École nationale
de la statistique
et de l'analyse
de l'information

Introduction à l'apprentissage statistique

Cours de Mastère Spécialisé

Claude PETIT

8 août 2025

Table des matières

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

À venir...

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.

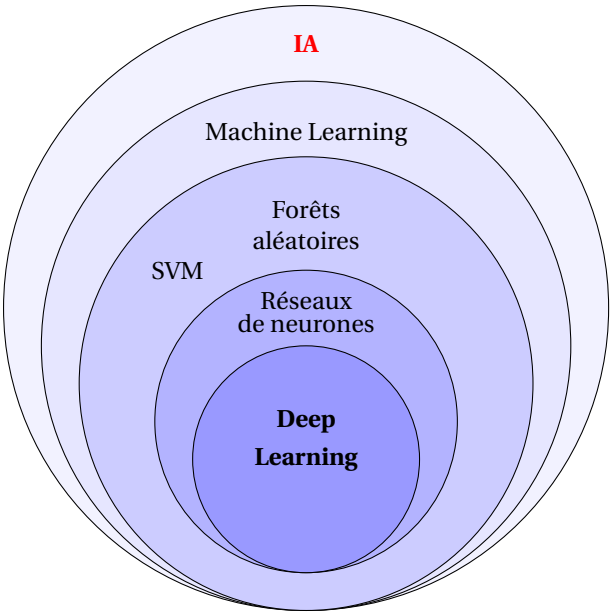


FIGURE 2.1 – Différents sous domaines de l'IA.

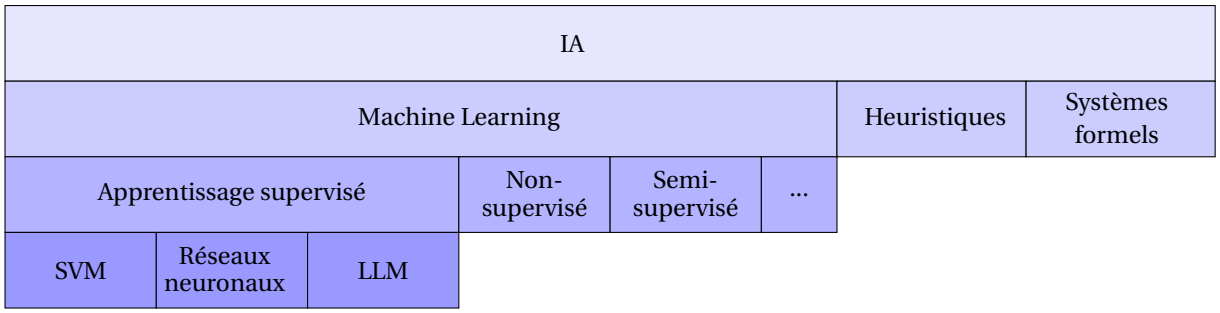


FIGURE 2.2 – Les différents sous domaines de l'IA.

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.

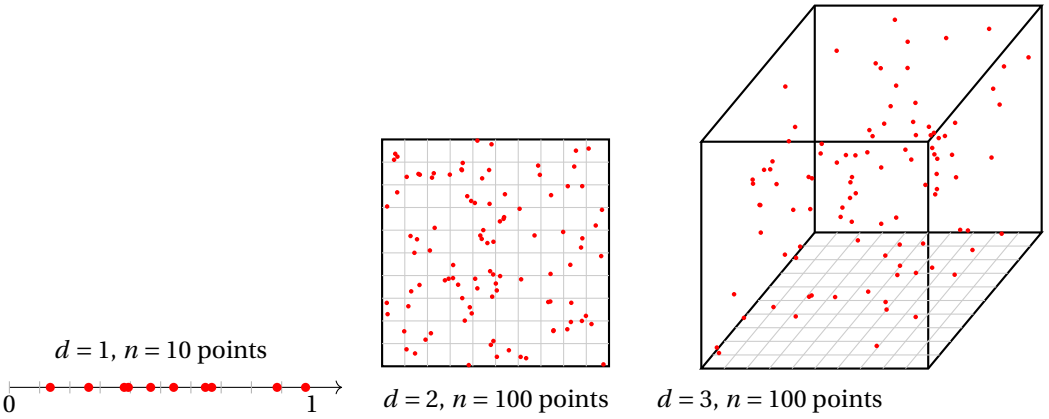


FIGURE 2.3 – Partition du cube unité et échantillonnage en dimension $d = 1, 2, 3$. Taille de grille = $1/10$. Le nombre de points nécessaires explose avec la dimension.

