I_G(A_1, A_2) = G(A) - qG(A_1) - (1-q)G(A_2) \tag{28}$$

avec $q = N_{A_1}/A_{A_2}$ proportion des $x_i \in A$ qui se dirigent vers A_1. CART choisit la partition qui maximise I_G à chaque étape.

Définition 9 — Entropie au sens de Shannon

• La quantité d'information de Shannon apportée par la réalisation d'un évènement A est

$$\mathbb{I}(A) = -\mathbb{P}(A)\log_2 \mathbb{P}(A). \tag{29}$$

Elle mesure la vraisemblance de cet évènement.

• L'entropie d'une variable aléatoire mesure l'information moyenne sur l'ensemble des réalisations possibles. C'est le nombre moyen de questions binaires que l'on doit poser pour déterminer la valeur exacte de la variable aléatoire :

$$H(X) = -\mathbb{E}[\log_2 \mathbb{P}(X)] = -\sum_{i=1}^{n} p_i \log_2 p_i \tag{30}$$

On utilise l'entropie comme critère de segmentation en calculant l'entropie de la mesure de probabilité empirique uniforme créée par la partition des données (p_i proportion de 1 (ou 0) dans chaque classe).

L'impureté d'un nœud $\mathcal{N}$ (contenant N observations) se calcule, en classification, finalement par une même formule

$$I(\mathcal{N}) = \sum_{i=1}^{N} f(p_i(\mathcal{N})), \tag{31}$$

où $p_i(\mathcal{N})$ représente la proportion d'observations de la classe i dans le nœud et f est une fonction concave de $[0,1]$ dans $\mathbb{R}^+$ telle que $f(0) = f(1) = 0$.

- Pour l'indice de Gini : $f(p) = p(1-p)$.
- Pour l'entropie : $f(p) = -pn\ln p$.

Pour la régression

On rappelle que dans une tâche de régression, la valeur affectée à un nœud est la moyenne des observations appartenant à ce nœud.

L'hétérogénéïté est mesurée par la variance du nœud. Si l'on note $\mathcal{N}$ le nœud et $\overline{y}$ la moyenne des valeurs y_i des observations se trouvant dans le nœud correspondant,

$$\mathbb{V}(\mathcal{N}) = \frac{1}{|\mathcal{N}|} \sum_{i:x_i \in \mathcal{N}} (y_i - \overline{y})^2. \tag{32}$$

La segmentation privilégie le découpage en deux nœuds homogènes dont la variance sera la plus faible possible.

Une opération de segmentation est dite admissible si les deux nœuds fils contiennent au moins une observation.

La segmentation s'arrête lorsqu'un critère d'arrêt est vérifié pour tous les nœuds. Ce critère peut-être défini par une ou plusieurs des conditions suivantes sur un seuil donné :

- Chaque nœud satisfait un seuil d'homogénéïté.
- La profondeur de l'arbre dépasse un seuil.
- Le nombre de feuilles dépasse un seuil.
- L'effectif des nœuds est inférieur un seuil ou bien il n'existe plus de segmentation admissible possible.

La segmentation peut produire de bonnes performances sur l'ensemble des données d'entrainement, mais est susceptible de provoquer du surapprentissage si l'arbre est trop complexe. Un arbre avec moins de branches aura une variance plus faible en contrepartie d'un biais un peu plus fort.

Un stratégie pour simplifier l'arbre est d'élaguer des branches qui ne contribuent pas trop à augmenter le risque estimé, c'est le processus d'élagage ("pruning") que nous allons détailler maintenant.

5.3.3 Les différents critères d'élagage de CART.

On élague un arbre pour éviter le surapprentissage et garder un modèle simple. La complexité d'un arbre est donnée par sa dimension de Vapnik (hors programme, mais une courte présentation est donnée en annexe), son nombre de coupures ou sa profondeur. Le résultat de la procédure d'élagage est un sous-arbre optimal au sens d'un critère pénalisé, c'est à dire qu'il s'agit de rechercher le meilleur compromis entre un arbre très complet, détaillé, fortement dépendant des observations et un arbre grossier, trop robuste aux observations. On élague lorsque l'augmentation de complexité n'est plus compensée par la diminution de la variance (déviance).

Tester tous les sous-arbres est trop coûteux en temps de calcul. Breiman propose de se limiter à une suite de sous-arbres emboités, de taille raisonnable, construit par élagage successif. On Choisit alors un arbre de la suite par minimisation d'un risque d'ajustement.

Soit T un arbre dont les noeuds terminaux sont les $\mathcal{N}_m$. Deux risques d'ajustement classiques sont proposés, respectivement en régression et classification :

$$\mathrm{R}_m(r) = \frac{1}{\mathrm{N}_m} \sum_{i:x_i \in \mathcal{N}_m} (y_i - \overline{y}_m)^2 \tag{33}$$

$$\mathrm{R}_m(c) = \frac{1}{\mathrm{N}_m} \sum_{i:x_i \in \mathcal{N}_m} \mathbb{1}_{[y_i \neq \overline{y}_m]} \tag{34}$$

On définit ensuite un critère côut/complexité par

$$\mathrm{C}_\alpha(\mathrm{T}) = \sum_{m=1}^{\mathrm{M}} \mathrm{N}_m \mathrm{R}_m(\mathrm{T}) + m\alpha, \tag{35}$$

où $\mathrm{M} = |\mathrm{T}|$ est le nombre de feuilles de l'arbre et $\alpha \geq 0$ un paramètre d'ajustement pénalisant le nombre de feuilles. Lorsque α augmente, des feuilles sont regroupées (élaguées) dans le nœud père qui se transforme alors en feuille. C'est ainsi que les sous-arbres sont emboités les uns dans les autres.

On cherche T qui minimise $\mathrm{C}_\alpha(\mathrm{T})$ pour un α bien choisi ($\alpha = 0$ arbre entier, $\alpha = \infty$ racine uniquement). Lorsque l'on a une suite de sous-arbres, ils sont classés dans l'ordre croissant des α.

Théorème 10 de Breiman, 1984

Il existe une suite finie $0 = \alpha_0 < \ldots < \alpha_\mathrm{M}$ avec $\mathrm{M} \leq |\mathrm{T}|$ et une suite imbriquées de sous-arbres $(\mathrm{T}_{\alpha_m})_m$ avec

$$\mathrm{T} = \mathrm{T}_{\alpha_0} \subseteq \mathrm{T}_{\alpha_1} \subseteq \ldots \subseteq \mathrm{T}_{\alpha_\mathrm{M}} = \text{racine} \tag{36}$$

telle que $\forall \alpha \in [\alpha_m, \alpha_{m+1}[$,

$$\mathrm{T}_m \in \underset{\mathrm{T}_i \subset \mathrm{T}}{\arg\min}\, \mathrm{C}_\alpha(\mathrm{T}_i). \tag{37}$$

Choisir un arbre revient à choisir une valeur de α (en choisissant un risque, en l'estimant par ré-échantillonnage, puis en sélection la valeur qui minimise le risque). On peut utiliser la méthode de validation croisée pour estimer le coût de chaque arbre de la suite, ce coût étant donné par le risque quadratique en régression et l'erreur de classification en classification.

Mesure d'importance d'un arbre.

La visualisation de l'arbre peut donner une idée sur l'importance des variables, mais certaines variables possèdent une grande importance sans apparaitre explicitement dans l'arbre et il est donc difficile de quantifier l'importance juste en regardant l'arbre ! Le problème est le même que la significativité d'un régresseur dans une régression linéaire ou logistique. On définit donc une mesure d'importance d'un arbre basée sur le gain d'impureté des noeuds internes. Plus de détails seront donnée en TP.

5.3.4 Exemples en classification et régression.

Les figures Fig. 5.12 et Fig. 5.13 illustrent la méthode CART respectivement sur une tâche de classification (à nouveau les iris de Fisher) et une tâche de régression.

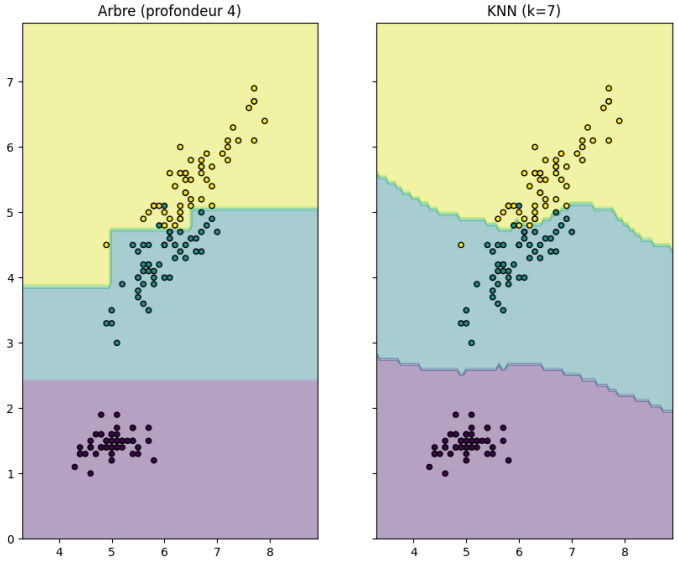


Figure 5.12 – Classification des trois espèces d'iris de Fisher par un arbre CART de profondeur 4 (à gauche) et par la méthode des k-ppv pour $k = 7$ (à droite). Noter le nombre et la position des iris mal classées dans les deux cas.

5.3.5 Remarques finales.

Les arbres CART forment une méthode de classification et de régression non linéaire (c.f. Fig. 5.14). En fonction des données, les arbres peuvent obtenir de meilleures performances (ou pas) que les méthodes linéaires.

- Les arbres sont des modèles hiérarchiques, non linéaires, itératifs.
- Ils sont adaptés aux problèmes continus, discrets, multiclasses, multimodes.
- Ils prédisent par l'intermédiaire d'une stratification (partition) de l'espace des données.
- Les arbres proposent une méthode graphique intuitive et facile à comprendre.
- Ils ne sont pas très performants.
- Ils d'une grande instabilité et peu robustes au bruit dans les données.
- Variantes diverses : CID3, C4.5, C5, CHAID, MARS, QUEST, etc.

Les inconvénients cités ci-dessus peuvent se résoudre en utilisant plusieurs arbres en même temps (une forêt...). C'est le principe de méthodes beaucoup plus robustes et efficaces, ayant de meilleures performances : les forêts aléatoires et le bagging, par exemple, dont nous parlerons dans un autre chapitre.

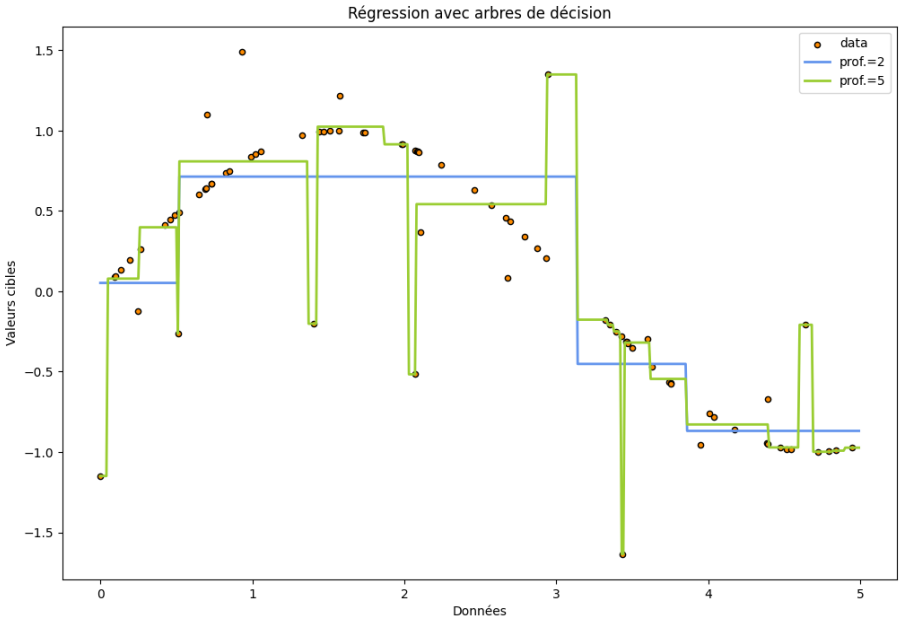


Figure 5.13 – Régression par arbre de décision. Un nuage de points (oranges) est généré à partir d'une sinusoïde perturbée par un bruit aléatoire. Il s'agit de reconstruire la courbe initiale à l'aide d'un arbre de décision de profondeur 2 (en bleu) ou de profondeur 5 (en vert)

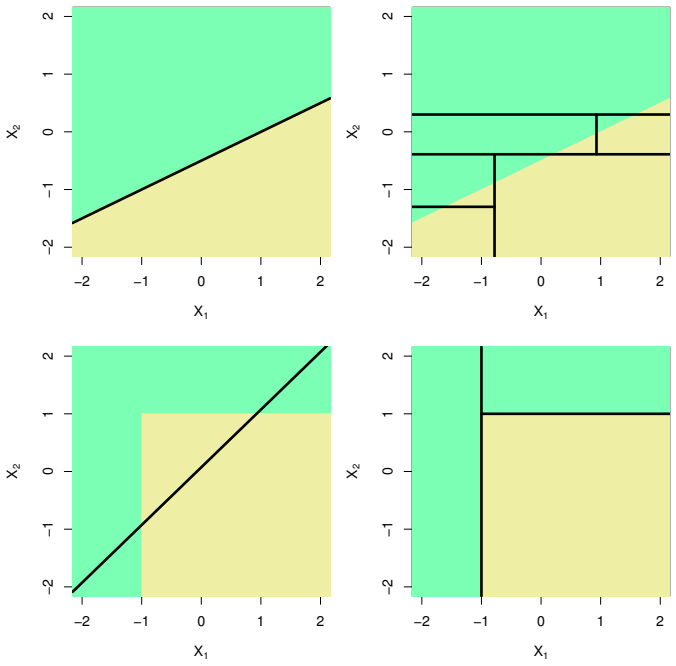


Figure 5.14 – Illustration de la non-linéarité des méthodes à base d'arbres. En haut : une tâche de classification binaire associée à deux régions linéairement séparables (une zone verte et une jaune) ; la méthode CART (colonne de droite) ne peut séparer que des régions à frontière horizontale et verticale. En bas : un classifieur linéaire ne pourra pas séparer une zone rectangulaire correctement, alors qu'un arbre y parvient. Crédit : Introduction to Machine Learning with Python. James, Witten, Hastie, Tibshirani. Figures du chapitre 8.

Chapitre 6

Méthodes à base de convexification

6.1 Introduction et problématique

La minimisation du risque empirique est difficile à mettre en œuvre à cause de la non-convexité de l'ensemble $\mathcal{G}$ et de la non-convexité de la fonction $\mathrm{R}_n(g)$. Minimiser une fonction convexe dans un ensemble convexe est un problème de programmation classique que l'on sait résoudre de façon efficace grâce à des algorithmes de programmation convexe. Pour résoudre plus efficacement la minimisation du risque empirique, on procède donc à la convexification du problème : il faut changer l'espace des classifieurs $\mathcal{G}$ pour qu'il devienne convexe et changer la forme du risque également.

Rappelons la définition d'un ensemble et d'une fonction convexe. Soit E un espace vectoriel et $\mathrm{C} \subset \mathrm{E}$. C est convexe si

$$\forall x, y \in \mathrm{C}, \text{ et } \lambda \in [0,1], \lambda x + (1-\lambda) y \in \mathrm{C}.$$

Soit $\mathrm{C} \subset \mathrm{E}$ un ensemble convexe. Une fonction $f : \mathrm{C} \longrightarrow \mathbb{R}$ est convexe si $\forall x, y \in \mathrm{C},\ \forall \lambda \in [0,1]$

$$f(\lambda x + (1-\lambda) y) \leq \lambda f(x) + (1-\lambda) f(y)$$

Une fonction convexe définie sur un ensemble convexe fermé atteint son minimum.

6.1.1 Convexification du dictionnaire

Les fonctions de prédictions binaires étaient à valeurs dans $\mathbb{Y} = \{0,1\}$. Pour simplifier les formules, on va supposer, sans perte de généralité que $\mathbb{Y} = \{-1,1\}$. On définit les nouvelles fonctions $h : \mathbb{X} \longrightarrow \mathbb{R}$ à valeurs réelles. On affecte à x l'étiquette -1 si $h(x) \leq 0$ et 1 si $h(x) > 0$: $g(x) = \mathrm{sgn}(h(x))$ où sgn est la fonction signe :

$$\mathrm{sgn}(u) = \begin{cases} 1 & \text{si } u > 0 \\ -1 & \text{si } u \leq 0 \end{cases} \tag{1}$$

Toute fonction réelle h peut alors être considérée comme prédicteur. $\mathcal{F}(\mathbb{X}, \mathbb{R})$ est convexe alors que $\mathcal{F}(\mathbb{X}, \{-1,1\})$ ne l'est pas. La perte de prédiction est donnée par

$$\begin{aligned} l(y, h(x)) &= \mathbb{1}_{[y \neq \mathrm{sgn}(h(x))]} = \mathbb{1}_{[y=1]} \mathbb{1}_{[1 \neq \mathrm{sgn}(h(x))]} + \mathbb{1}_{[y=-1]} \mathbb{1}_{[-1 \neq \mathrm{sgn}(h(x))]} \\ &= \mathbb{1}_{[y=1]} \mathbb{1}_{[h(x) \leq 0]} + \mathbb{1}_{[y=-1]} \mathbb{1}_{[h(x) > 0]} \\ &= \mathbb{1}_{[y=1]} \mathbb{1}_{[yh(x) \leq 0]} + \mathbb{1}_{[y=-1]} \mathbb{1}_{[yh(x) < 0]} \end{aligned}$$

L'ensemble $\{(y,x) : yh(x) < 0\}$ est inclut dans $\{(y,x) : yh(x) \leq 0\}$ et par conséquent l'indicatrice du premier est majorée par celle du second :

$$[yh(x) < 0] \subset [yh(x) \leq 0] \Rightarrow l(y, h(x)) \leq \mathbb{1}_{[y=1]} \mathbb{1}_{[yh(x) \leq 0]} + \mathbb{1}_{[y=-1]} \mathbb{1}_{[yh(x) \leq 0]} \leq \mathbb{1}_{[yh(x) \leq 0]}$$

De la même manière, $l(y, h(x)) \geq \mathbb{1}_{[yh(x) < 0]}$ et il est clair que si $\mathscr{H} \subset \mathcal{F}(\mathbb{X}, \mathbb{R})$ est tel que $\mathbb{P}[h(\mathrm{X}) = 0] = 0$ pour tout $h \in \mathscr{H}$, alors le risque de la fonction de prédiction $\mathrm{sgn}(h)$ est donné par

$$l(y, h(x)) = \mathbb{1}_{[yh(x) \leq 0]} = \mathbb{1}_{[0, +\infty[}(-yh(x)).$$

Ainsi,

$$\mathrm{R}(g) = \mathrm{R}(\mathrm{sgn}(h)) = \mathbb{E}\left[\mathbb{1}_{[0,+\infty[}(-\mathrm{Y}h(\mathrm{X}))\right]$$

Pour résumer, on remplace $g \in \mathcal{G}$ par $h \in \mathscr{H} \subset \mathcal{F}(\mathbb{X}, \mathbb{R})$, c'est à dire que les fonctions $g : \mathbb{X} = \mathbb{R}^d \longrightarrow \{-1,1\}$ sont remplacées par les fonctions $h : \mathbb{X} \longrightarrow \mathbb{R}$ car $\mathcal{F}(\mathbb{X}, \mathbb{R})$ est convexe.

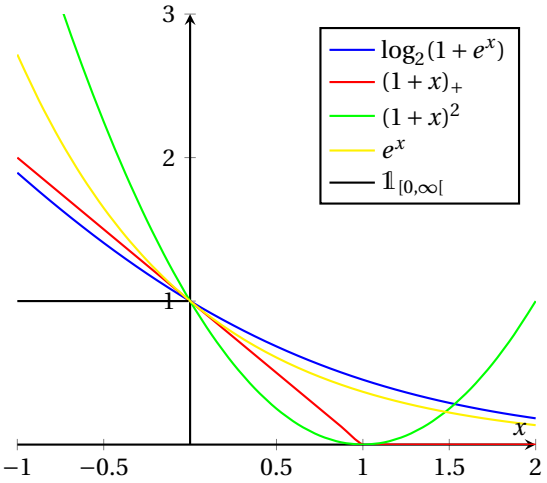


Figure 6.1 – Fonctions de perte convexe ϕ : courbes de $\phi(yh(x))$.

6.1.2 Convexification de la fonction perte

On définit le minimiseur du risque empirique convexifié par

$$\widehat{h}_n \in \operatorname*{arg\,min}_{h\in\mathscr{H}} \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}_{[0,+\infty[}(-\mathrm{Y}_i h(\mathrm{X}_i)) \tag{2}$$

$\widehat{h}_n$ appartient à $\mathscr{H}$. La fonction de coût n'est pas convexe. On remplace donc l'indicatrice $\mathbb{1}_{[0,+\infty[}$ par une fonction convexe ϕ et l'on appelle ϕ-risque du classifieur h la quantité :

$$\mathrm{A}(h) = \mathbb{E}[\phi(-\mathrm{Y}h(x))] \tag{3}$$

Le ϕ-classifieur de Bayes est le prédicteur minimisant le ϕ-risque :

$$h^\star \in \operatorname*{arg\,min}_{h\in\mathcal{F}} \mathrm{A}(h) \tag{4}$$

On appelle minimseur sur $\mathscr{H} \subset \mathcal{F}$ du ϕ-risque empirique, le prédicteur

$$\widehat{h}_n \in \operatorname*{arg\,min}_{h\in\mathscr{H}} \frac{1}{n}\sum_{i=1}^{n} \phi(-\mathrm{Y}_i h(\mathrm{X}_i)) \tag{5}$$

L'objectif est de choisir pour ϕ un majorant convexe de $\mathbb{1}_{[0,+\infty[}$ de sorte qu'un classifieur ayant un ϕ-risque faible ait également un faible risque de classification. Les fonctions traditionnellement utilisées sont les suivantes :

Perte charnière	$\phi(x) = (1+x)_+$
Perte de Boosting	$\phi(x) = e^x$
Perte logistique	$\phi(x) = \log_2(1+e^x)$
Perte quadratique	$\phi(x) = (1+x)^2$

6.1.3 Consistance du minimiseur du ϕ-risque

Théorème 11

Soit $\phi : \mathbb{R} \longrightarrow \mathbb{R}$ une fonction convexe telle que $\forall x \in \mathbb{R}$,

$$x(\phi(x) - \phi(-x)) \geq 0. \tag{6}$$

Soit $\psi : [0,1] \longrightarrow \mathbb{R}$ la fonction définie par $\psi(x) =$
Le minimiseur $\widehat{g}_n = \widehat{g}_n(.,\mathscr{A})$ du risque empirique sur le dictionnaire $\mathcal{G}(\mathscr{A})$ est

Soit $\psi : [0,1] \longrightarrow \mathbb{R}$ la fonction définie par

$$\psi(p) = \inf_{u \in \mathbb{R}} \left(p\phi(-u) + (1-p)\phi(u) \right) \tag{7}$$

S'il existe $\gamma \in [0,1]$ et $c > 0$ tels que $\forall p \in [0,1]$,

$$|1-2p| \leq c \left(\phi(0) - \psi(p) \right)^{\gamma}, \tag{8}$$

alors pour toute fonction de prédiction h,

$$\mathrm{R}[\mathrm{sgn}(h)] - \mathrm{R}[g^{\star}] \leq c \left(\mathrm{A}(h) - \mathrm{A}(h^{\star}) \right)^{\gamma} \tag{9}$$

6.2 Espaces de Hilbert et méthodes à noyaux

Dans cette section, nous résumons les résultats essentiels relatifs aux espaces de Hilbert utiles à l'introduction des machines à vecteurs de support (SVM). Même si l'exposé reste assez abstrait et qu'il ne s'agit pas de la méthode traditionnelle pour introduire les SVM, il est important de comprendre ces notions pour bien comprendre d'où viennent les SVM.

Définition 12

Un espace de Hilbert $\mathscr{H}$ est un espace vectoriel normé muni d'un produit scalaire qui en fait un espace métrique complet:

- $\mathscr{H}$ est un espace vectoriel normé.
- Il existe un produit scalaire $\langle .,. \rangle$ vérifiant $\forall x \in \mathscr{H}$, $\langle x, x \rangle = ||x||^2$.
- Toute suite de Cauchy $(x_n)_n$ de $\mathscr{H}$ converge pour la norme $||.||$.

Une suite de Cauchy est une suite $(x_n)_n$ vérifiant

$$\lim_{n \to +\infty} \sup_{m > n} ||x_n - x_m|| = 0. \tag{10}$$

Dans les espaces vectoriels normés de dimension finie, une suite vérifiant la condition de Cauchy est toujours convergente. La réciproque n'est pas toujours vraie et constitue la propriété de complétude. Dans notre cadre d'étude, cette notion n'est pas importante car nous travaillerons (presque) toujours en dimension finie. On peut donc oublier la condition de complétude.

Exemples :

- L'espace euclidien $\mathbb{R}^n$ muni de $\langle x, y \rangle = x^{\mathrm{T}} y$.
- L'ensemble des suites de carré sommable $l^2(\mathbb{N})$ muni de $\langle x, y \rangle = \sum_{k=1}^{+\infty} x_k y_k$.
- L'ensemble des fonctions de carré intégrable $\mathrm{L}^2(\mathbb{R})$. muni de $\langle f, g \rangle = \int_{-\infty}^{+\infty} f(x) g(x) dx$.
- Idem sur $\mathbb{C}$ en ajoutant un conjugué sur le second terme.

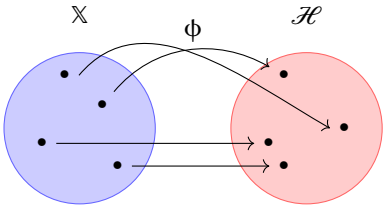


Figure 6.2 – $\mathscr{H}$ représente les données (features) issues de $\mathbb{X}$.

Les espaces de Hilbert généralisent en dimension infinie la notion d'espace euclidien ; on peut travailler de façon géométrique dans l'espace, avec des normes, des angles, des projections orthogonales.

Lorsque les données de $\mathbb{X}$ ne sont pas facilement manipulables, on les transforme en les envoyant dans un espace de Hilbert $\mathscr{H}$ appelé espace de représentation (de redescription, ou "features space"), via une fonction de représentation ϕ ("features map"). Plutôt qu'une représentation individuelle, il est souhaitable de représenter les données par couples en respectant les similarités entre objets de $\mathbb{X}$, via une fonction noyau K définie par

$$\begin{aligned} \mathrm{K} : \mathbb{X} \times \mathbb{X} &\longrightarrow \mathbb{R} \\ (x, y) &\longrightarrow \mathrm{K}(x, y) \end{aligned}$$

Un noyau K est une fonction symétrique et semi-définie positive :

- Symétrique : $\mathrm{K}(x, y) = \mathrm{K}(y, x)$.
- Semi-défini positif : $\forall x_i, x_j \in \mathbb{X}, \forall a_i, a_j \in \mathbb{R}$,

$$\sum_i \sum_j a_i a_j \mathrm{K}(x_i, x_j) \geq 0.$$

- Qui définit une matrice carrée réelle : $\big(\mathrm{K}(x_i, x_j)\big)_{i,j} \in \mathbb{R}^{n \times n}$ pour $i, j = 1, ..., n$.

On peut mélanger les deux points de vue en utilisant une fonction $\phi : \mathbb{X} \to \mathscr{H}$ et $\mathrm{K} : \mathbb{X} \times \mathbb{X} \to \mathbb{R}$:

$$\mathrm{K}(x, y) = \langle \phi(x), \phi(y) \rangle \tag{11}$$

C'est cette dernière formule qui va maintenant nous servir de définition du noyau. Si $\mathbb{X}$ possède beaucoup de paramètres (de features), $\mathscr{H}$ peut être de grande dimension, c'est même l'objectif de la fonction de représentation : travailler dans un espace où il y a beaucoup de "place", où l'on a beaucoup de degrés de liberté dans le choix de la représentation des données. En choisissant ϕ de façon adéquate, il est possible de ne pas avoir à effectuer explicitement les produits scalaires en grande dimension grâce à l'astuce du noyau (le "kernel trick"). Il est important de se souvenir que K mesure la similarité entre les données x et y; il s'agit en quelque sorte de la généralisation d'une fonction de covariance. La matrice extraite à partir de n données est une matrice de Gram et une matrice de covariance au sens statistique du terme.

Les fonctions noyaux jouent le même rôle que les fonctions périodiques dans la transformation de Fourier. On peut les décomposer en sommes de fonctions propres correspondant à des fréquences données et il existe toute une théorie similaire à l'analyse harmonique qui permet la décomposition spectrale des fonctions noyaux.

Nous admettons le résultat suivant : une fonction $\mathrm{K} : \mathbb{X} \times \mathbb{X} \longrightarrow \mathbb{R}$ est une fonction noyau si, et seulement si elle est symétrique et semi-définie positive.

Exemples de noyaux usuels :

- Vanille : $\mathrm{K}(x, y) = \langle x, y \rangle$.
- Sigmoïde : $\mathrm{K}(x, y) = \tanh(\langle x, y \rangle)$.
- Gaussien (RBF) : $\mathrm{K}(x, y) = \exp\left(-\frac{1}{2\sigma^2} ||x - y||^2\right)$.
- ReLU : $\mathrm{K}(x, y) = \min(x, y) = x - \mathrm{ReLU}(x - y) = y - \mathrm{ReLU}(y - x)$.
- Polynomial : $\mathrm{K}(x, y) = (c + \langle x, y \rangle)^p$.

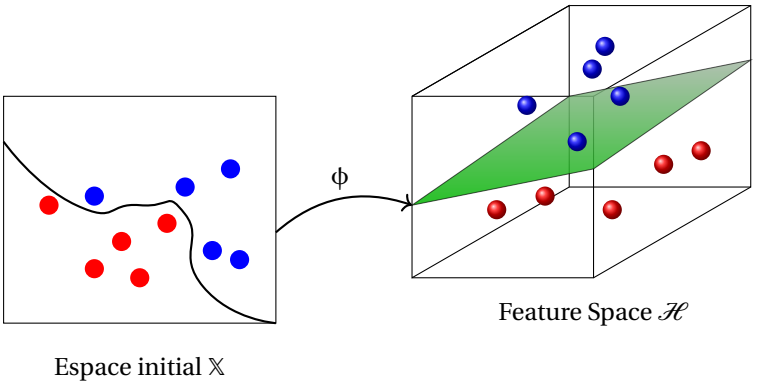


Figure 6.3 – On envoie les données initiales dans un espace (abstrait) de dimension plus grande, afin de pouvoir séparer plus facilement les classes.

- Exponentiel : $K(x,y) = \exp\left(-\gamma||x-y||\right)$.

L'espace de représentation associé au noyau gaussien est un espace de Hilbert de dimension infinie. Nous reviendrons (un peu) là dessus par la suite.

On remarque que ϕ n'apparaît pas dans les formules. C'est cette propriété qui permet de ne pas avoir à effectuer de calculs de produits scalaires explicitement.

Théorème 13 — Théorème de Mercer-Kolmogorov-Aronszajn

Si K est un noyau défini positif sur un ensemble $\mathbb{X}$, alors il existe un unique espace de Hilbert $\mathscr{H}$ et une application de représentation $\phi : \mathbb{X} \to \mathscr{H}$ tel que tels que $\forall x, y \in \mathbb{X}$,

$$K(x,y) = \langle \phi(x), \phi(y) \rangle$$

Ce théorème est dû à Mercer (1909) pour $\mathbb{X}$ compact de $\mathbb{R}^d$ et K continu par rapport à la tribu borélienne, Kolmogorov (1941) pour $\mathbb{X}$ dénombrable et K quelconque et Aronszajn (1944) pour le cas général. Il indique que tout noyau défini positif peut toujours s'écrire sous la forme du produit scalaire de deux vecteurs d'un espace de Hilbert représentant respectivement x et y.

Remarques :

- $\mathscr{H}$ est unique, mais pas ϕ.
- On espère que si $\mathscr{H}$ est de grande dimension (éventuellement infinie) les données vont devenir plus facile à classifier (c'est à dire à séparer).
- Le théorème de Mercer explicite l'espace de redescription et permet d'écrire le noyau comme somme dans une base de fonctions propres.
- Astuce du noyau (kernel trick) : on n'a pas besoin de spécifier ϕ et d'évaluer $\phi(x)$ et $\phi(y)$, seul compte $K(x,y)$.

On souhaite choisir pour $\mathscr{H}$ un espace de fonctions $h : \mathbb{X} \longrightarrow \mathbb{R}$. Cela signifie qu'une donnée $x \in \mathbb{X}$ va être représentée par une fonction de $\mathbb{X}$ dans $\mathbb{R}$. Une telle fonction reste un élément d'un espace vectoriel, donc un vecteur (en dimension infinie), mais l'idée de remplacer x par une fonction a tout de même de quoi surprendre. En fait, cette fonction (qui représente les features de x) intègre les informations de similarité entre x et l'ensemble de toutes les autres données $x' \in \mathbb{X}$. Pour la définir, on va passer par la notion de forme linéaire sur un espace.

Une forme linéaire continue (FLC) sur un espace de Hilbert est une fonction linéaire continue de $\mathscr{H}$ dans $\mathbb{R}$. Leur ensemble est un e.v. $\mathscr{H}'$ appelé dual de $\mathscr{H}$. Le théorème de représentation de Riesz nous dit en fait que $\mathscr{H} \simeq \mathscr{H}'$.

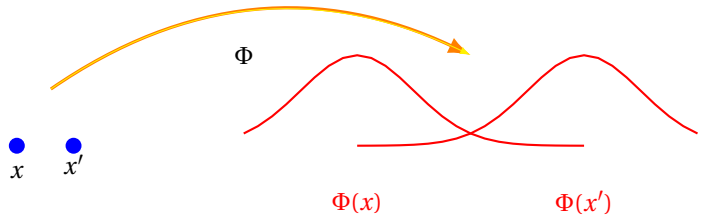


Figure 6.4 – La fonction de représentation ϕ transforme en fait un point de $\mathbb{X}$ en une fonction $K_x = \phi(x)$ de $\mathbb{X}$ dans $\mathbb{R}$ (considérée comme un élément de l'espace de Hilbert $\mathcal{H}$).

Théorème 14 — Théorème de représentation de Riesz

- $\forall L \in \mathcal{H}'$, il existe un unique $f \in \mathcal{H}$ tel que $L(h) = \langle f, h \rangle$, $\forall h \in \mathcal{H}$.
- Réciproquement, $\forall f \in \mathcal{H}$, $L(h) = \langle f, h \rangle$ définit un élément de $\mathcal{H}'$.

Dire que L est une application linéaire continue est équivalent à dire que $|L(h)| \leq \delta ||h||$, $\forall h \in \mathcal{H}$, ce qui est encore équivalent à $||L|| < \delta$, ou encore, que L est bornée.

$\forall x \in \mathbb{X}$, soit

$$L_x : \mathcal{H} \longrightarrow \mathbb{R} \tag{12}$$
$$h \longrightarrow L_x(h) = h(x) \tag{13}$$

L_x reproduit l'action de h sur x; c'est la fonction d'évaluation en x. $\forall x \in \mathbb{X}$, $L_x \in \mathcal{H}'$.

Définition 15

$\mathcal{H}$ est un espace de Hilbert à noyau reproduisant (RKHS pour Reproducing Kernel Hilbert Space) si, et seulement si, $\forall x \in \mathbb{X}$, L_x est continue.

Le théorème de Riesz implique alors que

$$\forall x \in \mathbb{X}, \exists\, ! \, K_x \in \mathcal{H} \text{ tel que } L_x(h) = \langle K_x, h \rangle = h(x), \ \forall h \in \mathcal{H}. \tag{14}$$

Autrement dit, dans un RKHS, toute forme linéaire provient d'un produit scalaire par un élément de $\mathcal{H}$. On peut alors poser

$$K(x, y) = \langle K_x, K_y \rangle = \langle \phi(x), \phi(y) \rangle . \tag{15}$$

Autrement dit,

$$K_x = \phi(x) \tag{16}$$

Ces deux formules sont très importantes. La première explicite l'astuce du noyau et explique pourquoi le produit scalaire peut être effectué implicitement sans avoir à connaître $\phi(x)$. La seconde explique pourquoi le vecteur de features $\phi(x)$ contient à lui tout seul l'ensemble des informations de similarités entre x et tous les autres points de $\mathbb{X}$. Cette information est disponible via les valeurs $K_x(x')$ que prend K_x lorsque x' parcourt $\mathbb{X}$.

La correspondance entre noyau et RKHS est précisée par les deux théorèmes suivants, qui forment en quelque sorte les réciproques du théorème de Mercer-Aronszajn.

Théorème 16

$\mathscr{H}$ est un espace de Hilbert à noyau reproduisant (RKHS) si, et seulement s'il existe un noyau reproduisant (unique) K vérifiant

$$\mathrm{K}(x,y) = \langle \mathrm{K}_x, \mathrm{K}_y \rangle, \forall x, y \in \mathbb{X}. \tag{17}$$

Tout élément de $\mathscr{H}$ s'écrit alors de façon unique $h = \sum_{i=1}^{n} \alpha_i \mathrm{K}_{x_i}$:

$$h(x) = \sum_{i=1}^{n} \alpha_i \mathrm{K}(x, x_i), \ \forall x \in \mathbb{X} = \langle h, \mathrm{K}_x \rangle. \tag{18}$$

L'espace de Hilbert associé au noyau reproduisant K est un espace de fonctions régulières. C'est en quelque sorte l'espace de fonctions minimal (dans le sens le plus économique) associé au noyau et qui joue le rôle d'espace de redescription canonique. Dans cet espace, la régularité des fonctions sera liée à la régularité du noyau. Les fonctions K_x ne sont pas issues d'un dictionnaire donné *a priori* et ne dépendent que des données $x \in \mathbb{X}$. Les fonctions analysantes K_x (les features de x) dépendent directement des (et s'adaptent aux) données $x_i \in \mathbb{X}$. La propriété "reproduisante" (18) indique que la valeur $h(x)$ est reproduite en effectuant le produit scalaire de h par la fonction analysante K_x.