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 :

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

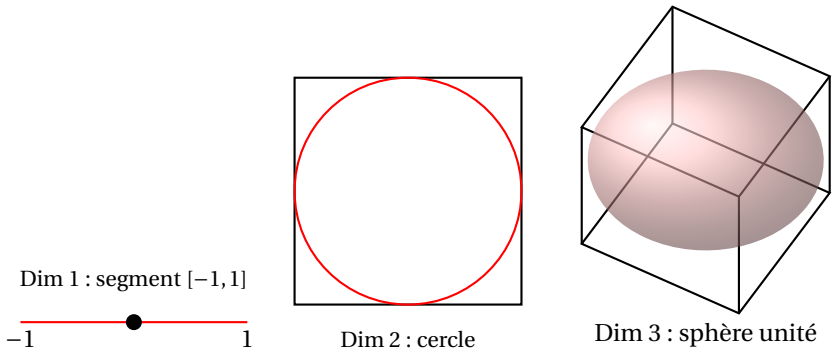


FIGURE 2.4 – Volume de la boule unité en dimension $d = 1, 2, 3$.

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.

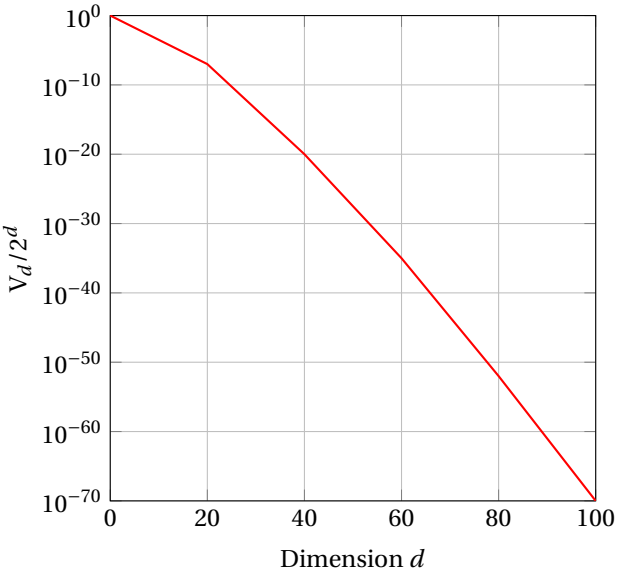


FIGURE 2.5 – Volume de la boule unité en dimension d rapporté au volume du cube de côté 2.

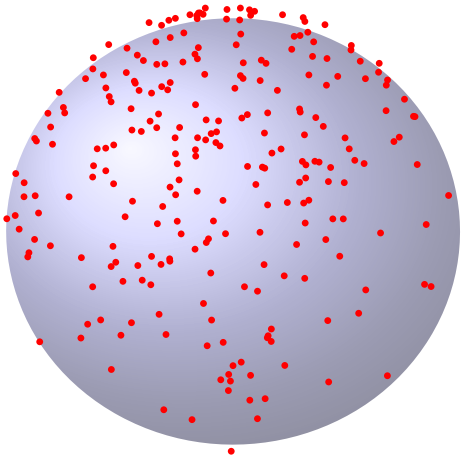


FIGURE 2.6 – Points aléatoires générés de façon uniforme dans l'espace, se concentrant à la surface de la sphère.

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

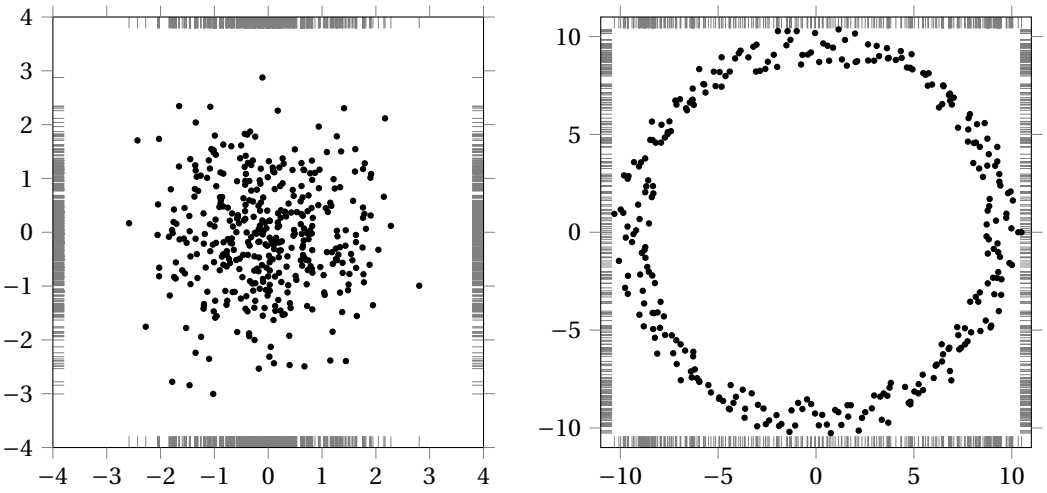


FIGURE 2.7 – Gauche : échantillon gaussien standard en dimension $d = 2$. Droit : Échantillon gaussien standard en dimension $d = 100$, projeté dans un sous-espace de dimension 2.