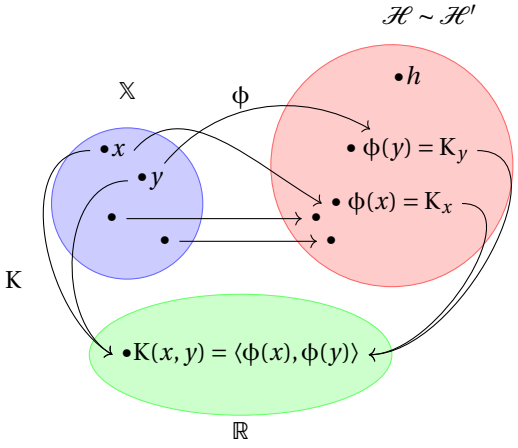


Figure 6.5 – Les espaces $\mathbb{X}$, $\mathscr{H}$, $\mathscr{H}'$ et $\mathbb{R}$, ainsi que les fonctions ϕ et K.

Théorème 17 du représentant

- Soit $\mathrm{K} : \mathbb{X} \times \mathbb{X} \longrightarrow \mathbb{R}$ un noyau symétrique, défini positif.
- Soit $\mathscr{H}$ son RKHS.
- Soit $\mathscr{D}_n = \{(x_i, y_i)\} \in (\mathbb{X} \times \mathbb{R})^n$ un échantillon.
- Soit l une fonction de perte.
- Soient $\lambda > 0$ et J une fonction réelle strictement croissante.

Alors toute solution $\widehat{h}_n$ du problème de minimisation

$$\underset{h \in \mathscr{H}}{\arg\min} \left(\frac{1}{n} \sum_{i=1}^{n} l(y_i, h(x_i)) + \lambda \mathrm{J}(||h||) \right). \tag{19}$$

s'écrit de façon unique sous la forme : $\forall x \in \mathbb{X}$,

$$\widehat{h}_n(x) = \sum_{i=1}^{n} \alpha_i \mathrm{K}(x, x_i). \tag{20}$$

λ est un paramètre de régularisation et J une fonction pénalisant h pour conserver la norme proche de zéro.

Ce théorème (admis) a été démontré par Kimeldorf et Wahba en 1970. La solution h aura une norme $||h||$ aussi petite que possible et appartiendra à un sous-espace vectoriel de dimension n connue et finie, même si $\mathscr{H}$ est de très grande dimension.

Nous avons ainsi deux interprétations différentes concernant les espaces en jeu : une interprétation géométrique dans $\mathscr{H}$, via l'astuce du noyau et une interprétation fonctionnelle via le théorème du représentant. Celui-ci établit un lien fondamental entres inductifs régularisés, réglant le compromis biais-variance, et les espaces de Hilbert RKHS.

6.3 Machines à vecteurs de support (SVM)

6.3.1 Machine à vecteurs de support : point de vue hilbertien

Revenons maintenant au minimiseur du risque empirique convexifié que nous avons défini dans l'avant-dernière section :

$$\widehat{h}_n \in \underset{h\in\mathscr{H}}{\operatorname{argmin}} \frac{1}{n}\sum_{i=1}^{n} \phi(-\mathrm{Y}_i h(\mathrm{X}_i)). \tag{21}$$

$\mathscr{H}$ a une structure d'espace de Hilbert. Pour un $t>0$, on pose

$$\widehat{h}_{\mathrm{SVM}}(t) \in \underset{h\in\mathscr{H}:||h||\leq t}{\operatorname{argmin}} \frac{1}{n}\sum_{i=1}^{n} \phi(-\mathrm{Y}_i h(\mathrm{X}_i)). \tag{22}$$

En classification, on choisit souvent $\phi(x) = (1+x)_+$, appelée perte "hing". En régression, on utilise $\phi(x) = x^2$ et l'expression $\phi(-\mathrm{Y}h(\mathrm{X}))$ est remplacée par $\phi(\mathrm{Y}-h(\mathrm{X}))$. Si $\mathscr{H}$ est un espace de Hilbert à noyau reproduisant (RKHS) alors on dit que $\widehat{h}$ est un prédicteur à base de noyau, ou une machine à vecteurs supports (SVM). L'hyperparamètre t (ou λ) est calculé par validation croisée.

On présente souvent l'expression de $\widehat{h}$ sous la forme d'un multiplicateur de Lagrange : pour un $t>0$, on pose

$$\widehat{h}_{\mathrm{SVM}}(t) \in \underset{h\in\mathscr{H}:||h||\leq t}{\operatorname{argmin}} \frac{1}{n}\sum_{i=1}^{n} \phi(-\mathrm{Y}_i h(\mathrm{X}_i)) \iff \widehat{h}_{\mathrm{SVM}}(\lambda) \in \underset{h\in\mathscr{H}}{\operatorname{argmin}} \left(\frac{1}{n}\sum_{i=1}^{n} \phi(-\mathrm{Y}_i h(\mathrm{X}_i)) + \lambda||h||^2 \right). \tag{23}$$

Comme $\mathscr{H}$ est un RKHS, la solution de

$$\underset{h\in\mathscr{H}:||h||\leq t}{\operatorname{argmin}} \frac{1}{n}\sum_{i=1}^{n} \phi(-\mathrm{Y}_i h(\mathrm{X}_i)). \tag{24}$$

est, par application du théorème du représentant, de la forme :

$$\widehat{h}_{\mathrm{SVM}}(.) = \widehat{h}_n(.) = \sum_{i=1}^{n} \alpha_i \mathrm{K}(., x_i) \tag{25}$$

où K est le noyau RKHS de l'espace $\mathscr{H}$, $\alpha_1, ...\alpha_n \in \mathbb{R}$ et $x_1, .., x_n$ les données observées de l'échantillon.

Les α_i sont facilement et rapidement déterminés par une méthode d'optimisation convexe.

6.3.2 Machine à vecteurs de supports : point de vue traditionnel

Les SVM ont été inventées par Vapnik à la fin des années 1970, mais n'ont été connues qu'à partir des années 90 (Vapnik 1995) durant lesquelles elles ont été beaucoup utilisées pour la reconnaissance de caractères manuscrits ou de façon plus générale dans des problèmes de reconnaissance de formes.

L'exposé qui suit est la présentation traditionnelle des SVM, dont nous allons voir qu'elle est équivalente à la présentation précédente. On dispose d'un échantillon $\mathscr{D}_n$ de n observations (x_k, y_k), avec $x_k \in \mathbb{R}^d$ et $y = \pm 1$, que l'on peut séparer linéairement par un hyperplan :

$$\begin{cases} w^{\mathrm{T}} x_k + b > 0 & \text{si } y_k = 1 \\ w^{\mathrm{T}} x_k + b < 0 & \text{si } y_k = -1 \end{cases}$$

L'hyperplan séparateur est caractérisé par son vecteur normal $w \in \mathbb{R}^d$ et un seuil b. Son équation est

$$w^{\mathrm{T}}x + b = 0$$

La distance euclidienne d'un point $x \in \mathbb{R}^d$ à l'hyperplan H est donnée par:

$$d(x, \mathrm{H}) = \frac{|w^{\mathrm{T}}x + b|}{||w||}.$$

C'est la formule que nous connaissance depuis le collège et qui nous donne la distance d'un point à une droite. On peut la redémontrer en quelques lignes ; le point de l'hyperplan le plus proche de x est de la forme $x + \lambda w$ car w est orthogonal à l'hyperplan. Cette projection appartient à l'hyperplan, elle vérifie donc son équation : $w^{\mathrm{T}}(x + \lambda w) + b = 0$. Ainsi, $\lambda = -(b + w^{\mathrm{T}}x)/||w||^2$ et la distance est égale à $||x + \lambda w - x|| = |\lambda|.||w|| = |w^{\mathrm{T}}x + b|/||w||$. La norme $||.||$ est la norme euclidienne et w et x sont des vecteurs de $\mathbb{R}^d$.

La plus petite distance entre les observations et un hyperplan séparateur (w, b) est

$$\frac{1}{||w||} \min_{i=1}^{n} y_i(w^{\mathrm{T}}x_i + b).$$

Cette distance est la plus petite possible lorsqu'on se trouve sur l'un des deux hyperplans frontières, c'est à dire lorsque $w^{\mathrm{T}}x_i + b = \pm 1$ et donc lorsque $y_i(w^{\mathrm{T}}x_i + b) = 1$. Ainsi, le minimum vaut 1 et la distance minimale vaut $1/||w||$.

Le double de ce minimum s'appelle la marge. La marge est donc la largeur du tube séparant les deux hyperplans frontières et dans lequel ne se trouve aucun point des deux classes (c.f. fig. 6.6). L'hyperplan (séparateur) de plus grande marge a pour paramètres (w, b) les solutions de

$$\arg\max_{w,b} \left(\frac{1}{||w||} \min_{i=1}^{n} y_i(w^{\mathrm{T}}x_i + b) \right)$$

Les observations qui réalisent le minimum sont appelées les vecteurs de support. Les séparateurs à vaste marge (SVM) sont des machines à vecteurs de support au sens défini précédemment.

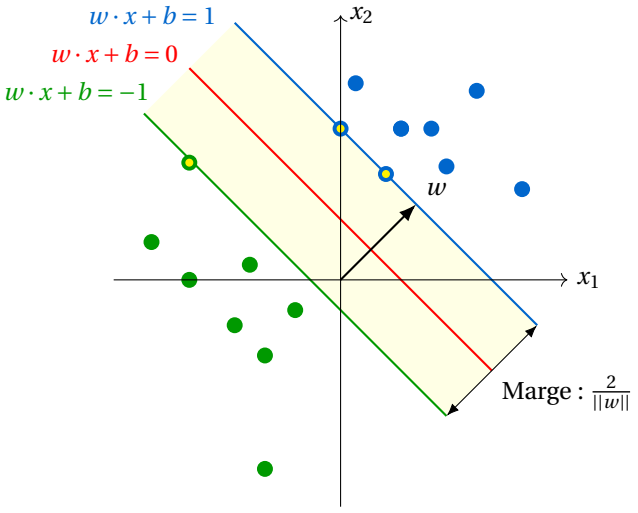


Figure 6.6 – Deux classes d'observations (en bleu et vert) linéairement séparables. En rouge, l'hyperplan séparateur de plus grande marge, en vert et bleu les hyperplans frontières de chaque classe avec les vecteurs supports (dont le centre est de couleur jaune). La largeur du tube coloriée en jaune représente la marge.

La marge vaut $\gamma = 2/||w||$. Maximiser γ est équivalent à minimiser $||w||$. La condition de séparation des classes est, $\forall i = 1, ..., n$:

$$\mathrm{sgn}(h(x_i)) = y_i \iff y_i h(x_i) \geq 1 \iff y_i(w^{\mathrm{T}}x_i + b) \geq 1$$

Ceci conduit au problème d'optimisation (primal) suivant :

$$\begin{cases} \min_{w,b} ||w||^2/2 \\ \text{s.c. } y_i(w^{\mathrm{T}}x_i + b) \geq 1 \ \forall i = 1, ..., n. \end{cases} \tag{26}$$

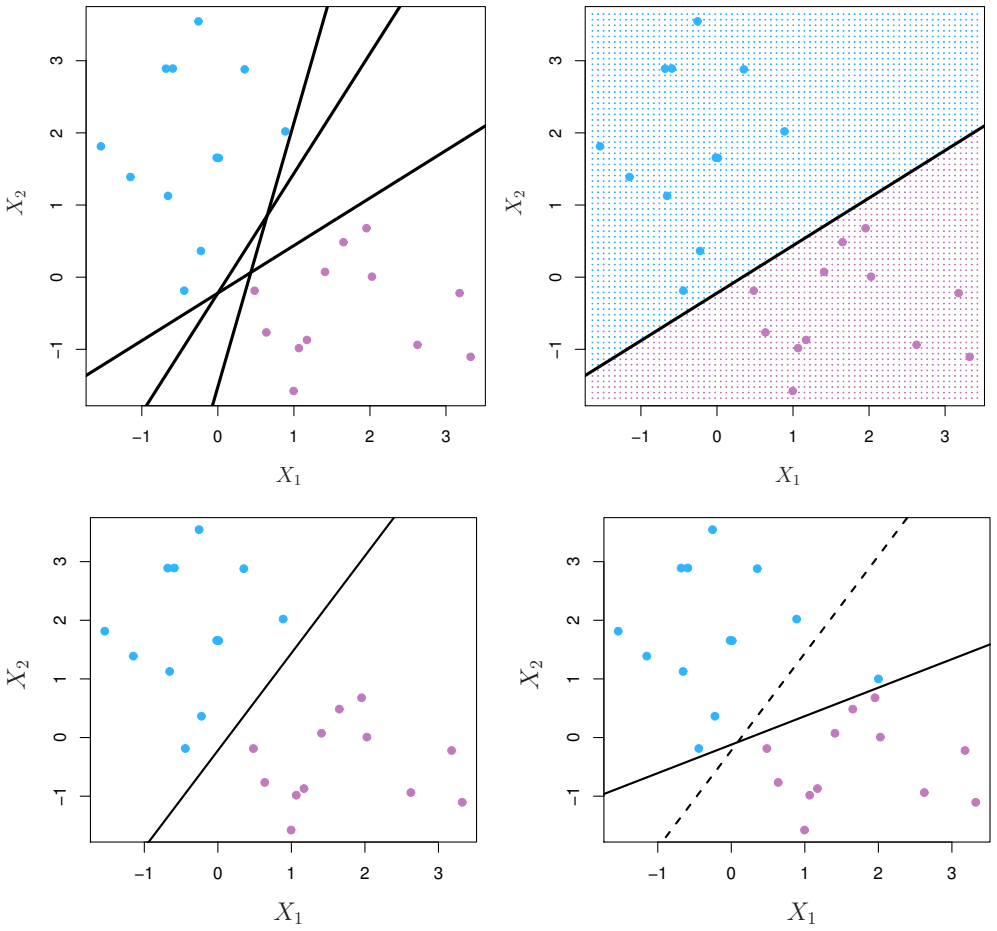


Figure 6.7 – Deux classes d'observations (bleu et rouge) mesurées par l'intermédiaire de deux variables x_1 et x_2. À gauche, trois exemples de plans séparateurs. À droite, le plan séparateur optimal, le plus éloigné des deux nuages. Crédit : Introduction to Machine Learning with Python. James, Witten, Hastie, Tibshirani. Figures du chapitre 9.

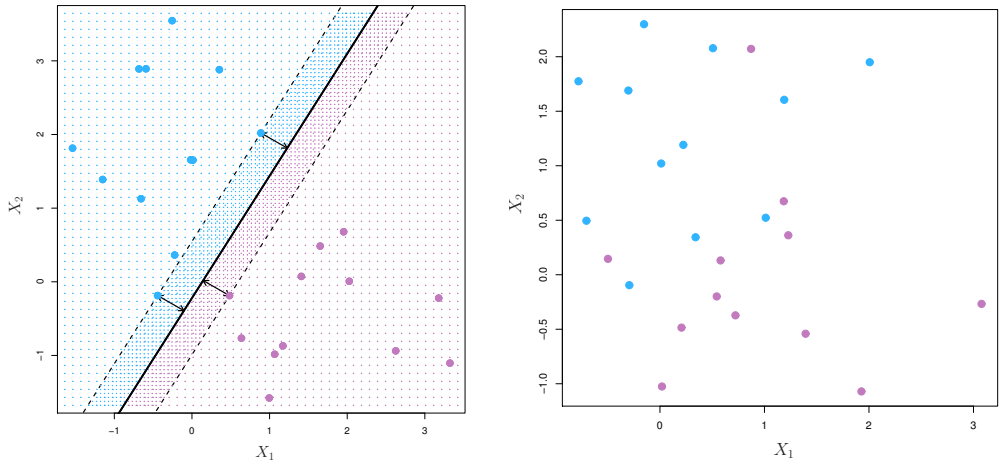


Figure 6.8 – Deux classes d'observations (bleu et rouge) mesurées par l'intermédiaire de deux variables x_1 et x_2. À gauche, le plan séparateur de plus grande marge avec les points les vecteurs de support à la frontière. À droite, un exemple de nuages non linéairement séparable à cause d'un point rouge ajouté par rapport au nuage précédent (trouver où...). Crédit : Introduction to Machine Learning with Python. James, Witten, Hastie, Tibshirani. Figures du chapitre 9.

Le facteur 1/2 sert uniquement à simplifier les calculs et ne change pas le résultat. Il s'agit d'un problème d'optimisation convexe que l'on peut résoudre en définition le lagrangien, puis en vérifiant les conditions de Karush-Kuhn-Tucker (KKT). On en déduit alors un problème dual, plus facile à résoudre.

On forme le lagrangien du problème :

$$\mathrm{L}(w,b,\lambda) = \frac{1}{2}||w||^2 - \sum_{i=1}^{n} \alpha_i \left[y_i (w^{\mathrm{T}} x_i + b) - 1 \right]. \tag{27}$$

Les α_i sont les multiplicateurs de Lagrange et les contraintes sont données par les n équations $-y_i(x^{\mathrm{T}} x_i + b) - 1$. Les conditions KKT sont nécessaires et suffisantes pour avoir une solution au problème car la fonction objectif et les contraintes sont des fonctions convexes et différentiables. Le couple optimal $(w^\star, b^\star)$ est solution du primal si, et seulement s'il existe $\alpha^\star \in \mathbb{R}^n$ tel que :

$$\forall i = 1,...,n,\ \alpha_i^\star \left[y_i (w^{\star\mathrm{T}} x_i + b^\star) - 1 \right] = 0 \tag{28}$$

et $\alpha^\star$ doit maximiser $\mathrm{L}(w^\star, b^\star, \alpha)$ sous les contraintes

$$\frac{\partial \mathrm{L}}{\partial w} = 0 \text{ et } \frac{\partial \mathrm{L}}{\partial b} = 0. \tag{29}$$

Ceci implique

$$w^\star = \sum_{i=1}^{n} \alpha_i^\star y_i x_i \text{ et } \sum_{i=1}^{n} \alpha_i y_i = 0. \tag{30}$$

En injectant ces équations dans l'expression du lagrangien, il vient

$$\mathrm{L}(w^\star, b^\star, \alpha) = \frac{1}{2} \left|\left| \sum_{i=1}^{n} \alpha_i y_i x_i \right|\right|^2 - \sum_{i=1}^{n} \alpha_i \left[y_i \left((\sum_{j=1}^{n} \alpha_j y_j x_j)^{\mathrm{T}} x_i + b^\star \right) - 1 \right] \tag{31}$$

Nous avons

$$\left|\left| \sum_{i=1}^{n} \alpha_i y_i x_i \right|\right|^2 = \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j x_i^{\mathrm{T}} x_j \tag{32}$$

et

$$\sum_{i=1}^{n} \alpha_i \left[y_i \left((\sum_{j=1}^{n} \alpha_j y_j x_j)^{\mathrm{T}} x_i + b^\star \right) - 1 \right] = \sum_{i,j} \alpha_i \alpha_j y_i y_j x_i^{\mathrm{T}} x_j + b \sum_{i=1}^{n} \alpha_i y_i - \sum_{i=1}^{n} \alpha_i. \tag{33}$$

Mais $\sum_i \alpha_i y_i = 0$, on a donc :

$$\mathrm{L}(w^\star, b^\star, \alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j x_i^{\mathrm{T}} x_j. \tag{34}$$

On en déduit le problème d'optimisation (dual) sous la forme suivante :

$$\begin{cases} \underset{\alpha \in \mathbb{R}_+^n}{\arg\max} \left(\sum_{i=1}^{n} \alpha_i - \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle / 2 \right) \\ \text{s.c. } \alpha_i \geq 0\ \forall i = 1,...,n, \ ; \ \sum_{i=1}^{n} \alpha_i y_i = 0. \end{cases} \tag{35}$$

On obtient d'abord les $\alpha_i^\star$, on en déduit $w^\star$ et $b^\star$ qui donnent l'hyperplan optimal:

$$w^\star = \sum_{i=1}^{n} \alpha_i^\star y_i x_i$$

$$\widehat{h}_n(x) = h^\star(x) = \sum_{i=1}^{n} \alpha_i^\star y_i \langle x_i, x \rangle + b^\star$$

Les données de l'échantillon correspondant à $\alpha_i^\star > 0$ sont les vecteurs support et sont les seuls à contribuer à la solution. En effet, à l'optimum, soit $w^{\star\mathrm{T}} x_i + b = y_i$ et en ce cas l'observation x_i est un vecteur support, soit $\alpha_i^\star = 0$ et ce terme n'apparaît pas dans la somme. L'hyperplan séparateur optimal ne dépend donc que des vecteurs supports. Ceci explique que les SVM ne souffrent pas du problème du fléau de la dimension.