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, demeure donc un paradoxe non expliqué.

On pourra se reporter au contenu du blog suivant pour plus de détails :
https://37steps.com/2322/hughes-phenomenon/.

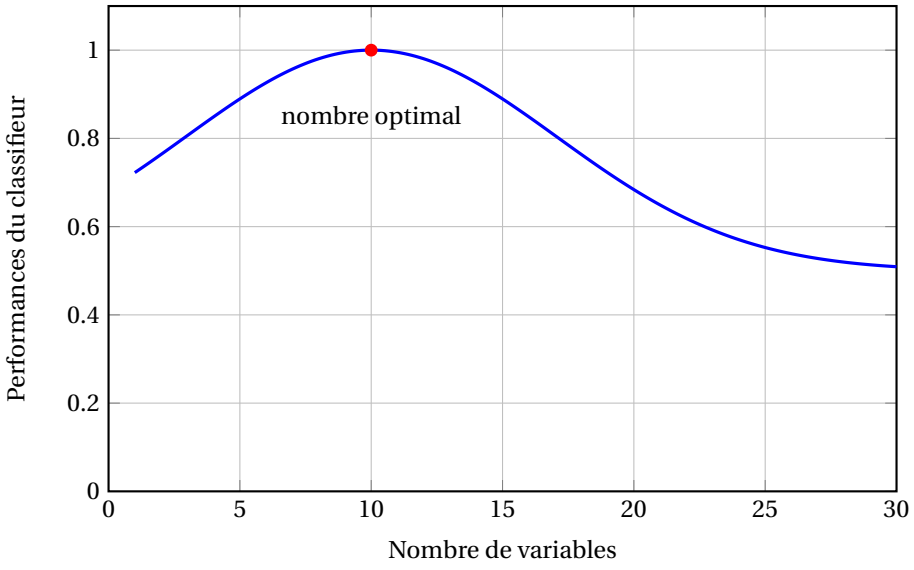


FIGURE 2.8 – Illustration du paradoxe de Hughes

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.

Démonstration.

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

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 :

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

Démonstration. 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

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

Démonstration. 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 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. 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 :

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

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 le risque théorique diminue, puis augmente à nouveau ; nous allons voir pourquoi dans le paragraphe suivant.

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.

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.

L'erreur d'estimation (ou risque stochastique), qui est homogène à une variance (et qui est aléatoire), 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, mais l'erreur stochastique devient en moyenne plus grande. 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 $\mathcal{G}$.

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

Démonstration. 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 $1-\delta$,

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

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

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

De

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

on tire

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

Mais

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

Posons alors $\mathrm{V}_i = h_m(\mathrm{X}_i,\mathrm{Y}_i) = l(\mathrm{Y}_i, g_m(\mathrm{Y}_i)) \in [0,1]$. Ces va sont i.i.d. car les $(\mathrm{X}_i,\mathrm{Y}_i)$ 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 $\mathrm{R}_n(g_m)$ est égale à $\mathrm{R}(g_m)$. On peut alors poser $a = 0$ et $b = 1$ et appliquer l'inégalité de Hoeffding :

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

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

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

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

□

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.

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

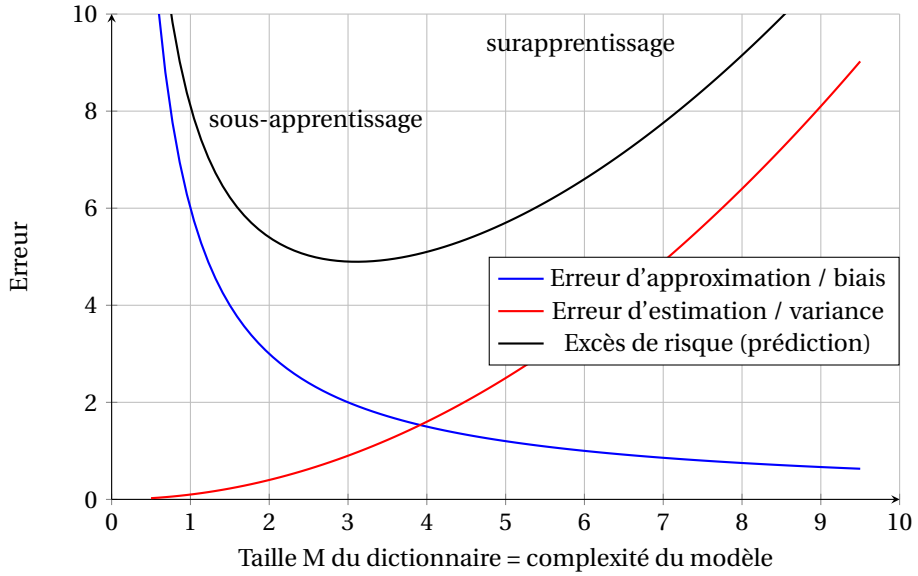


FIGURE 3.1 – Un exemple de dilemne biais / variance. En bleu, l'erreur d'approximation pour un dictionnaire de taille M. En rouge, l'erreur d'estimation, en noir le risque théorique ou excès de risque. La zone de sous-apprentissage caractérise un biais fort et une variance faible, tandis que la zone 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.

La complexité des modèles et la présentation du dilemme biais-variance 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

3.3 Sélection de modèles et validation croisée

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

$R(g^\star)$ est le risque de Bayes, que nous avons aussi appelé risque optimal, risque théorique ou encore risque de prédiction. $R_n = R_n(g)$ 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 $R(\widehat{g}_n)$ est possiblement éloigné de $R_n(\widehat{g}_n)$. Ces deux quantités dépendent de l'échantillon $\mathcal{D}_n$ au travers de $\widehat{g}_n$, ce sont donc des variables aléatoires fonctions de (X_i, 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 $l(Y_i, \widehat{g}_n(X_i))$ 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 découpage en deux ne suffit plus, car les deux sous-échantillons deviennent alors dépendants et il est nécessaire

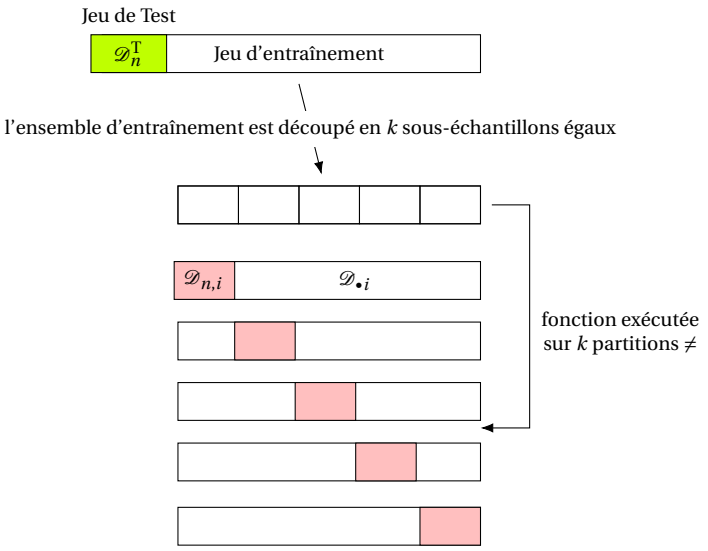


FIGURE 3.2 – Illustration de la validation croisée k-Fold pour $k = 5$. La base initiale est $\mathscr{D}_n$, partitionnée en trois sous-échantillon : $\mathscr{D}_n^{\mathrm{T}}$ (en vert, fixe durant tout le processus), $\mathscr{D}_{n,i}$ (en rose) et $\mathscr{D}_{\bullet i}$ qui varient à chaque itération. Souvent, $k = 5$ ou $k = 10$. $k = n$ est la méthode « leave-one-out ».

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. 3.2) 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 $i = 1, \ldots, k$, 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) :

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

• 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).

Chapitre 4

Méthodes à base de partition

4.1 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

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

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

Théorème 7 — Minimiseur du risque empirique pour la régression MC

$$\widehat{g}_n(x) = \sum_{m=1}^{\mathrm{M}} \overline{\mathrm{Y}}_{\mathrm{A}_m} \mathbb{1}_{\mathrm{A}_m}(x) = \sum_{m=1}^{\mathrm{M}} \left(\frac{1}{\mathrm{N}_m} \sum_{i=1}^{n} \mathrm{Y}_i \mathbb{1}_{\mathrm{A}_m}(\mathrm{X}_i) \right) \mathbb{1}_{\mathrm{A}_m}(x) \tag{8}$$

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

$$\frac{1}{n} \sum_{i=1}^{n} l(\mathrm{Y}_i, g(\mathrm{X}_i)) = \frac{1}{n} \sum_{i=1}^{n} (\mathrm{Y}_i - g(\mathrm{X}_i))^2. \tag{9}$$