Si les données ne sont pas linéairement séparables (c'est la grande majorité des cas), on généralise par une perte convexe ϕ et on re-écrit le problème d'optimisation :

$$\min_{w,b}\left(||w||^2 + \frac{c}{n}\sum_{i=1}^{n}\phi(-y_i h(x_i))\right) \Longleftrightarrow \min_{w,b}\left(\frac{1}{n}\sum_{i=1}^{n}\phi(-y_i h(x_i)) + \lambda||w||^2\right)$$

avec $\lambda = 1/2c$ qui est un paramètre de régularisation.

Dans le problème dual, $\langle x, y\rangle$ se transforme en $K(x,y) = \langle\phi(x),\phi(y)\rangle$ et... on retrouve le premier point de vue relatif aux espaces de Hilbert.

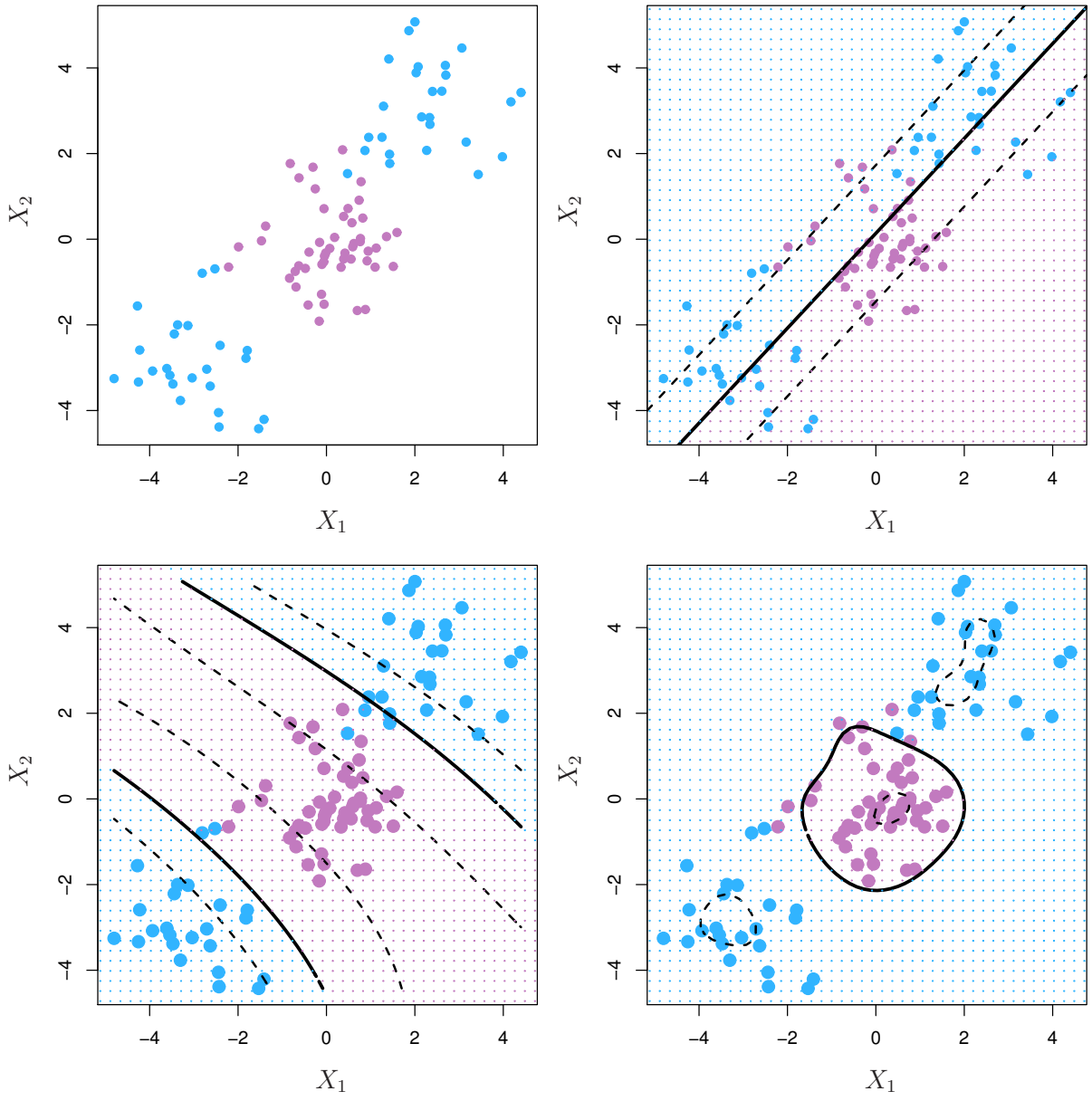


Figure 6.9 – Deux classes d'observations (bleu et rouge) mesurées par l'intermédiaire de deux variables x_1 et x_2. En haut, les données ne sont clairement plus linéairement séparables. En bas, après un changement de variables de x en $\phi(x)$ par une fonction non linéaire, il est à nouveau possible de séparer les deux nuages. En bas à gauche, ϕ est une fonction polynomiale de degré 3. En bas à droite, ϕ est une fonction gaussienne radiale. Crédit : Introduction to Machine Learning with Pyton. James, Witten, Hastie, Tibshirani. Figures du chapitre 9.

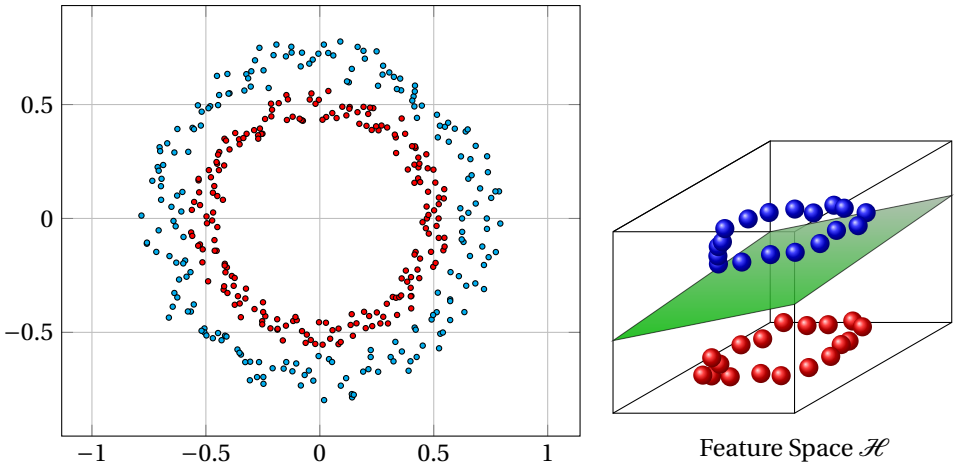


Figure 6.10 – Données non linéairement séparables en dimension 2, mais dont on se doute qu'elles sont à symétrie circulaire. En dimension 3, elles correspondent à deux nuages de hauteurs différentes qui sont linéairement séparables. $\phi(x,y) = (x,y,x^2+y^2)$ et $K(x,y) = \langle \phi(x), \phi(y) \rangle = x^T y + ||x||^2 ||y||^2$.

6.3.3 Notion de marge souple de variables d'écart

Un cas particulier presque linéaire est donné par le problème à marge souple : on autorise quelques écarts à l'hyperplan séparateur en introduisant une variable d'ajustement ("slack variable") ou variable ressort, d'écart, souple, molle, etc.) :

$$\xi_i = \left(1 - y_i(wx_i + b)\right)_+, \ i = 1, ..., n.$$

$$\begin{cases} \underset{w,b,\xi}{\arg\min} \left(\frac{1}{2}||w||^2 + c\sum_{i=1}^{n} \xi_i \right) \\ \text{s.c. } \xi_i \geq \left(1 - y_i(wx_i + b)\right)_+, \ i = 1, ..., n. \end{cases}$$

ξ_i représente la distance du mauvais côté du plan frontière qu'on s'autorise dépasser (mais que l'on souhaite la plus petite possible) pour une donnée x_i (c.f. fig. 6.11). On ajoute alors un terme de régularisation à la fonction à minimiser, avec une constante de pénalisation c qui va réaliser un compromis entre la largeur de la marge et les écarts que l'on s'autorise. Lorsque c est grand, on pénalise fortement les ξ_i (peu de souplesse). Lorsque c est petit : on peut s'écarter sensiblement de l'hyperplan.

Dans le probème dual, les ξ_i disparaissent et la seule différence par rapport au problème précédent est la condition $\alpha_i \leq c$. L'introduction des variables d'écart a donc juste ajouter au problème une borne supérieure aux multiplicateurs de Lagrange. w ne dépend toujours que des vecteurs supports, pour lesquels la contrainte est saturée :

$$y_i(w^T x_i + b) = 1 - \xi_i.$$

Les vecteurs supports sont les observations x_i pour lesquelles $\xi_i = 0$ ou bien $\xi_i > 0$ mais x_i est dans la marge (bien classé si $0 < \xi_i < 1$ et mal classé si $\xi_i > 1$). Le classifieur tolère ainsi quelques données mal classées mais permet d'être appliqué à des données non linéairement séparables.

6.3.4 SVR : régression à vecteurs de support

Les sous-sections précédentes présentaient les SVM pour une tâche de classification, mais elles peuvent également être utilisées pour des régressions ; on parle alors de régression à vecteurs de support (SVR) [Burges, 1998, Schölkopf and Smola, 2002]. Ce qui suit provient essentiellement du polycopié de Frédéric Sur [Sur, 2024].

Dans une tâche de regression, on cherche une fonction h approchant au mieux le nuage de points $(x_i, y_i)_i$. $h(x,w)$ dépend d'un paramètre à optimiser w, et pour tout $i = 1, ..., n$, $h(x_i)$ doit s'écarter le moins possible de $y_i \in \mathbb{R}$. On souhaite par ailleurs que h soit la plus plate (régulière) possible, afin d'éviter le surapprentissage.

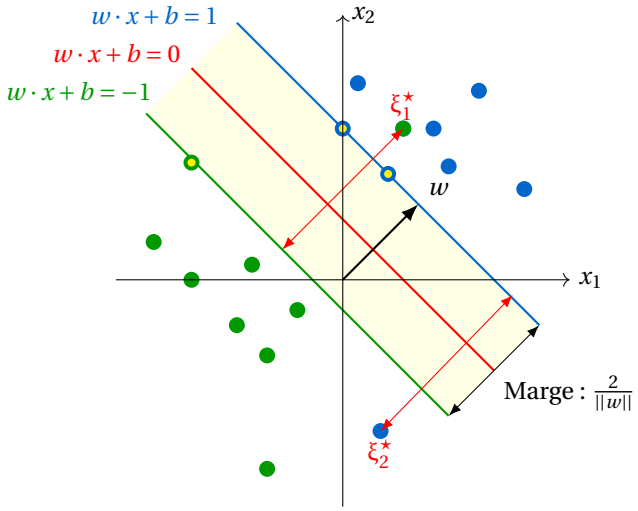


Figure 6.11 – Marge souple avec deux classes d'observations (en bleu et vert) non linéairement séparables. Deux points (l'un vert, l'autre bleu) sont de part et d'autre de la marge. La distance $\xi_1^\star$ et $\xi_2^\star$ à la frontière de leur région les caractérisent.

Il s'agit encore de minimiser une fonction coût, mais dans un sens autre que les moindres carrés. On utilise ici une méthode d'estimation robuste au sens de Huber, qui évalue les écarts en valeur absolue. La perte utilisée en SVR est la perte insensible ("insensitive") définie pour un paramètre ε par

$$|x|_\varepsilon = \begin{cases} 0 & \text{si } |x| < \varepsilon, \\ |x| - \varepsilon & \text{sinon.} \end{cases} \tag{36}$$

On a ainsi :

$$\big|y - h(x)\big|_\varepsilon = \max\big(0, |y - h(x)| - \varepsilon\big). \tag{37}$$

Pour la perte ε-insensible, un écart inférieur à ε entre $h(x_i)$ et y_i n'est pas pris en compte (c.f. fig 6.12). Cela définit un tube de largeur ε de la part et d'autre de la courbe de la fonction h à l'intérieur duquel doivent se trouver tous les points de l'échantillon.

Commençons par une fonction linéaire $h(x) = w^{\mathrm{T}} x + b$. Nous cherchons à résoudre le problème d'optimisation suivant :

$$\begin{cases} \underset{w,b}{\arg\min}\left(||w||^2/2\right) \\ \text{s.c. } |y_i - (w^t x_i + b)| \le \varepsilon, \ \forall i = 1, ..., n. \end{cases} \tag{38}$$

Il peut ne pas y avoir de solution si ε est trop petit. On introduit alors des écarts possibles ξ_i vérifiant :

$$\xi_i = \begin{cases} |y_i - (w^t x_i + b)| - \varepsilon & \text{si } |y_i - (w^t x_i + b)| > \varepsilon, \\ 0 & \text{sinon.} \end{cases} \tag{39}$$

et le problème se ramène à minimiser (sous contraintes des égalités précédentes)

$$\mathrm{E}(w, \lambda) = \frac{1}{n} \sum_{i=1}^{n} \big|y_i - h(x_i, w)\big|_\varepsilon + \lambda ||w||^2 \tag{40}$$

où λ est un paramètre de régularisation. Cela conduit au problème d'optimisation primal SVR

$$\begin{cases} \underset{w,b,\xi,\xi'}{\arg\min}\left(||w||^2/2 + \mathrm{C} \sum_{i=1}^{n} (\xi_i + \xi_i')\right) \\ \text{s.c. } y_i - w^t x_i + b \le \varepsilon + \xi_i \ , \ w^t x_i + b - y_i \le \varepsilon + \xi_i' \ , \ \forall i = 1, .., n. \end{cases}$$

La présence de deux types de variables ξ_i et ξ_i' est due à la présence de valeurs absolues dans les contraintes initiales. Cette fonction n'est pas linéaire, mais on peut la faire disparaître en évaluant son signe et en introduisant deux familles de variables d'écart au lieu d'une. Après transformation, on admet que le problème d'optimisation dual SVR est le suivant :

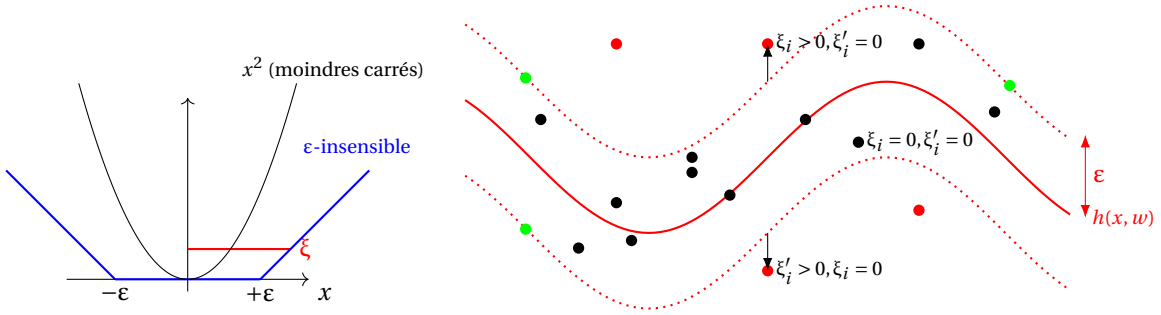


Figure 6.12 – À gauche : la fonction de perte ε-insensible comparée à la perte au sens des moindres carrées. À droite : dans la SVR, la fonction perte définie un ε-tube de part et d'autre de la fonction de régression h dans laquelle doivent se trouver tous les points de l'échantillon. Il est possible de définir des variables d'écarts ξ comme pour une marge souple. En noir, les observations se trouvant dans le ε-tube, en rouge, les données à l'extérieur (pour lesquelles l'une des variables d'écart est non nulle) et en vert les données qui correspondent aux vecteurs de support.

$$\begin{cases} \underset{\alpha,\alpha'}{\arg\min} -\frac{1}{2}\sum_{i,j}(\alpha_i - \alpha'_i)(\alpha_j - \alpha'_j)x_i^{\mathrm{T}}x_j - \varepsilon\sum_i(\alpha_i + \alpha'_i) + \sum_i y_i(\alpha_i - \alpha'_i) \\ \text{s.c. } 0 \le \alpha_i, \alpha'_i \le \mathrm{C},\ \forall i = 1, .., n\ ; \sum_i(\alpha_i - \alpha'_i) = 0. \end{cases}$$

Après résolution, le régresseur s'écrit sous la forme:

$$h(x, w) = \sum_{i=1}^{n}(\alpha_i - \alpha'_i)x_i^{\mathrm{T}}x + b \tag{41}$$

et si des noyaux non linéaires sont utilisés, il s'exprime sous la forme:

$$h(x) = \sum_{i=1}^{n}(\alpha_i - \alpha'_i)\mathrm{K}(x_i, x) + b \tag{42}$$

Les figures Fig. 6.13 et Fig. 6.14 illustrent la régression à base de machines à vecteurs de support.

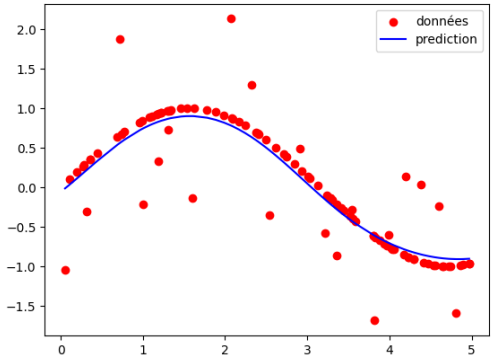


Figure 6.13 – Un nuage de points (en rouge) générés autour d'une sinusoïde perturbée par un bruit aléatoire et la courbe de régression SVR (en bleu).

6.3.5 Conclusion et bibliographie sommaire

Les SVM et SVR nécessitent une recherche délicate de la bonne valeur des hyperparamètres en utilisant, par exemple, la validation croisée ; les modèles sont très sensibles aux valeurs de ces hyperparamètres. Ceci implique qu'il faut toujours d'une part normaliser les features et d'autre part ne considérer qu'un seul hyperparamètre par feature (ce qui n'est pas possible avec scikit-learn).