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

$$\mathrm{R}_n(g) = \frac{1}{n} \sum_{i=1}^{n} \sum_{m=1}^{\mathrm{M}} (\mathrm{Y}_i - g(\mathrm{X}_i))^2 \mathbb{1}_{\mathrm{A}_m}(\mathrm{X}_i) \tag{10}$$

$$= \frac{1}{n} \sum_{m=1}^{\mathrm{M}} \sum_{i=1}^{n} (\mathrm{Y}_i - a_m)^2 \mathbb{1}_{\mathrm{A}_m}(\mathrm{X}_i) \tag{11}$$

$$\min_{g \in \mathcal{G}} \mathrm{R}_n(g) = \min_{a \in \mathbb{R}^{\mathrm{M}}} \left(\frac{1}{n} \sum_{i=1}^{n} \sum_{m=1}^{\mathrm{M}} (\mathrm{Y}_i - g(\mathrm{X}_i))^2 \mathbb{1}_{\mathrm{A}_m}(\mathrm{X}_i) \right) \tag{12}$$

$$= \frac{1}{n} \sum_{m=1}^{\mathrm{M}} \min_{a \in \mathbb{R}^{\mathrm{M}}} \left(\sum_{i=1}^{n} (\mathrm{Y}_i - a_m)^2 \mathbb{1}_{\mathrm{A}_m}(\mathrm{X}_i) \right) \tag{13}$$

$$\sum_{i=1}^{n} (\mathrm{Y}_i - a_m)^2 \mathbb{1}_{\mathrm{A}_m}(\mathrm{X}_i) = \sum_{i=1}^{n} \mathrm{Y}^2 \mathbb{1}_{\mathrm{A}_m}(\mathrm{X}_i) - 2a_m \sum_{i=1}^{n} \mathrm{Y}_i \mathbb{1}_{\mathrm{A}_m}(\mathrm{X}_i) + a_m^2 \mathrm{N}_m. \tag{14}$$

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

$$a_m = \frac{1}{\mathrm{N}_m} \sum_{i=1}^{n} \mathrm{Y}_i \mathbb{1}_{\mathrm{A}_m}(\mathrm{X}_i). \tag{15}$$

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

$$\min_{g \in \mathcal{G}} \mathrm{R}_n(g) = \frac{1}{n} \sum_{m=1}^{\mathrm{M}} \min_{a_m = 0,1} \left(\sum_{i=1}^{n} \mathbb{1}_{[\mathrm{Y}_i \neq a_m]} \mathbb{1}_{\mathrm{A}_m}(\mathrm{X}_i) \right) \tag{16}$$

$$\widehat{a}_m \in \operatorname*{arg\,min}_{a=0,1} \sum_{i=1}^{n} \mathbb{1}_{[\mathrm{Y}_i \neq a]} \mathbb{1}_{\mathrm{A}_m}(\mathrm{X}_i) \tag{17}$$

et cette expression vaut 0 ou 1 selon que

$$\sum_{i=1}^{n} \mathbb{1}_{[\mathrm{Y}_i \neq 0]} \mathbb{1}_{\mathrm{A}_m}(\mathrm{X}_i) < \sum_{i=1}^{n} \mathbb{1}_{[\mathrm{Y}_i \neq 1]} \mathbb{1}_{\mathrm{A}_m}(\mathrm{X}_i) \tag{18}$$

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

$$\sum_{i=1}^{n} \left(\mathbb{1}_{[\mathrm{Y}_i \neq 0]} - \mathbb{1}_{[\mathrm{Y}_i \neq 0]} \right) \mathbb{1}_{\mathrm{A}_m}(\mathrm{X}_i) = \sum_{i=1}^{n} (2\mathrm{Y}_i - 1) \mathbb{1}_{\mathrm{A}_m}(\mathrm{X}_i) = 2\mathrm{N}_m (\overline{\mathrm{Y}}_{\mathrm{A}_m} - 1/2). \tag{19}$$

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

On peut résumer le résultat des deux théorèmes de la façon suivante : pour estimer $\mathbb{E}[\mathrm{Y}|\mathrm{X} = x]$ 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 $\mathbb{Y} = \{1, ..., \mathrm{K}\}$,

$$\forall x \in \mathbb{X},\ \widehat{g}_n(x) = \operatorname*{arg\,max}_{k=1,...,\mathrm{K}} \sum_{m=1}^{\mathrm{M}} \left(\frac{1}{\mathrm{N}_m} \sum_{i=1}^{n} \mathbb{1}_{[\mathrm{Y}_i = k]} \mathbb{1}_{\mathrm{A}_m}(\mathrm{X}_i) \right) \mathbb{1}_{\mathrm{A}_m}(x) \tag{20}$$

$$= \operatorname*{arg\,max}_{k=1,...,\mathrm{K}} \sum_{m=1}^{\mathrm{M}} w_{m,k} \mathbb{1}_{\mathrm{A}_m}(x). \tag{21}$$

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

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.

4.2 La méthode des k plus proches voisins

4.2.1 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 $m = 1, ..., \binom{n}{k}$, 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. 4.1 et Fig. 4.2) 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.