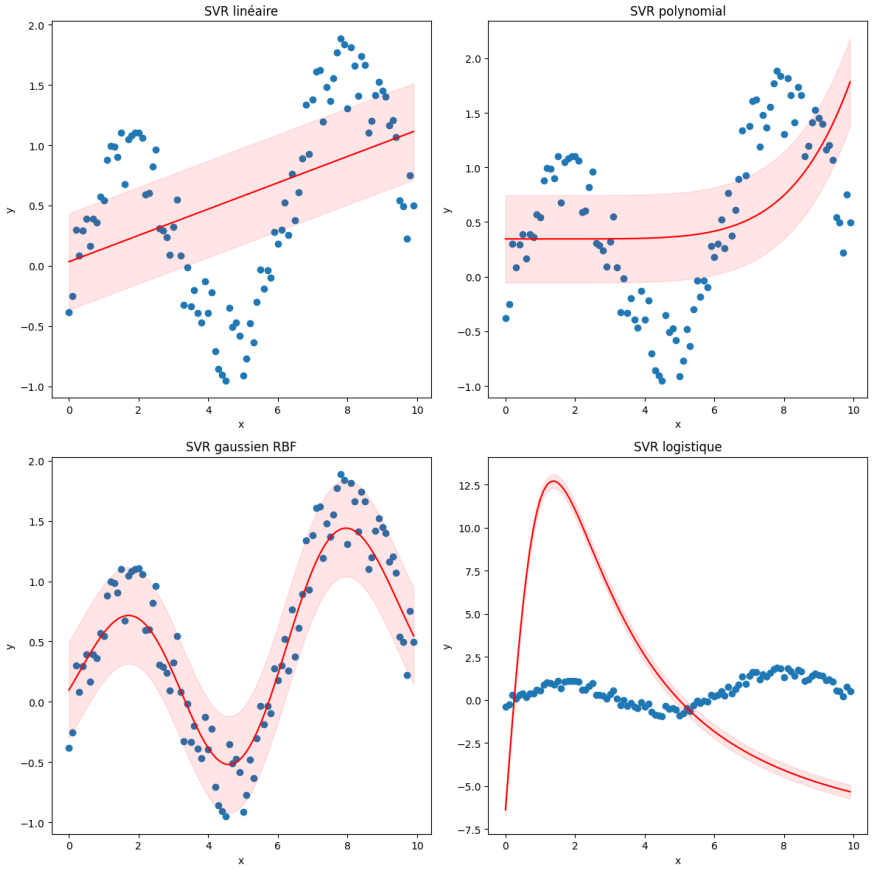


Figure 6.14 – Un nuage de points (en bleu) générés autour d'une sinusoïde perturbée par un bruit aléatoire et les courbes de régressions SVR (en rouge, avec les ε-tubes en rose, pour ε = 0.5) pour 4 types de noyaux : linéaire, polynomial, gaussien et logistique. Noter les performances des noyaux gaussiens.

On peut généraliser les SVM à des problèmes de classification non binaire grâce à des méthodes OVR (one-versus-rest) ou bien OVA (one-versus-all) ou encore OVO (one-versus-one). On pose

$$h(x) = \text{sgn}\left(\underset{m=1,\ldots,\text{M}}{\arg\max}\, h_m(x)\right)$$

où m est le nombre de classes et h_m la fonction de décision en classification binaire de la classe m par rapport à toutes les autres. La figure 6.15 illustre le résultat classification des iris de Fisher avec quatre noyaux différents.

Les méthodes à noyau permettent d'utiliser les SVM ou les SVR pour des problèmes de classification ou régression sur des arbres, des graphes, des chaînes de caractères, des documents constitués de textes. Nous verrons en exercices des noyaux adaptés aux grands modèles de langages (LLM).

En résumé :

- Les SVM permettent de faire de la classification et de la régression.
- Les SVM sont considérées comme l'une des meilleures méthodes de classification ou de régression (la plus utilisée dans les années 1990 avec le Boosting), avec parfois de meilleurs résultats que les réseaux de neurones.
- C'est une bonne alternative aux réseaux de neurones car elles sont plus faciles à entraîner.
- Les estimateurs ont de bonnes propriétés de parcimoine et ne souffrent pas du fléau de la dimension.
- En grande dimension ($d >> n$), les SVM sont efficaces....
- ... mais elles n'estiment pas de probabilité (contrairement à la régression logistique, par exemple).

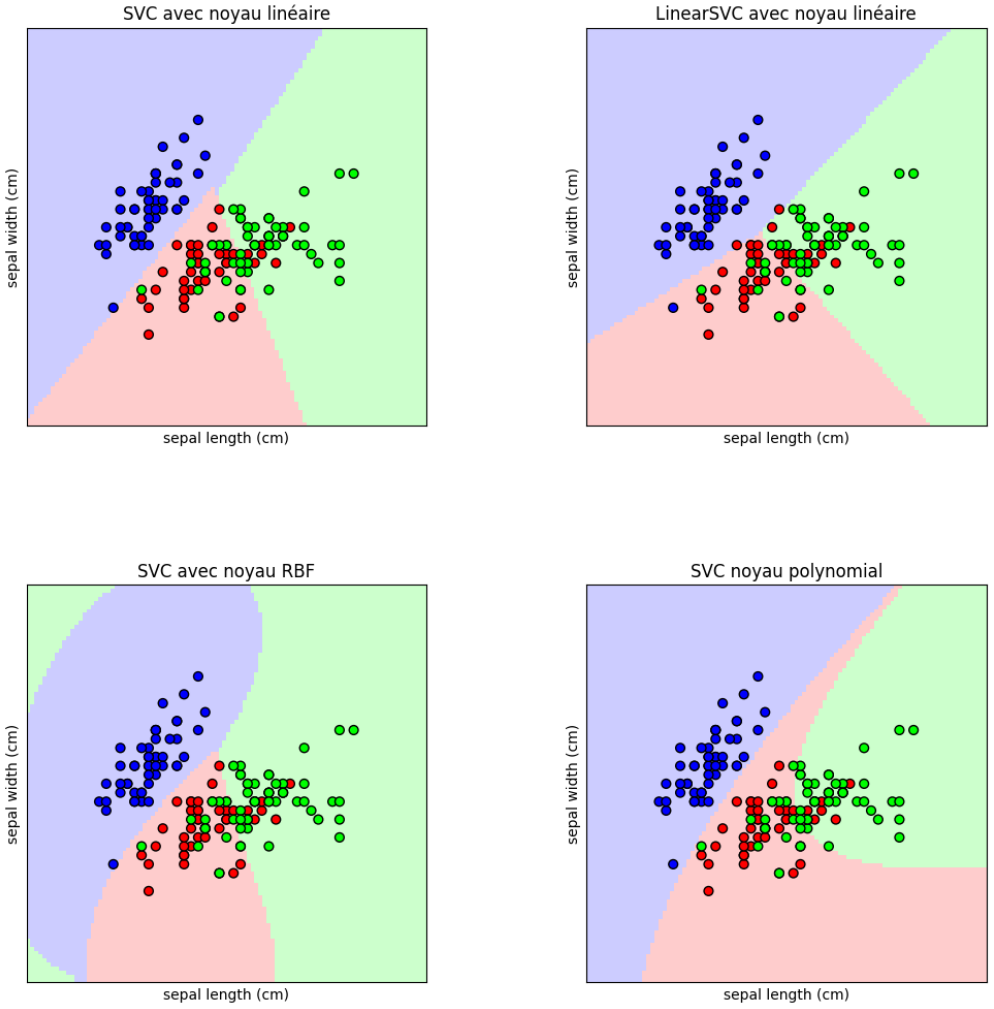


Figure 6.15 – Utilisation des SVM pour classer les 3 espèces d'iris de Fisher. 4 noyaux différents sont proposés. Les régions dédiées à chaque espèce sont représentées dans le plan caractérisé par la longueur et la largeur du sépale.

- Les SVM ne sont pas interprétables et pas toujours très rapides.
- Le paramétrage des hyperparamètres est délicat : choix du noyau, valeur de la constante C de régularisation, choix de l'algorithme d'optimisation.
- Elles sont parfois moins performantes que les forêts aléatoires ou le Boosting.

Bibliographie :

- Learning With Kernels: Support Vector Machines, Regularization, Optimization and Beyond, 2002.
- Hofmann, Schölkopf, Smola, Kernel methods in machine learning, Annals of Statistics, 2008.
- Burges, A tutorial on support vector machines for pattern recognition, Data Mining, 1998.
- Smola, Schölkopf, A tutorial on support vector regression, Statistics and Computing, 2004.
- + Chapitre SVM (p.367) Intro to Statistical Learning with Python.
- + Chapitre 8 du polycopié de Frédéric Sur.

6.4 Boosting

6.4.1 Introduction et motivation

Le Boosting est une méthode d'apprentissage supervisé consistant à construire un prédicteur fiable en agrégeant des prédicteurs faibles de façon itérative, sur le principe de la sagesse des foules ("Wisdom of the Many").

Il existe de nombreux algorithmes de Boosting : Adaboost [Freund and Schapire, 1996], historiquement le premier, Gradient Boosting [Friedman, 2001] et les plus récents XGBoost [Chen and Guestrin, 2016] et Light GBM.

À l'instar des SVM, nous allons voir la technique de Boosting selon deux points de vue différents. Le point de vue qui suit est celui d'une méthode de convexification et celui que nous verrons dans la section suivante sera une méthode d'agrégation ; le Boosting appartient à ces deux catégories à la fois.

Une façon de convexifier l'ensemble des prédicteurs $\mathscr{H} \subset \mathscr{F}(\mathbb{X}, \{-1,1\})$, différente de celle exposée pour les machines à vecteurs de support, est de considérer des combinaisons linéaires convexes de prédicteurs :

$$\mathscr{H}_\lambda = \{ \sum_{m=1}^{M} \lambda_m h_m;\ \lambda_m \geq 0,\ h_m \in \mathscr{H},\ \sum_m \lambda_m \leq \lambda \}. \tag{43}$$

L'ensemble $\mathscr{H}_\lambda$ est convexe et en posant

$$\widehat{h}_{n,\lambda} \in \operatorname*{arg\,min}_{h \in \mathscr{H}_\lambda} A_n(h) \tag{44}$$

on définit un prédicteur Boosting de façon générique.

On montre que si $\lambda = \lambda_n \to +\infty$, si $\lambda_n \phi'(\lambda_n)\sqrt{\ln n/n} \to +\infty$ et si $\mathscr{H}$ a une dimension de Vapnik finie, alors $\widehat{h}_{n,\lambda_n}$ est universellement consistant. Il s'agit néanmoins d'un problème difficile à résoudre car de dimension infinie. L'algorithme Adaboost propose une méthode permettant de le résoudre de façon efficace.

6.4.2 Adaboost : Adaptive Boosting

La méthode a été inventée par Freund et Shapire dans les années 90 [Freund and Schapire, 1996]. À chaque itération, l'algorithme met à jour l'estimateur en favorisant son apprentissage sur les données erronées de l'itération précédente, qui sont pondérées de façon plus importante : on "booste" les données mal classées à chaque itération. Il produit un classifieur performant à partir de classifieurs faibles (et réduit le biais), en donnant plus de poids aux observations difficiles à prédire. C'est un algorithme simple, rapide et facile à implémenter car il possède très peu de paramètres. Il est flexible et s'adapte à tout type de classifieur faible sans *a priori*, même s'il est très souvent utilisé avec des arbres de décisions de pronfondeur 1 ou 2 comme classifieurs faibles. Il permet également d'effectuer de la régression. C'est au final un algorithme versatile avec beaucoup d'applications possibles (reconnaissance d'images, de textes, moteurs de recherche).

Son comportement vis à vis du surapprentissage est par contre ambigu et l'entrainement séquentiel est coûteux en temps de calcul. Il est par ailleurs sensible aux valeurs aberrantes et au bruit. Le nombre M d'itérations ne doit pas être trop grand pour éviter le surapprentissage. La figure 6.16 illustre de façon traditionnelle cette méthode.

Le principe de l'algorithme est le suivant : à chaque itération m, on met à jour l'estimateur $h_m(x) = h_{m-1}(x) + \alpha h$, où α et h doivent rendre h_m minimal pour le risque empirique R_n associé à la fonction de perte exponentielle (qui est convexe). Les différentes étapes sont

0 • Initialisation : $w_i(0) = 1/n$.

Puis à chaque itération $m = 1, \ldots, M$,

1 • Entraîner $h_m(x)$ sur l'échantillon pondéré par $w = (w_i(m))_i$:

$$h_m \in \operatorname*{arg\,min}_{h \in \mathscr{H}} \sum_{i=1}^{n} w_i(m) \mathbb{1}_{[y_i \neq h(x_i)]}. \tag{45}$$

2 • Calculer l'erreur normalisée :

$$\varepsilon_m = \sum_{i=1}^{n} \frac{w_i(m)}{||w||_1} \mathbb{1}_{[y_i \neq h_m(x_i)]}. \tag{46}$$

3 • Calculer le poids de l'itération m : $\alpha_m = \ln \sqrt{(1-\varepsilon_m)/\varepsilon_m}$.

4 • Mettre à jour les poids des observations :

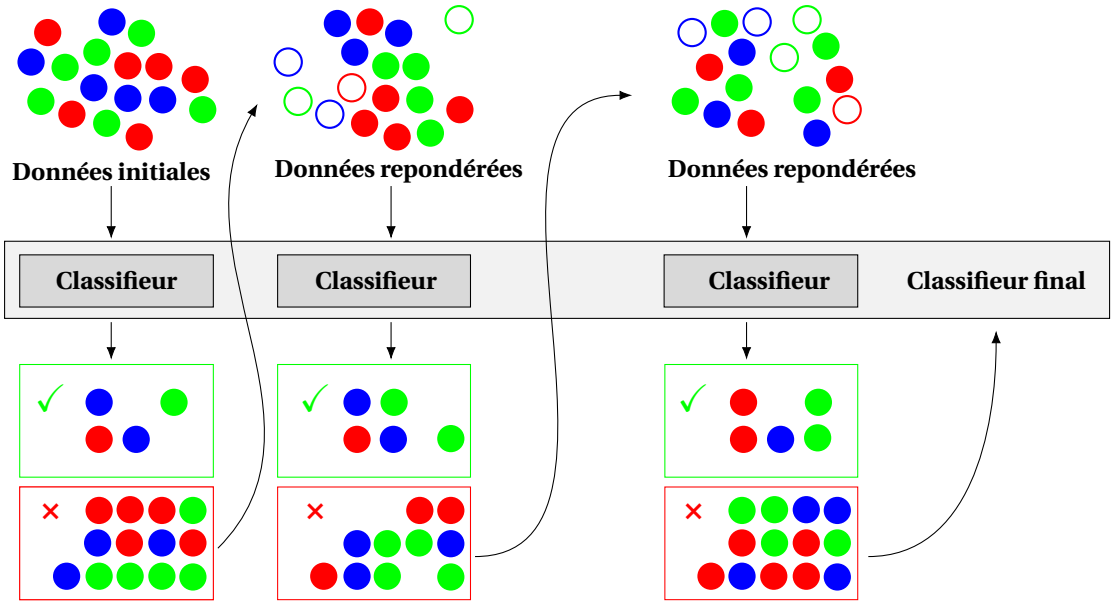


Figure 6.16 – Schéma traditionnel présentant la technique de Boosting : un premier classifieur faible est entraîné sur l'ensemble des données initiales. Il pondère de façon différentes les données bien (rectangle vert) et mal classées (rectangle rouge) en favorisant ces dernières afin que le classifieur se focalise dessus. Un second classifieur est entraîné sur les données pondérées (les données bien classées, moins prises en compte, sont représentées par des cercles) et remet à jour les pondérations, etc. Le classifieur final est une moyenne pondérée des classifieurs faibles utilisés à chaque itéralion.

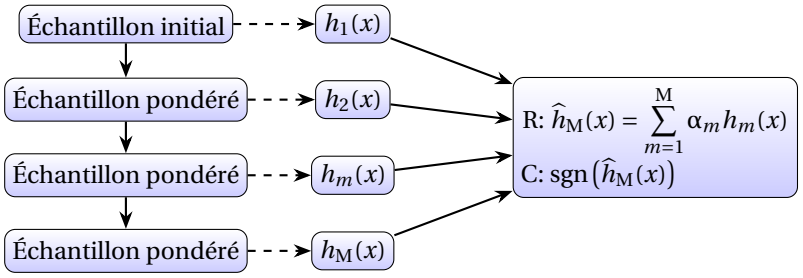


Figure 6.17 – Forme du prédicteur Boosting (C : classifieur, R: régresseur).

$$w_i(m+1) = w_i(m)\exp\left(\alpha_m \mathbb{1}_{[y_i \neq h_m(x_i)]}\right),\ \forall i = 1,\dots,n. \tag{47}$$

$$\Rightarrow \widehat{h}_M(x) = \sum_{m=1}^{M} \alpha_m h_m(x) \tag{48}$$

$\forall m$, $h_m = h_{m-1} + \widehat{\alpha}\widehat{h}$, où h_m est le minimiseur du ϕ-risque empirique par rapport à la perte Boosting :

$$(\widehat{h}, \widehat{\alpha}) \in \underset{h\in\mathcal{H},\alpha\in\mathbb{R}}{\arg\min} \frac{1}{n}\sum_{i=1}^{n} \phi\left(-y_i[h_{m-1}(x_i) + \alpha h(x_i)]\right) \tag{49}$$

$$= \underset{h,\alpha}{\arg\min} \frac{1}{n}\sum_{i=1}^{n} w_i(m)\exp\left(-y_i \alpha h(x_i)\right) \tag{50}$$

$$= \underset{h,\alpha}{\arg\min}\left(\frac{e^{-\alpha}}{n}\sum_{y_i = h(x_i)} w_i(m) + \frac{e^{\alpha}}{n}\sum_{y_i \neq h(x_i)} w_i(m)\right) \tag{51}$$

$\phi(x) = e^x$, $w_i(m) = \exp(-y_i h_{m-1}(x_i))$.

On trouve $\widehat{h}$ et l'on en déduit $\widehat{\alpha} = \alpha_m$. Pourquoi est-ce que cela fonctionne ?

Proof. Soit $h_0 \in \mathcal{F}(\mathbb{X},\mathbb{R})$, $\phi(x) = e^x$ la perte boosting et A_n le ϕ-risque. On cherche le couple $(\widehat{\alpha}, \widehat{h})$ qui minimise ce ϕ-risque :

$$(\widehat{\alpha}, \widehat{h}) =\in \underset{\alpha\geq 0, h\in\mathscr{H}}{\arg\min} A_n(h_0 + \alpha h) \tag{52}$$

L'expression à minimiser s'écrit

$$A_n(h_0 + \alpha h) = \frac{1}{n}\sum_{i=1}^{n} \exp\left(-Y_i h_0(X_i) - \alpha Y_i h(X_i)\right) \tag{53}$$

$$= \frac{1}{n}\sum_{i=1}^{n} w_i e^{-\alpha Y_i h(X_i)} \tag{54}$$

$$= \frac{1}{n}\sum_{i=1}^{n} w_i e^{-\alpha Y_i h(X_i)} \left[\mathbb{1}_{[Y_i = h(X_i)]} - \mathbb{1}_{[Y_i \neq h(X_i)]}\right] \tag{55}$$

en posant (pour alléger les formules et sans perte de généralité) :

$$w_i = \phi(-Y_i h_0(X_i)) / \sum_{i=1}^{n} \phi(-Y_i h_0(X_i)) \tag{56}$$

qui ne dépend que de l'échantillon de données et de h_0 (autrement dit on normalise les poids w_i).