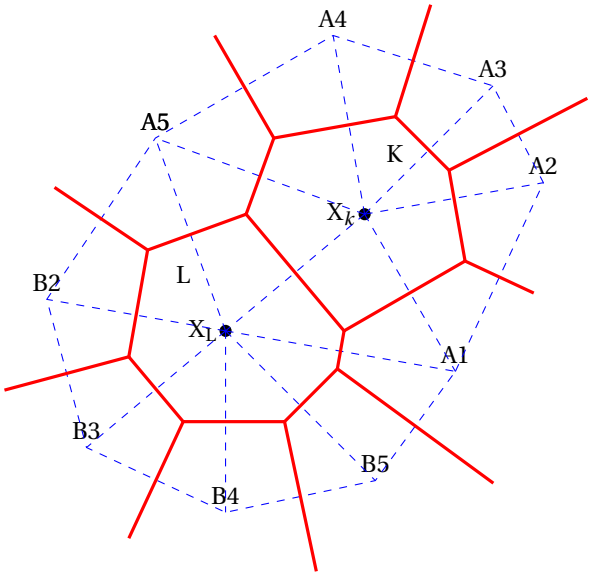


FIGURE 4.1 – k-ppv et cellule de Voronoï.

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.

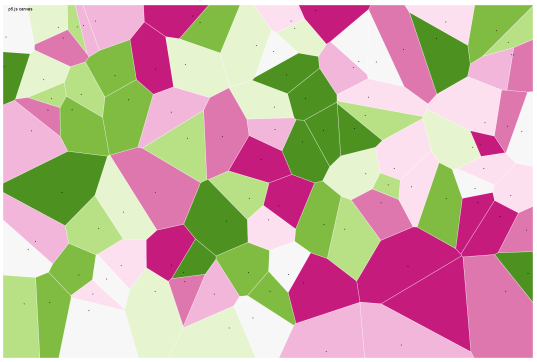


FIGURE 4.2 – Cellules de Voronoi pour $k = 1$.

FIGURE 4.3 – 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.

4.2.2 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. 4.4 illustre ce jeu de données et le résultat d'une classification des 3 espèces.

La figure Fig. 4.5 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.

4.3 Les arbres de décision et de régression

4.3.1 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).

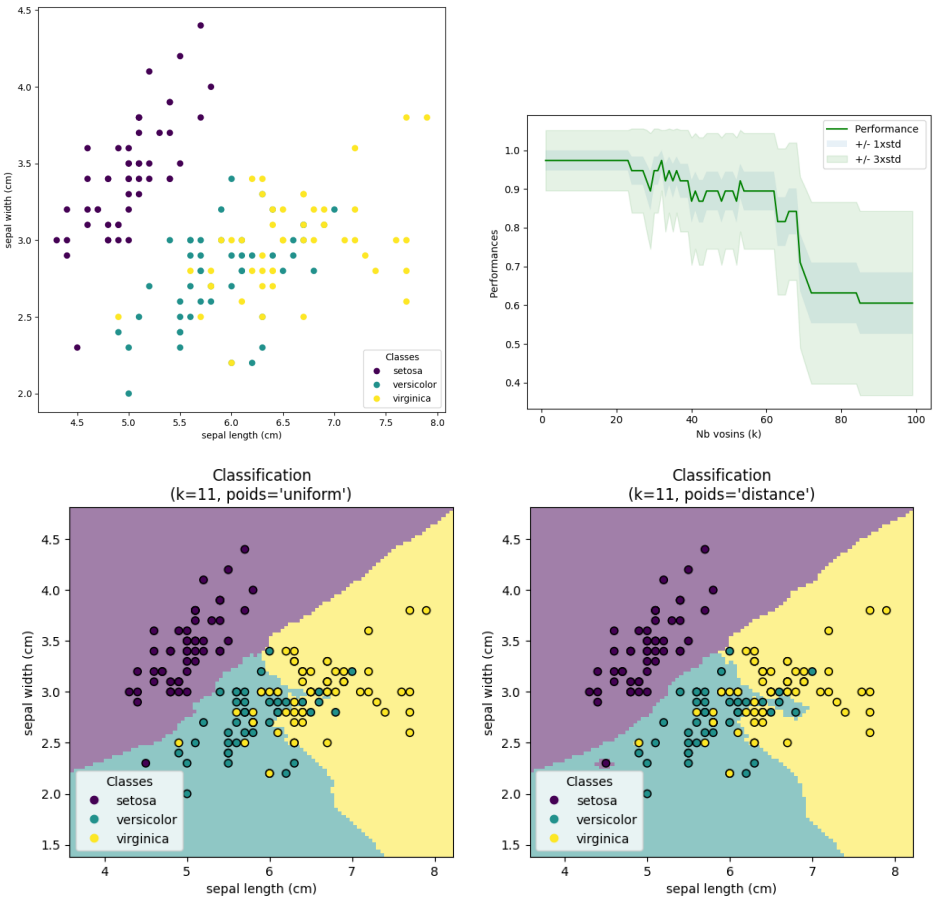


FIGURE 4.4 – Haut : à gauche, nuage de points représentant les iris de Fisher (une espèce par couleur) en fonction de la longueur et de la largeur de leur sépale. À droite, une courbe indiquant les performances de k-ppv sur la tâche de classification des iris, en fonction du nombre k. Bas : illustration 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.