Puisque Y_i et $h(X_i)$ sont à valeurs binaires (± 1), $Y_i = h(X_i) \iff Y_i h(X_i) = 1$ et de même, $Y_i \neq h(X_i) \iff Y_i h(X_i) = -1$. Ainsi,

$$\begin{aligned}(\widehat{\alpha}, \widehat{h}) &= \underset{\alpha\geq 0, h\in\mathscr{H}}{\arg\min} \left(e^{-\alpha}\frac{1}{n}\sum_{i=1}^{n} w_i \mathbb{1}_{[Y_i = h(X_i)]} + e^{\alpha}\frac{1}{n}\sum_{i=1}^{n} w_i \mathbb{1}_{[Y_i \neq h(X_i)]} \right) \\ &= \underset{\alpha>0}{\arg\min} \left((e^{\alpha} - e^{-\alpha})\frac{1}{n}\sum_{i=1}^{n} w_i \mathbb{1}_{[Y_i \neq h(X_i)]} + \frac{1}{n}e^{-\alpha} \right).\end{aligned}$$

$\forall \alpha \geq 0$, le minimum est atteint quand

$$\widehat{h} \in \underset{h}{\arg\min} \sum_i w_i \mathbb{1}_{[Y_i \neq h(X_i)]}. \tag{57}$$

La forme de l'expression permet de maximiser séparemment h et α. $\widehat{h}$ étant choisi pour maximiser h, alors

$$\widehat{\alpha} \in \underset{\alpha>0}{\arg\min} \left((e^{\alpha} + e^{-\alpha})\varepsilon + e^{-\alpha} \right) = \underset{\alpha}{\arg\min} G(\alpha), \tag{58}$$

avec $\varepsilon = \sum_i w_i \mathbb{1}_{[Y_i \neq h(X_i)]}$.

$G'(\alpha) = (e^{\alpha} - e^{-\alpha})\varepsilon - e^{-\alpha}$ et $G''(\alpha) = e^{\alpha}\varepsilon + e^{-\alpha}(1-\varepsilon)$. Ainsi, G est convexe et sa dérivée s'annule en

$$\alpha = \frac{1}{2}\ln\left(\frac{1-\varepsilon}{\varepsilon}\right) \tag{59}$$

que l'on choisit dans l'étape 3 de l'algorithme.

Il est important de noter que la perte Boosting $l_b(x) = e^x$ est convexe et que $l_b(x) \geq l_{0-1}(x)$. Adaboost minimise cette perte en minimisant l'expression $\sum_i w_i \mathbb{1}_{[\bullet]}$. □

Quelques remarques :

• La magnitude de $yh(x)$ est la marge, qui peut être interprêtée comme la confiance qu'a le classifieur h en sa prédiction.

• L'expression (57) peut se réécrire sous la forme

$$\widehat{h} = \underset{h}{\arg\min}\, \mathbb{E}_{x\sim\mathbb{P}}\left[\mathbb{1}_{[Y_i \neq h(X_i)]}\right] \tag{60}$$

où $\mathbb{P}$ est la distribution de probabilités empirique des x_i pondérés par les poids $\mathbb{P}(i) = w_i$. Cette expression va nous permettre de voir le Boosting comme un algorithme de descente de gradient stochasique.

• Les classifieurs faibles utilisés ne doivent pas être trop faibles : $\varepsilon_m = 1/2 - \gamma_n$ avec $\gamma_n > 0$.

• Majoration de l'erreur empirique : on peut montrer que

$$\mathrm{R}_n(\widehat{h}_{\mathrm{M}}) = \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}_{[y_i \neq \widehat{h}_{\mathrm{M}}(x_i)]} \leq \exp\left(-2\sum_{m=1}^{\mathrm{M}} \gamma_m^2\right) \leq e^{-2\mathrm{M}\gamma^2} \tag{61}$$

avec $\gamma \leq \gamma_m$. Ainsi, l'erreur empirique tend exponentiellement vite vers 0 quand le nombre d'itérations tend vers l'infini.

• Majoration de l'erreur de généralisation : Freund et Shapire on également montré que :

$$\mathrm{R}(\widehat{h}_{\mathrm{M}}) = \mathbb{P}[\widehat{h}_{\mathrm{M}}(\mathrm{X}) \neq \mathrm{Y}] \leq \mathrm{R}_n(\widehat{h}_{\mathrm{M}}) + \tilde{\mathcal{O}}\left(\sqrt{\frac{\mathrm{MV}}{n}}\right) \tag{62}$$

où V est la dimension de Vapnik de l'espace des prédicteurs (c.f. annexes). Ainsi, lorsque M est grand (et V aussi), on a un risque de surapprentissage, car le second terme domine. Il est donc important de bien choisir M. Les grandes valeurs de V correspondent à des classifieurs faibles complexes.

• Enfin, Bartlett et Traskin ont montré que si $\mathrm{M} = \mathrm{M}_n = n^{1-\varepsilon}$, $\varepsilon \in]0,1[$, alors Adaboost est fortement et universellement consistant en classification :

$$\lim_{n\to+\infty} \mathrm{L}(\widehat{h}_{\mathrm{M}_n}) = \mathrm{L}^\star, \text{ p.s.} \tag{63}$$

Adaboost peut être étendu à des étiquettes non binaires. L'une des méthodes correspondantes s'appelle SAMME pour Stagewise Additive Modeling Multi-class Exponentiel et effectue une classification dans le cas où $\mathbb{Y} = \{1, ..., \mathrm{K}\}$. Le principe est le suivant : partant de classifieurs faibles de la forme

$$\mathbb{P}[h(\mathrm{X}) = \mathrm{Y}] \geq \frac{1}{\mathrm{K}} + \gamma, \tag{64}$$

les étapes sont les mêmes à l'exception de l'étape 3 où le poids est donné par

$$\alpha_m = \ln\frac{1-\varepsilon_m}{\varepsilon_m} + \ln(\mathrm{K}-1) \tag{65}$$

et de la sortie qui est donnée par

$$\widehat{h}_{\mathrm{M}}(x) = \underset{k=1...\mathrm{K}}{\arg\max} \sum_{m=1}^{\mathrm{M}} \alpha_m \mathbb{1}_{[h_m(x)=k]}. \tag{66}$$

6.4.3 Gradient Boosting

Le Gradient Boosting interprète le Boosting comme une descente de gradient particulière.

Rappelons brièvement que l'algorithme de descente de gradient s'inspire de la méthode de Newton-Raphson qui permet de trouver de façon itérative un zéro d'une fonction u strictement convexe de $\mathbb{R}$ dans $\mathbb{R}$ (c.f. fig. 6.4.3). L'algorithme construit une suite récurrente de la forme

$$x_{n+1} = x_n - \frac{u(x_n)}{u'(x_n)}.$$

Au point x_n l'équation de la tangente à la courbe de u est

$$y = u(x) = u(x_n) + u'(x_n)(x - x_n)$$

et en remplaçant y par 0, l'équation précédente montre que cette tangente coupe l'axe des abscisses en x_{n+1}. De proche en proche, on se rapproche d'un zéro de la fonction à une vitesse de convergence quadratique.

Dans l'algorithme de descente de gradient, on remplace u par sa dérivée et l'on fixe un paramètre η de telle sorte que

$$x_{n+1} = x_n - \eta u'(x_n)$$

afin de déterminer un minimum local de la fonction u (et si possible son minimum global) : lorsque la dérivée s'annule et change de signe, on a atteint un minimum. Les déplacements successifs s'effectuent dans la direction opposée au gradient (qui pointe vers la direction de plus forte croissance) afin de converger vers le minimum. Le paramètre η influe sur la vitesse de convergence (il joue le rôle de "learning rate" en apprentissage).

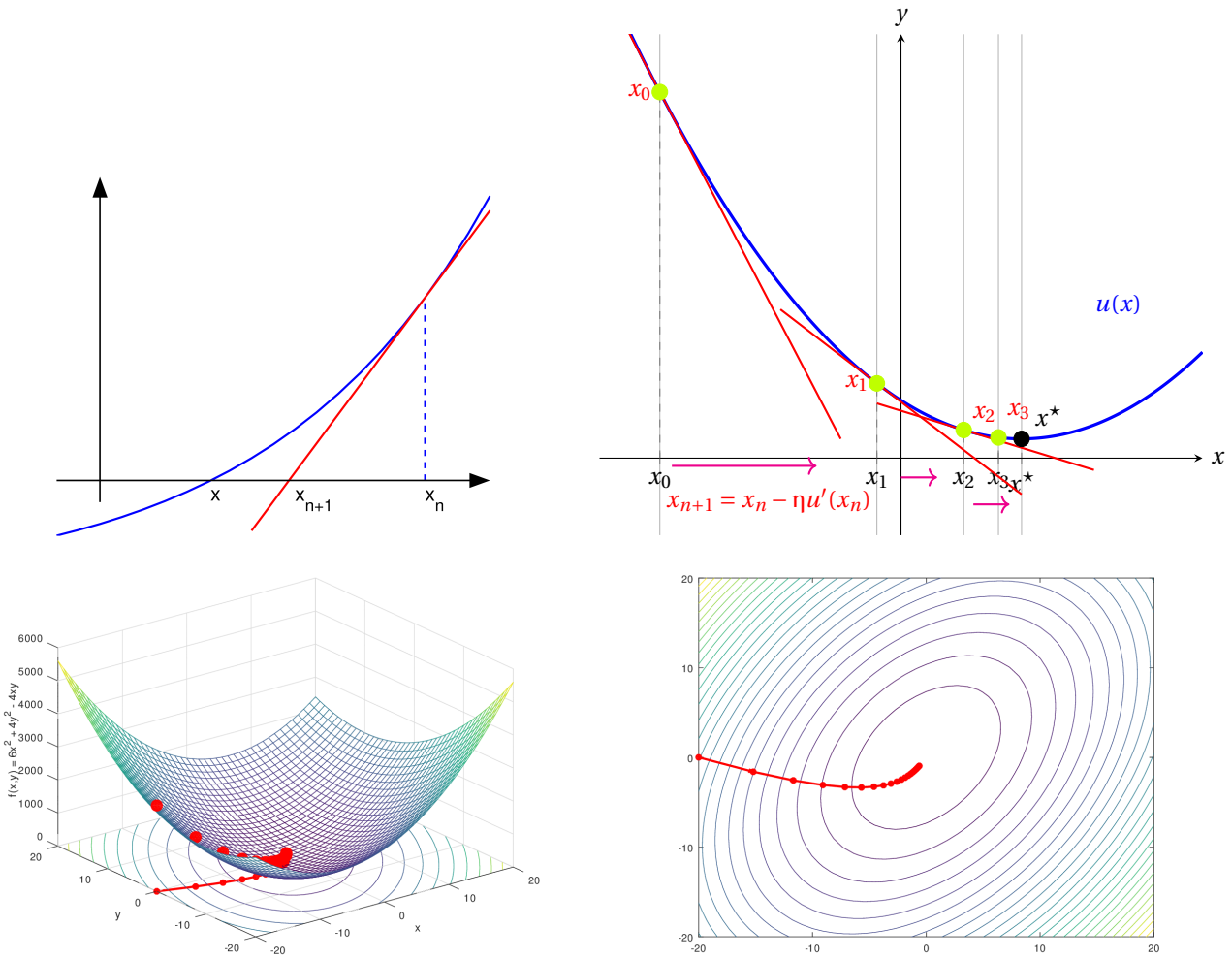


Figure 6.18 – Illustration de l'algorithme de Newton-Raphson (à gauche) et de celui de la descente de gradient (à droite) ; première ligne en dimension 1, seconde ligne en dimension 2.

Plaçons dans un cadre de régression, en choisissant la perte quadratique $l(y,y') = (y-y')^2/2$. Pour un indice i fixé, en posant $y_i{}' = h(x_i)$, l'erreur de prédiction de l'observation (x_i, y_i) est $y_i - y_i{}' = y_i - h(x_i)$. La dérivée partielle par rapport à y' de la perte est donnée par

$$\frac{\partial}{\partial y'} l(y,y') = \frac{1}{2}\frac{\partial l}{\partial y'}(y-y')^2 = y'-y. \tag{67}$$

Cette dérivée partielle, évaluée en $(y_i, h(x_i))$ est égale à l'opposé de cette perte. L'expression du risque empirique est

$$\mathrm{R}_n(h) = \frac{1}{n}\sum_{i=1}^{n} l(y_i, h(x_i)) = \frac{1}{2n}\sum_{i=1}^{n}(y_i - h(x_i))^2. \tag{68}$$

$\mathrm{R}_n(h)$ est une fonction du prédicteur h, mais également une fonction des variables $y_1,\dots,y_n$ et $y_1{}' = h(x_1),\dots,y_n{}' = h(x_n)$. Le gradient de $\mathrm{R}_n(h)$ en tant que fonction de la variable y', évalué aux points de l'échantillon $(y_i, h(x_i))_i$, est le vecteur dont les coordonnées sont

$$\frac{\partial \mathrm{R}_n(h)}{\partial y_i{}'}(y_1,\dots,y_n,y_1{}',\dots,y_n{}')\Big|_{y_i{}'=h(x_i)} = \frac{1}{n}\frac{\partial l}{\partial y_i{}'}(y_i,y_i{}')\Big|_{y_i{}'=h(x_i)} = \frac{1}{n}\big(h(x_i)-y_i\big). \tag{69}$$

Les coordonnées du vecteur gradient sont donc proportionnelles aux erreurs de prédiction de chaque observation. On note traditionnellement ce vecteur, de façon peu rigoureuse et très ambiguë :

$$\nabla \mathrm{R}_n(h) = \left(\frac{\partial l(y_i, h(x_i))}{\partial h(x_i)}\right)^{\mathrm{T}}_{i=1,..,n} \tag{70}$$

Il s'agit d'un vecteur à n coordonnées, chaque coordonnée étant égale à la dérivée partielle du risque, par rapport à la seconde coordonnée (relative à $h(x_i)$), évalué au point $(y_i, h(x_i))$.

En fait, la minimisation du risque $\mathrm{R}_n(h)$ est effectuée sur la fonction h et non sur les points de l'échantillon d'observation. C'est donc en tant que fonction de h qu'elle est pertinente. En ce sens, on parle de gradient fonctionnel car la minimisation se fait en choissant la fonction h, à x_i et y_i fixés.

À l'itération m, on dispose dans l'algorithme de Boosting classique d'un prédicteur $h_{m-1}(x)$ et l'on veut déterminer $h_m(x)$ de la forme

$$h_m(x) = h_{m-1}(x) + \alpha h(x) \tag{71}$$

où $\alpha h(x)$ minimise l'erreur de prédiction. Dans l'idéal, si $\alpha h(x)$ est égal à la perte quadratique calculée ci-dessus, alors $h_m(x_i) = y_i$ et l'erreur de prédiction est nulle. C'est la raison pour laquelle on choisit comme expression de $h(x)$ le gradient du risque.

L'idée du gradient boosting est donc de généraliser la méthode à une perte quelconque et à un problème de regréssion comme de classification.

Nous reprenons maintenant les hypothèses du problème de classification binaire : $\mathscr{D}_n = \{(x_1, y_1, ..., (x_n, y_n)\}$ avec $x_i \in \mathbb{R}^d$ et $y_i = \pm 1$.

À une itération donnée, on a un classifieur h_{m-1} à améliorer. On lui ajoute αh de telle sorte que $\mathrm{R}_n(h_{m-1} + \alpha h)$ diminue au maximum et d'après la discussion précédente, il faut prendre l'opposé du gradient: $-\nabla \mathrm{R}_n(h_{m-1})$. On pose donc :

$$h_m(x) = h_{m-1}(x) - \alpha \nabla \mathrm{R}_n(h_{m-1}) = \left(h_{m-1}(x_i) - \frac{\alpha}{n} \frac{\partial l}{\partial h(x_i)}(y_i, h_{m-1}(x_i)) \right)_i$$

Il y a deux problèmes :

- La récurrence donne h_m uniquement aux points x_i.
- Le gradient $\nabla \mathrm{R}_n$ n'a aucune raison d'appartenir à $\mathscr{H}$, donc h_m non plus.

Pour obtenir $h_m(x)$ $\forall x \in \mathbb{R}^d$ et s'assurer que $h_m \in \mathscr{H}$, on effectue une régression sur $\mathscr{D}_{n,m} = (x_i, u_i)_{i=1,..,n}$ avec

$$u_i = u_i(m) = -\frac{\partial l}{\partial x_i}(y_i, h_{m-1}(x_i))$$

On cherche en quelque sorte la fonction $h \in \mathscr{H}$ la plus proche de $\nabla \mathrm{R}_n(h_{m-1})$: la solution h_m s'obtient en ajustant le classifieur sur $(x_i, u_i)_i$. On détermine ensuite la valeur optimale α_m de α.

Principe de l'algorithme de Friedman [Friedman, 2001] qui implémente cette méthode :

- Initialisation : $h_0(.) = \frac{1}{n} \arg\min_c \sum_{i=1}^n l(c, y_i)$
- Pour $m = 1, ..., \mathrm{M}$, étant donné h_{m-1} calculé à l'itération précédente,
  - $\forall i = 1, .., n$, calculer les pseudos-résidus u_i.
  - ajuster un classifieur de $\mathscr{H}$ sur l'échantillon $(x_i, u_i)_i$:
  $$\overline{h}_m = \arg\min_{h,\alpha} \sum_i [u_i - \alpha h(x_i)]^2$$
  - déterminer α optimal :
  $$\alpha_m = \arg\min_{\alpha} \sum_i l\left(h_{m-1}(x_i) + \alpha \overline{h}_m(x_i)\right)$$
  - mettre à jour : $h_m(x) = h_{m-1}(x) + \alpha_m \overline{h}_m(x)$.

En sortie, on obtient bien une combinaison linéaire (convexe) d'estimateurs de $\mathscr{H}$

$$\widehat{h}_{\mathrm{M}}(x) = \sum_{m=1}^{\mathrm{M}} \alpha_m h_m(x). \tag{72}$$

que l'on transforme en classifieur binaire :

$$\mathrm{sgn}\left(\widehat{h}_{\mathrm{M}}(x)\right). \tag{73}$$

Contrairement à Adaboost, on ne modifie pas directement les poids de certaines données entre les itérations, mais on agrandit l'ensemble possible des classifieurs faibles à ajouter au classifieur courant. Plusieurs paramètres sont à calibrer :

- La fonction de perte (dérivable et convexe). En classification binaire, on utilise $l(y,h(x)) = \exp(-yh(x))$ pour Adaboost et $l(y,h(x)) = \ln(1+\exp(-2yh(x)))$ pour Logiboost (variante de Adaboost).
- Le nombre M d'itérations, qui est à déterminer avec soin pour éviter le surapprentissage.
- Le paramètre de régularisation α, lorsqu'il est fixe (c'est le "learning rate" : souvent ~ 0.1). Si $\alpha = 1$, avec la perte exponentielle, on obtient à peu près l'algorithme Adaboost. α contrôle la vitesse de minimisation : lorsque α augmente, M diminue et vice-versa. Dans la version des deux algorithmes précédents, α n'est pas fixe et ses différentes valeurs α_m sont optimisées dans l'algorithme à chaque itération.
- Les paramètres de chaque classifieur constituant la combinaison linéaire.

Le Boosting réduit le biais mais augmente la variance. Il est par conséquent préférable de choisir un classifieur faible possédant un biais élevé, afin d'avoir une variance faible. Le biais sera réduit par l'algorithme, mais pas la variance. Ceci explique l'utilisation courante d'arbre peu profonds comme classifieurs faibles.

Pour déterminer le nombre optimal M d'itérations, on peut l'estimer à partir du risque empirique en évaluant

$$\widehat{\mathrm{M}} = \arg\min_{\mathrm{M}} \frac{1}{n}\sum_{i=1}^{n} l(\mathrm{Y}_i, \widehat{h}_{\mathrm{M}}(\mathrm{X}_i)), \tag{74}$$

mais la somme est constituée de variables aléatoires qui ne sont pas indépendantes. Il faut donc utiliser une validation croisée.

Adaboost correspond (presque) au Gradient Boosting pour la perte $l(y,h(x)) = \exp(-yh(x))$. La sortie est un estimateur de

$$h^\star(x) = \frac{1}{2}\ln\left(\frac{\mathbb{P}[\mathrm{Y}=1|\mathrm{X}=x]}{\mathbb{P}[\mathrm{Y}=-1|\mathrm{X}=x]}\right). \tag{75}$$

Si Y est à valeurs dans $\{0,1\}$, nous avons vu dans de cours de statistique mathématique, en étudiant la régression logistique, que la loi conditionnelle de Y sachant $[\mathrm{X}=x]$ est une loi de Bernoulli de paramètre $p(x) = \mathbb{P}[\mathrm{Y}=1|\mathrm{X}=x]$ et que

$$p(x) = \frac{e^{x^{\mathrm{T}}\beta}}{1+e^{x^{\mathrm{T}}\beta}} \tag{76}$$

où β est le vecteur des coefficients d'une régression linéaire. Si, à la place de cette définition, on pose

$$p(x) = \frac{e^{h(x)}}{1+e^{h(x)}} \tag{77}$$

pour une fonction vectorielle h, alors l'expression de la log-vraisemblance permet de définir la fonction de perte à minimiser :

$$l(x,y) = \ln\left(1+\exp(-2yh(x))\right). \tag{78}$$

C'est la perte logistique, qui définit une fonction convexe. L'algorithme correspondant s'appelle Logisboost et minimise le même estimateur logistique qu'Adaboost. Il est moins sensible aux observations mal classées.

Il existe également des versions stochastiques de Gradient Boosting utilisant un algorithme de descente de gradient stochastique, qui est à la fois plus rapide et de meilleure précision.

6.4.4 Boosting multi-classes : SAMME

Comme précédemment, il est possible de généraliser l'algorithme au cas multi-classes, en définissant la probabilité d'appartenir à la classe $k \in \{1,...,\mathrm{K}\}$ par

$$p_k(x) = \frac{e^{h_k(x)}}{\sum_{i=1}^{\mathrm{K}} e^{h_i(x)}}, \tag{79}$$

avec $\sum_i h_i(x) = 0$. On calcule simultanément le vecteur de fonctions $\widehat{h}_m = (\widehat{h}_{m_1},...,\widehat{h}_{m_{\mathrm{K}}})$ avec

$$\mathrm{L}(y,h(x)) = -\sum_{k=1}^{\mathrm{K}} \mathbb{1}_{[y=k]} \ln p_k(x) \tag{80}$$

$$\frac{\partial \mathrm{L}}{\partial x}\mathrm{L}(y,x)|_{x=h_k} = \mathbb{1}_{[y=k]} - p_k(x) \tag{81}$$

L'algorithme correspondant s'appelle SAMME pour "Stagewise Additive Modeling using a Multi-class Exponential loss function" :

À chaque itération $m = 1, ..., \mathrm{M}$,

1 • Entraîner $h_m(x)$ sur l'échantillon pondéré par $w = (w_i(m))_i$:

$$h_m \in \underset{h\in\mathscr{H}}{\arg\min} \sum_{i=1}^{n} w_i(m)\mathbb{1}_{[y_i\neq h(x_i)]}. \tag{82}$$

2 • Calculer l'erreur normalisée :

$$\varepsilon_m = \sum_{i=1}^{n} \frac{w_i(m)}{||w||_1}\mathbb{1}_{[y_i\neq h_m(x_i)]}. \tag{83}$$

3 • Calculer le poids de l'itération m : $\alpha_m = \ln\sqrt{(1-\varepsilon_m)/\varepsilon_m} + \ln(\mathrm{K}-1)$.

4 • Mettre à jour les poids des observations :

$$w_i(m+1) = w_i(m)\exp\left(\alpha_m \mathbb{1}_{[y_i\neq h_m(x_i)]}\right),\ \forall i = 1, ..., n. \tag{84}$$

La sortie est calculée à partir de

$$\widehat{h}_{\mathrm{M}}(x) = \underset{k=1,..,\mathrm{K}}{\arg\max} \sum_{m=1}^{\mathrm{M}} \alpha_m \mathbb{1}_{[h_m(x)=k]} \tag{85}$$

6.4.5 Extreme Gradient Boosting : XGBoost

XGBoost [Chen and Guestrin, 2016] est une version sophistiquée de Gradient Boosting qui optimise la plupart des étapes de l'algorithme initial :

- Ajoute de la régularisation dans les entrainements des classifieurs faibles.
- Utilise Newton-Raphson à la place du gradient.
- Utilise un développement limité d'ordre 2 de la fonction de perte.
- Parallélise certaines étapes lorsque c'est possible.
- Exploite la parcimonie lorsque c'est possible.

Toutes ces optimisations font de GXBoost un algorithme très efficace avec une grande scalabilité.

6.4.6 Boosting pour la régression : L^2-Boosting

En régression, on utilise simplement l'algorithme de Gradient Boosting avec la fonction de perte donnée par

$$l(y, h(x)) = \frac{1}{2}(y - h(x))^2. \tag{86}$$

dans un cadre où $\mathbb{Y} = \mathbb{R}$.

À l'issue des M itérations, l'estimateur $\widehat{h}_{\mathrm{M}}$ est un estimateur du minimiseur du risque empirique :

$$h^\star = \mathbb{E}[\mathrm{Y}|\mathrm{X} = x] = \underset{h}{\arg\min}\,\mathbb{E}\left[(\mathrm{Y} - h(\mathrm{X}))^2\right] \tag{87}$$

Dans ce cas particulier, les variables u_i sont simplement de la forme $u_i = y_i - h_{m-1}(x_i)$ et correspondent aux résidus de la régression à l'itération $m-1$.

Les figures Fig. 6.19 et Fig. 6.20 illustrent les méthodes de Boosting pour la régression.

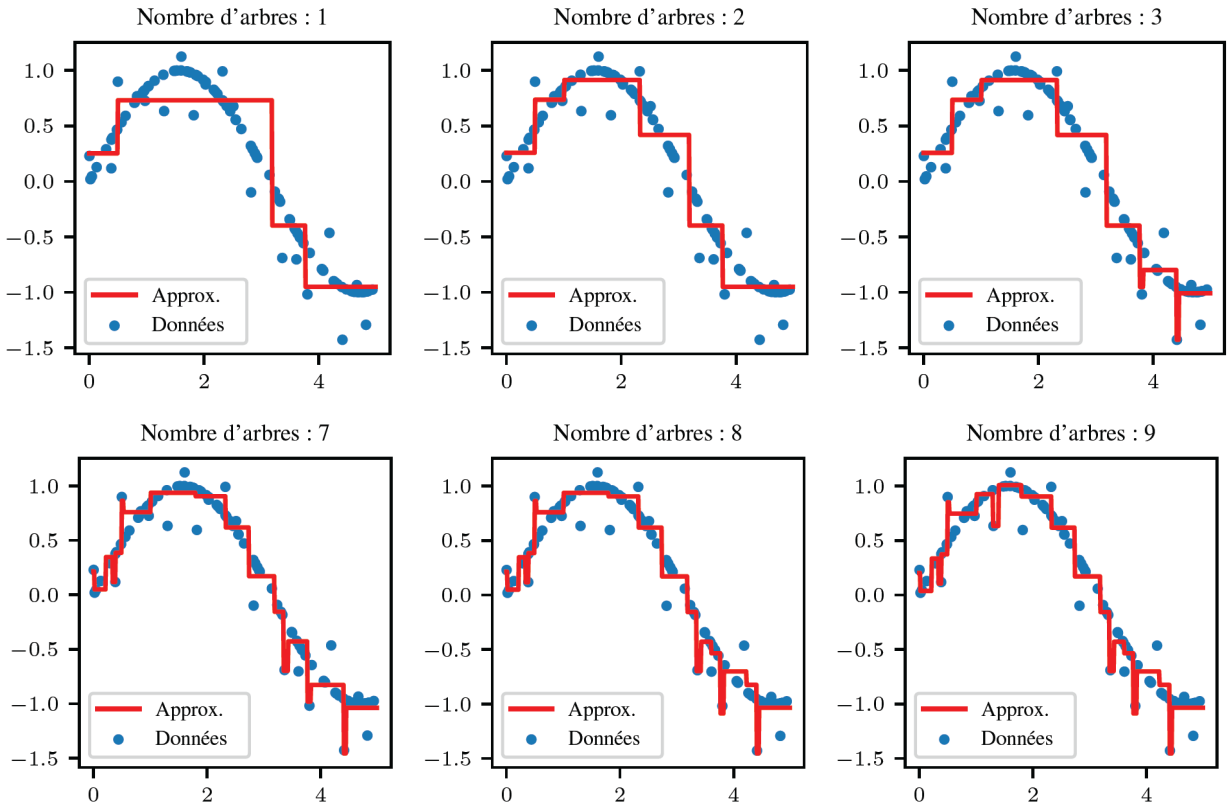


Figure 6.19 – Performances d'Ababoost dans une tâche de régression. Chaque classifieur faible est ici un arbre de profondeur 2.

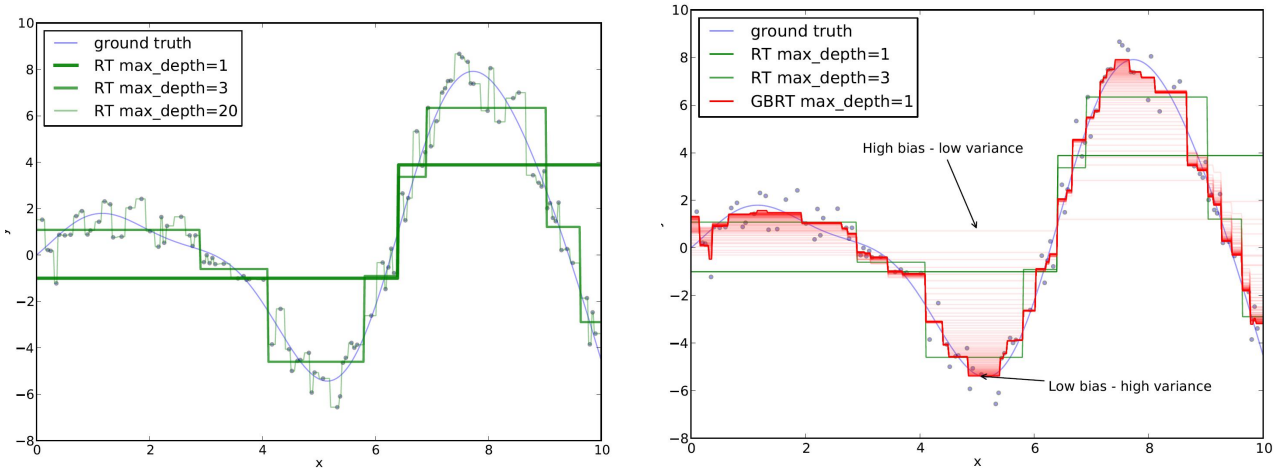


Figure 6.20 – Performances de Gradient Boosting dans une tâche de régression. À gauche, des arbres de décision, seuls, en régression (profondeur 1, 3, 20). À droite, une exécution de Gradient Boosting avec des arbres de profondeur 1.

6.4.7 Conclusion et bibliographie sommaire

- On utilise souvent Adaboost et Gradient Boost avec des arbres CART peu profonds (classifieurs faibles) : $\mathscr{H}$ est un ensemble de combinaisons linéaires d'arbres de décision.
- Gradient Boosting stochastique : à chaque itération, on entraîne le classifieur sur un sous échantillon aléatoire de $\mathscr{D}_n$ (sans remise).

- Gradient Boosting avec $\alpha = 1$ et perte Boosting ~ Adaboost.

Bibliographie :

- Freund, Schapire. Experiments with a new boosting algorithm. roceedings of the 13th Inter. Conf. on Machine Learning. 1996.
- Freund, Shapire. Boosting. MIT Press. 2012.
- Friedman. Greedy function approximation : A gradient boosting machine. Annals of Statistics. 2001.
- Friedman, Hastie, Tibshirani. Additive logistic regression: A statistical view of boosting. Annals of statistics. 2000.
- Chen, Guestrin. XGBoost: A Scalable Tree Boosting System. 2016.
- + Chapitre 8: Tree-Based Methods (p.331-363) Intro to Statistical Learning with Python.
- + Chapitre 7 du polycopié de Frédéric Sur.

Chapitre 7

Méthodes à base d'agrégation

7.1 introduction et motivation

Les méthodes d'agrégation ("ensemble learning") sont des techniques d'apprentissage statistique qui combinent plusieurs modèles de base afin d'améliorer les performances globales. Les principales méthodes sont : Bagging, Boosting et Stacking. Elles permettent d'améliorer la robustesse et les performances des modèles d'apprentissage statistique en combinant les forces de plusieurs modèles de base.

L'idée sous jacente derrière les techniques d'agrégation est d'essayer de réduire la variance des modèles, sans augmenter le biais. Il est acquis que la moyenne d'estimateurs réduit la variance dès lors que les modèles sont indépendants. Mais des modèles entraînés à partir de même données ne sont pas indépendants. Le Boosting utilise par exemple des techniques de rééchantillonnage, dont nous rappelons quelques résultats dans la section suivante, pour pallier ce problème de dépendance des données.

7.2 Boostrap et rééchantillonnage

Le Bootstrap a été introduit par Efron en 1979. Il s'agit de créer des échantillons aléatoires permettant de simuler une loi lorsque l'on n'a pas beaucoup d'échantillons disponibles. Le principe est de générer des échantillons qui ressemblent à l'échantillon de départ.



Figure 7.1 – L'expression Bootstrap vient des aventures du baron de Münchhausen, de Rudolph Raspe (1785) : "to pull oneself up by one's bootstraps" (se hisser en tirant sur les languettes de ses bottes ~ se sortir seul d'un situation difficile).

Rappels sur les estimateurs "plug-in" :

- $\mathscr{D}_n = \{Z_1, ..., Z_n\}$ n-échantillon de v.a.i.i.d. $Z_i = (X_i, Y_i)$.
- X_i observations issues d'une v.a. X : données, variables explicatives.
- Y_i issues d'une v.a. Y, catégories des X_i : étiquettes ou labels.
- $X \in \mathbb{X}$, $Y \in \mathbb{Y}$.
- $\mathbb{P}_\theta$ proba sur $\mathscr{E} = \mathbb{X} \times \mathbb{Y}$: loi inconnue de (X, Y) et (X_i, Y_i).
- $\theta = \theta(\mathbb{P})$ paramètre inconnu dépendant de $\mathbb{P}$.

- $T = T_n = T(X_1, ..., X_n) = T(\mathbb{P})$ estimateur de θ.
- Comme la loi de X est inconnue, on utilise la mesure empirique :

$$\mathbb{P}_n = \frac{1}{n} \sum_{i=1}^{n} \delta_{X_i} \tag{1}$$

On sait (théorèmes de Glivenko-Cantelli, de la limite centrale fonctionnelle et de Kolmogorov-Smirnov) que $\mathbb{P}_n$ est fortement et uniformément consistant pour $\mathbb{P}$.

Dans un estimateur plug-in, on remplace $\mathbb{P}$ par $\mathbb{P}_n$ dans l'expression de $\theta = T(\mathbb{P})$:

$$\widehat{\theta} = T(\mathbb{P}_n) = T(X_1, ..., X_n) \tag{2}$$

Exemple : $\theta = \mathbb{E}[X] = \int x\mathbb{P}(dx) \Rightarrow \widehat{\theta} = \overline{X}_n = \frac{1}{n}\sum_i X_i = \int x\mathbb{P}_n(dx)$

$\mathbb{P}_n$ est la mesure discrète qui sélectionne de façon uniforme chaque observation X_i avec probabilité $1/n$.

Les estimateurs plug-in ont de bonnes propriétés : $||\mathbb{P}_n - \mathbb{P}||_\infty = \sup_t |F_n(t) - F(t)| \longrightarrow 0$ p.s.

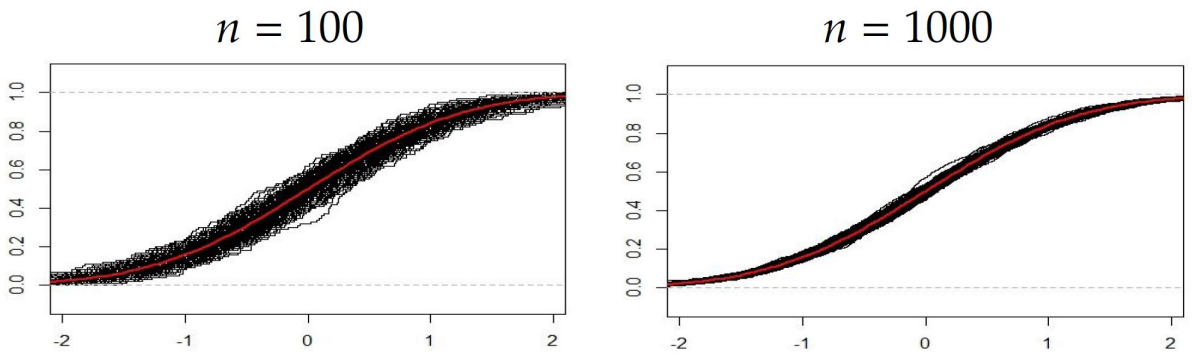


Figure 7.2 – Convergence uniforme des fonctions de distributions empiriques (en noir) d'échantillons de taille n vers la fonction de répartition théorique (en rouge) de la loi initiale à densité. Crédit : Arthur Charpentier.

Principe du Bootstrap.

On remplace $\mathbb{P}$ par $\mathbb{P}_n$ et on ré-échantillonne à partir de $\mathbb{P}_n$.

Soit $(X_1^\star, ..., X_n^\star)$ l'échantillon bootstrap issu de $(x_1, ..., x_n)$.

Dans le Bootstrap naïf, les $X_i^\star$ sont tirés de façon uniforme et avec remise parmi les $(x_1, ..., x_n)$ (~ tirages avec remise dans une urne). En quelque sorte, on échantillonne l'échantillon initial $(X_1, ..., X_n)$.

• Dans le "monde réel", on estime θ par $\widehat{\theta} = T(\mathbb{P}_n) = T(X_1, ..., X_n)$.

• Dans le monde bootstrap, on estime θ par $\theta^\star = T(X_1^\star, ..., X_n^\star)$.

Ceci n'est valide que si la loi de $\theta^\star$ approche bien celle de $\widehat{\theta}$, c'est à dire si $\mathcal{L}(\theta^\star | X_1, ..., X_n)$ et $\mathcal{L}(\widehat{\theta})$ ont même limite. Les conditions de validité dépendent également de la régularité des fonctions en jeu :

- La loi limite de $T(\mathbb{P}_n)$ doit dépendre continûment de $\mathbb{P}$.
- On doit avoir convergence uniforme en $\mathbb{P}$ dans un voisinage d'une mesure.
- Il y a des conditions de régularité (continuité, différentiabilité) dans l'espace des mesures de probabilités (la variable est une mesure de probabilité).