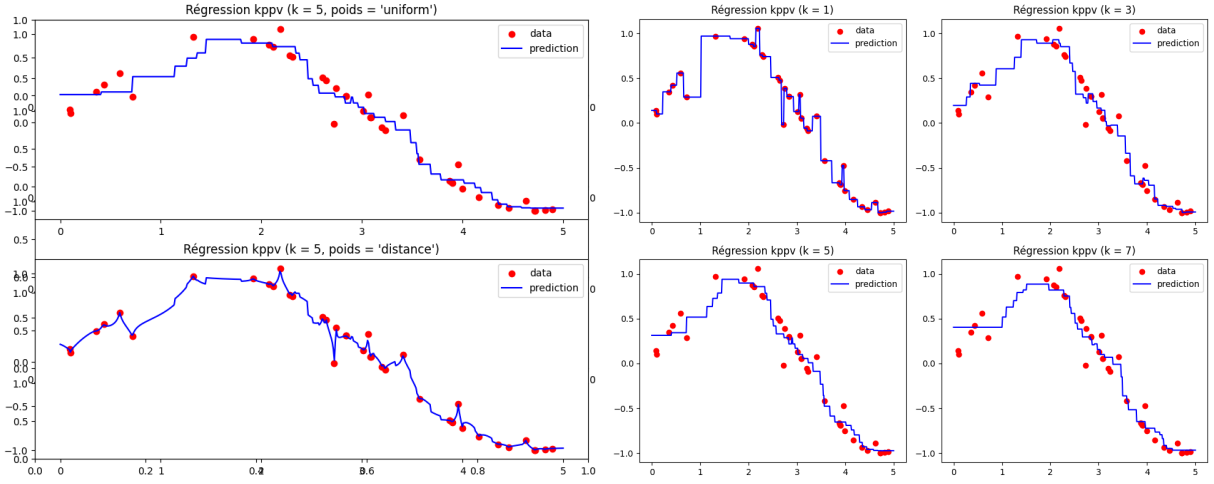


FIGURE 4.5 – Régression à l'aide de la méthode k-ppv. Un nuage de points (rouges) bruité est généré en perturbant aléatoirement les points d'une sinusoïde. La courbe bleue, issue de la méthode des k-ppv, doit reconstituer au mieux cette sinusoïde. Les deux courbes à gauche correspondent à $k = 5$ pour 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.

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. 4.6).

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.

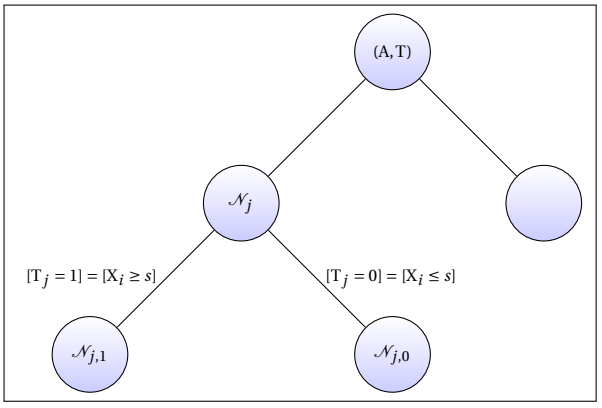


FIGURE 4.6 – Exemple d'arbre avec 5 nœuds et 4 arêtes. La racine A correspond à l'ensemble $\mathbb{X}$. À un nœud donné $\mathcal{N}_j$ de profondeur j est affecté un sous-ensemble de $\mathbb{X}$. Le test courant T_j est effectué sur les observations contenues dans $\mathcal{N}_j$, qui sont réparties 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.