Cette technique fonctionne bien si les lois sont bien approchées par des gaussiennes, mais fonctionne mal typiquement pour les lois extrêmes (min, max, quantiles) et/ou pour des fonctions irrégulières de $\mathbb{P}$.

Exemple : pour $T(X_1, ..., X_n) = \overline{X}$, si $\mathbb{E}[X^2] < \infty$, $\overline{X}^\star$ covnerge bien vers $\mathbb{E}[X]$.

Exemple : Pour $T(X_1,...,X_n) = \max(X_1,...,X_n)$ si $X \sim \mathscr{E}$ loi exponentielle, les échantillons bootstrap ne converge pas.

Il est enfin nécessaire de vérifier les moments d'ordre 2, et de toujours vérifier par des simulations le comportement bootstrap.

- On peut construire plusieurs échantillons bootstrap de façon à simuler plusieurs tirages : n^n échantillons bootstrap différents sont possibles avec une taille initiale de n.
- En pratique le nombre B de tirages est choisi entre B = 200 et B = 1000.
- Les échantillons sont non indépendants, mais sont i.i.d. conditionnellement à $\mathscr{D}_n$.

$$\begin{cases} \theta_1^\star = T(X_{11}^\star,...,X_{1n}^\star) \\ ... \\ \theta_b^\star = T(X_{b1}^\star,...,X_{bn}^\star) \\ ... \\ \theta_B^\star = T(X_{B1}^\star,...,X_{Bn}^\star) \end{cases}$$

Les vecteurs $\theta_b^\star$ pour $b = 1,...,B$, sous $\mathbb{P}_n$, sont des réplications bootstrap de $\widehat{\theta} = T(X_1,...,X_n)$.

Il y a une double approximation : on approche $\mathbb{P}$ et θ par $\mathbb{P}_n$ et $\widehat{\theta}$, puis par $\mathbb{P}_n^\star$ et $\theta^\star$. Sachant $\mathscr{D}_n$, $\mathbb{P}_n^\star$ est la loi du tirage uniforme avec remise dans une urne contenant $\{x_1,...,x_n\}$.

Les quelques résultats ci-dessus ne constituent absolument pas un cours sur le bootstrap (qui nécessiterait une trentaine d'heures à lui tout seul) mais juste quelques rappels (mathématiquement non rigoureux) pour pouvoir présenter le Bagging.

À compléter...

7.3 Bagging : Bootstrap Agregating

7.3.1 Principe du Bagging

Le principe des méthodes de Bagging (pour Bootstrap Aggregating), découvertes par Leo Breiman [Breiman, 1996], est d'agréger plusieurs classifieurs faibles pour obtenir un classifieur possédant de bonnes propriétés. Contrairement au Boosting, l'agrégation n'est pas obtenue de façon itérative mais en effectuant une moyenne pondérée (pour la régression) ou un vote (pour la classification) parmi les estimateurs de base.

Chaque estimateur de base est entraîné sur des échantillons bootstrap $\mathscr{D}_{nb}$ issus de l'échantillon initial, avec $b = 1,...,B$, donc par tirage aléatoire avec remise.

Nous allons présenter le Bagging à la fois pour une tâche de régression et pour une tâche de classification.

Considérons B estimateurs $g_1,...,g_B$. Pour une tâche de régression posons comme modèle : $Y = g(X) + \varepsilon$ avec $g(x) = \mathbb{E}[Y|X = x]$.

Pour chaque échantillon bootstrap $\mathscr{D}_{nb}$ issu de l'échantillon initial, on ajuste un régresseur $g_b(x)$. Le régresseur bagging est

$$\widehat{g}_B(x) = \frac{1}{B}\sum_{b=1}^{B} g_b(x) \tag{3}$$

En régression, l'algorithme est le suivant :

0 • En entrée : l'échantillon $\mathscr{D}_n$, un régresseur ou classifieur g et le nombre B.

Puis pour $b = 1,...,B$:

1 • Tirer un échantillon bootstrap $\mathscr{D}_{nb}$ dans $\mathscr{D}_n$.

2 • Ajuster un régresseur ou classifieur $g_b(x)$ sur cet échantillon.

3 • En sortie, on obtient l'estimateur Bagging suivant

$$\widehat{g}_B(x) = \frac{1}{B}\sum_{b=1}^{B} g_b(x) \tag{4}$$

$g_b(x)$ est une v.a. qui dépend de $\mathscr{D}_{nb} \subset \mathscr{D}_n$ (tiré aléatoirement). Attention ! Les $g_b(x)$ ne sont pas indépendants, mais conditionnellement à $\mathscr{D}_n$ ils sont i.i.d.

$$\mathbb{E}[\widehat{g}_{\mathrm{B}}(x)] = \mathbb{E}[g_b(x)] \tag{5}$$

En général, à cause de la dépendance entre les $g_b(x)$,

$$\mathbb{V}\left(\widehat{g}_{\mathrm{B}}(x)\right) \neq \frac{1}{\mathrm{B}}\mathbb{V}(g_b(x)) \tag{6}$$

Malgré tout, le tirage bootstrap atténue la dépendance en introduisant une nouvelle source d'aléa.

On passe de $\mathscr{D}_n$ à $\mathscr{D}_{nb}$ par un tirage uniforme avec remise. Un vecteur *aléatoire* d'indices $\mathrm{I}_b \subset [[1..n]]^n$ est généré pour chaque échantillon.

$$g_b(x) = g_b(x, \mathscr{D}_{nb}, \mathscr{D}_n) = g_b(x, \mathrm{I}_b, \mathscr{D}_n) \tag{7}$$

Conditionnellement à $\mathscr{D}_n$, les $g_b(x)$ sont des v.a.i.i.d. (dépendant de I_b).

$$\lim_{\mathrm{B}\to+\infty} \widehat{g}_{\mathrm{B}}(x) = \mathbb{E}_{\mathrm{I}}[g(x,\mathrm{I})|\mathscr{D}_n] \tag{8}$$

Escroquerie : où est passé le b dans I_b ?

$$\mathbb{V}\left(\widehat{g}_{\mathrm{B}}(x)\right) = \rho(x)\mathbb{V}(g_b(x)) + \frac{1-\rho(x)}{\mathrm{B}}\mathbb{V}(g_b(x)) \tag{9}$$

Avec $\rho(x) = \mathrm{cov}(g_b(x), g_{b'}(x))$. Quand $\mathrm{B} \to +\infty$, de façon peu rigoureuse, on a :

$$\widehat{g}(x) = \lim_{\mathrm{B}\to+\infty} \mathbb{V}\left(\widehat{g}(x)\right) = \rho(x)\mathbb{V}(g_b(x)) \tag{10}$$

Plus la corrélation $\rho(x)$ est faible, plus la variance diminue. L'objectif du bootstrap est d'ajouter de l'aléatoire pour (presque) créer de l'indépendance : le hasard fait bien les choses. Il faut donc agréger des estimateurs sensibles aux perturbations de $\mathscr{D}_n$, de telle sorte de varier les sous-échantillons revienne à varier les performances de chaque estimateur de base. Les arbres de décision sont de bons candidats pour constituer les modèles de base, car ils sont typiquement instables.

Si chaque échantillon tiré est de taille b_n avec $b_n \to +\infty$ et $b_n/n \to 0$, on peut montrer que $\widehat{g}(x)$ est universellement consistant.

7.3.2 Forêts aléatoires

Les forêts aléatoires ("Random Forest") ont été introduites par Breiman également. Il s'agit simplement de la méthode de Bagging, appliquée avec des arbres de décisions comme classifieurs faibles.

$$\widehat{\mathrm{T}}_{\mathrm{B}}(x) = \frac{1}{\mathrm{B}}\sum_{b=1}^{\mathrm{B}} \mathrm{T}_b(x)$$

- On sélectionne aléatoirement m variables de l'arbre parmi les d variables initiales.
- On cherche à diminuer la corrélation entre les arbres.
- On a deux nouvelles sources d'aléa : le tirage bootstrap de l'échantillon et le choix des m variables sur les arbres.
- Profondeur typique : 5 en régression, 1 en classification (heuristique).
- Valeur typique de m : $d/3$ en régression, $\sqrt{d}$ en classification (heuristique).
- Attention au biais : le biais ne diminue pas en Bagging.

7.3.3 Mesures de performances

Comme pour les autres méthodes d'apprentissage statistique, on mesure l'erreur de prédiction en régression par les moindres carrés et par la probabilité d'erreur en classification.

Utilisation par sous-ensemble d'apprentissage/validation ou par validation croisée.

Le bootstrap permet une estimation de l'erreur par OOB (Out of Bag).

Pour tout (X_i, Y_i) de $\mathscr{D}_n$, soit $\mathbb{J}_i$ le sous-ensemble des arbres qui ne contient pas l'observation i. La prévision de Y en X_i est

$$\widehat{Y}_i = \frac{1}{|\mathbb{J}_i|} \sum_{b \in \mathbb{J}_i} T(X_i, \mathscr{D}_{nb}).$$

L'erreur de prédiction OOB est : $\frac{1}{n}\sum_{i=1}^{n}(\widehat{Y}_i - Y_i)^2$.

La probabilité d'erreur OOB est : $\frac{1}{n}\sum_{i=1}^{n}\mathbb{1}_{[\widehat{Y}_i \neq Y_i]}$.

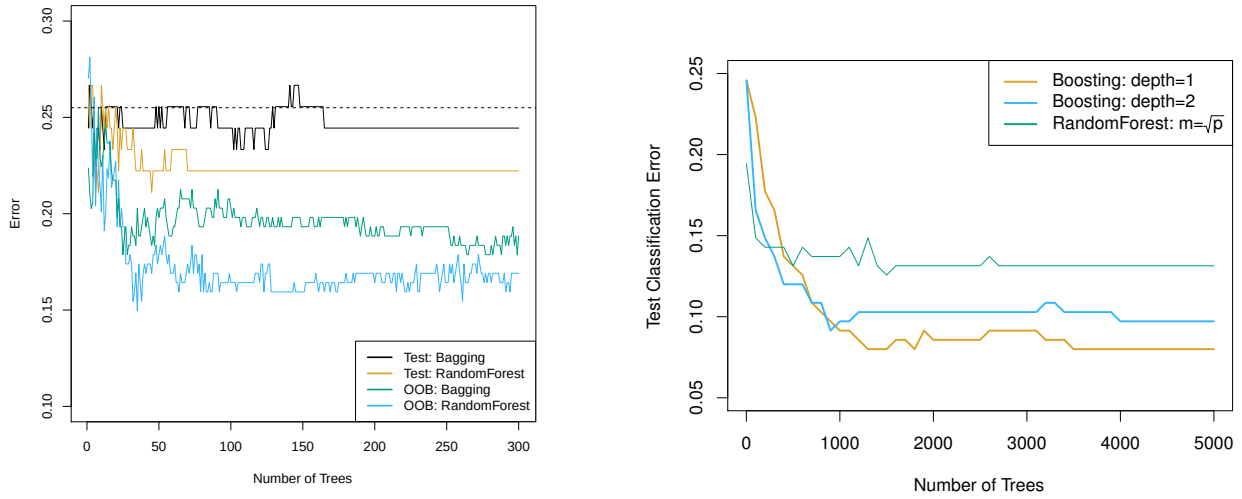


Figure 7.3 – À gauche : illustration des performances du Bagging et de forêts aléatoires. À droite : comparaison des performances en Boosting et Bagging en fonction du nombre d'arbres. Crédit : Introduction to Machine Learning with Python. James, Witten, Hastie, Tibshirani. Figures du chapitre 8.

7.4 Le Boosting vu comme une méthode d'agrégation

Comme nous l'avons vu au chapitre précédent, le Boosting est une méthode itérative qui combine plusieurs modèles de base faibles pour créer un modèle fort. Chaque modèle de base est entraîné pour corriger les erreurs du modèle précédent.

On Entraînee un modèle de base sur l'échantillon originel, puis on ajuste les poids des échantillons en fonction des erreurs du modèle précédent. On entraîne un nouveau modèle de base sur ces données pondérées, puis on répéte les étapes jusqu'à ce qu'un critère d'arrêt soit atteint.

On peut donc considérer le Boosting comme une méthode d'agrégation (itérative).

7.5 Stacking

7.5.1 Principe du stacking

Le stacking est une technique d'agrégation d'un petit nombre d'estimateurs (classifieurs ou régresseurs) servant de modèles de base à un méta-modèle. Contrairement aux techniques précédentes, les estimateurs de base ne sont pas des estimateurs faibles et ils sont éventuellement de nature différente.

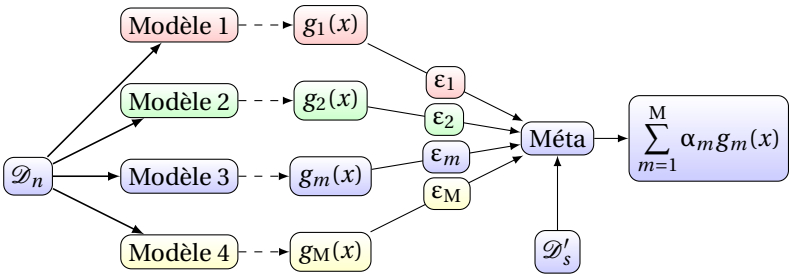


Figure 7.4 – Illustration du Stacking. Chaque classifieur niv.1 g_m fournit au méta-modèle son erreur de régression / classification ε_m. Le méta-modèle initialise les poids α_m à partir de ε_m. Il met à jour les poids lors de son entraînement sur $\mathscr{D}'_s$.

Une approche naïve consisterait à créer une combinaison linéaire pondérée de bons estimateurs en espérant qu'elle soit meilleure que chaque estimateur de base. Cela n'est vrai que si les poids minimisent l'erreur théorique (inconnue). Mais si l'on minimise l'erreur d'entraînement, on se retrouve rapidement en situation de surapprentissage. Il y a donc ,écessité d'une structure à deux niveaux et la combinaison linéaire (CL) sera performante si les estimateurs sont individuellement performants, tout en étant très différents les uns des autres. Le stacking se distingue donc des autres techniques d'agrégation par son approche hiérarchique : dans un premier temps, plusieurs modèles de base sont entraînés sur les données disponibles. Ensuite, un méta-modèle, souvent plus simple, est entraîné pour apprendre à combiner au mieux les prédictions des modèles de base. L'idée sous-jacente est que le méta-modèle peut capturer les forces et les faiblesses des différents modèles de base, conduisant ainsi à une prédiction finale plus robuste et précise.

Précisons : le premier niveau est formé de plusieurs modèles de base, chacun entraîné séparément sur les mêmes données. Chaque modèle donne des prédictions différentes à une donnée. Le second niveau est appelé méta-modèle. Les prédictions du niveau 1 servent de données au niveau 2. Le méta-modèle est entraîné sur ces prédictions mais pas sur l'échantillon du niveau 1. Les poids initiaux di niveau 2 sont fonction des performances de chaque modèle du niveau 1. Ces poids sont modifiés lors de l'entraînement du méta-modèle et celui-ci propose une prédiction finale. Il est important que les différents modèles soient construits de façon indépendantes.

Exemples de modèles de base : régression logistique, arbre de décision, k-ppv, SVM, réseaux de neurones, etc.

Quelques remarques.

• Le gain vient de la diversité des modèles !

• Sur quelles données d'apprentissage doit-on entraîner les algorithmes ?

- Bagging : sur des versions modifiées du même échantillon (tirages avec remise).
- Boosting : sur les même données mais avec pondération des individus.
- Stacking niveau 1 : sur le même échantillon à tous les classifieurs de niveau 1.
- Stacking niveau 2 : sur un autre échantillon pour entraîner le niveau 2.

• Il est nécessaire d'utiliser une validation croisée pour éviter le surapprentissage entre niveaux et il est important de garder peu de modèles pour ne pas avoir trop de paramètres ou de complexité.

• Le stacking peut réduire à la fois la variance et le biais.

7.6 Comparaison des méthodes

	Bagging	Boosting	Stacking
Objectif	Réduire la variance	Réduire le biais	Augmenter les perf.
Modèles de base	Identiques	Identiques	Différents
Entraînement	Parallèle	Séquentiel	Au niveau du méta-modèle
Agrégation	Vote majoritaire ou moyenne	Moyenne pondérée	Moyenne pondérée

Nous reprenons ici un type de tableau utilisé par Laurent Rouvière dans ces cours pour résumer les points forts et les points faibles des différentes méthodes.

	Linéaire	SVM	RN	Arbre	Forêt	Boosting
Performances	■	■	■	×	✓	✓
Calibration		×	×	✓	✓	✓
Coût en calcul	■	×	×	✓	✓	✓
Interprétation	✓	×	×	■	×	×

Bibliographie :

- Bagging Predictors. Breiman. Machine Learning. 1996.
- Forêts aléatoires. Genuer. Thèse de doctorat. 2010.
- Stacked Generalization. D. Wolpert. Neural Networks. 1992.
- Super Learner. Van der Laan, Polley, Hubbard. Statistical Applications in Genetics. 2007.
- + Chapitre 5 (bootstrap p.212) Intro to Statistical Learning with Python.
- + Chapitre 8 (Bagging et random forests p.343) Intro to Statistical Learning with Python.
- + Chapitre 7 (Bagging p.88) du polycopié de Frédéric Sur.

Bibliographie

[Arlot and Celisse, 2010] Arlot, S. and Celisse, A. (2010). A survey of cross-validation procedures for model selection. *Statistics Surveys*, 4(none):40 – 79.

[Bach, 2024] Bach, F. (2024). *Learning from first principles.* MIT Press.

[Breiman, 1996] Breiman, L. (1996). Bagging predictors. *Mach. Learn.*, 24(2):123–140.

[Breiman et al., 1984] Breiman, L., Friedman, J. H., Olshen, R. A., and Stone, C. J. (1984). *Classification and Regression Trees.* Wadsworth.

[Burges, 1998] Burges, C. J. (1998). A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167.

[Chen and Guestrin, 2016] Chen, T. and Guestrin, C. (2016). Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, page 785–794. Association for Computing Machinery.

[Freund and Schapire, 1996] Freund, Y. and Schapire, R. E. (1996). Experiments with a new boosting algorithm. In *International Conference on Machine Learning*, pages 148–156.

[Friedman, 2001] Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, 29(5):1189 – 1232.

[Girardon et al., 2020] Girardon, J., Le Cun, Y., and Dehaene, S. (2020). *la plus belle histoire de l'intelligence.* Points documents.

[Hughes, 1968] Hughes, G. (1968). On the mean accuracy of statistical pattern recognizers. *IEEE Transactions on Information Theory*, 14(1):55–63.

[Rolls and Deco, 2010] Rolls, E. and Deco, G. (2010). *The Noisy Brain, Stochastic Dynamic as a Principle of Brain Function.* Oxford Univsersity Press.

[Schölkopf and Smola, 2002] Schölkopf, B. and Smola, A. J. (2002). *Learning with kernels : support vector machines, regularization, optimization, and beyond.* Adaptive computation and machine learning. MIT Press.

[Sur, 2024] Sur, F. (2024). Introduction à l'apprentissage automatique. Polycopié de l'école des Mines de Nancy.

[Van Campenhout, 1978] Van Campenhout, J. M. (1978). On the peaking of the hughes mean recognition accuracy the resolution of an apparent paradox. *IEEE Transactions on Systems, Man, and Cybernetics*, 8(5):390–395.

[Wolpert and Macready, 1997] Wolpert, D. and Macready, W. (1997). No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82.