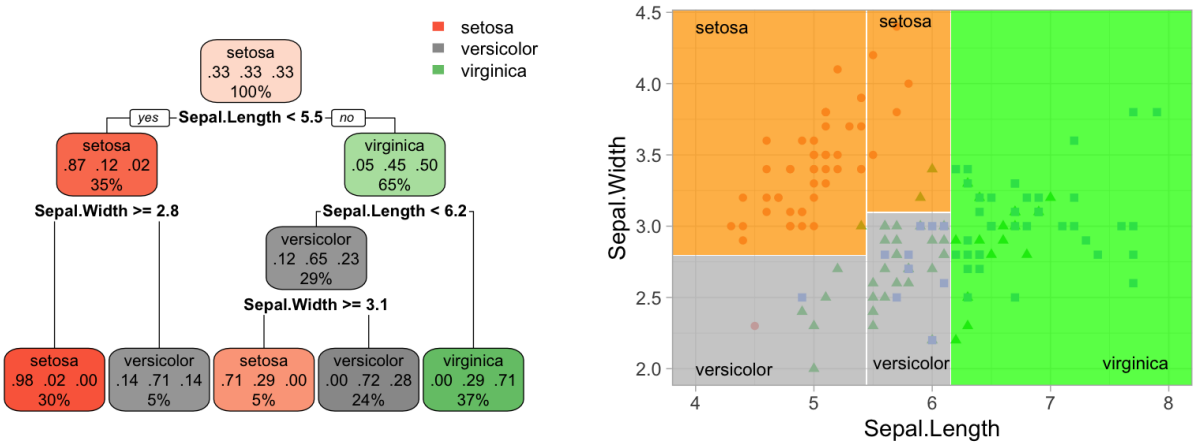


FIGURE 4.7 – Algorithme CART appliqué à la base de données des iris de Fisher. À gauche, l'arbre de décision construit sur les variables longueur et largeur de la sépale. À droite, la partition de l'ensemble $\mathbb{X} = \mathbb{R}^2$ en fonction des critères. Les frontières des classes sont nécessairement parallèles aux axes du repère.

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.
- 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. 4.8 et Fig. 4.9) illustrent des arbres CART pour différents jeux de données traditionnels.

4.3.2 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 :

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

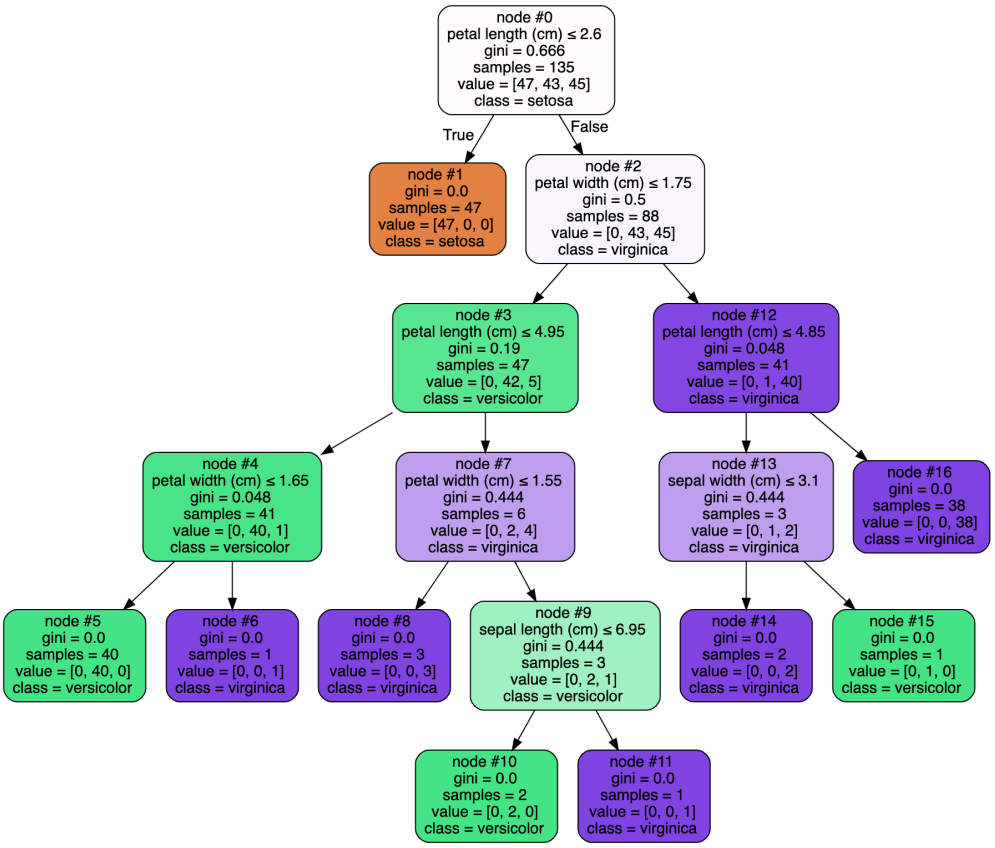


FIGURE 4.8 – Arbre de décision issu de la base des Iris de Fisher avec des critères de segmentation portant sur les 4 variables.

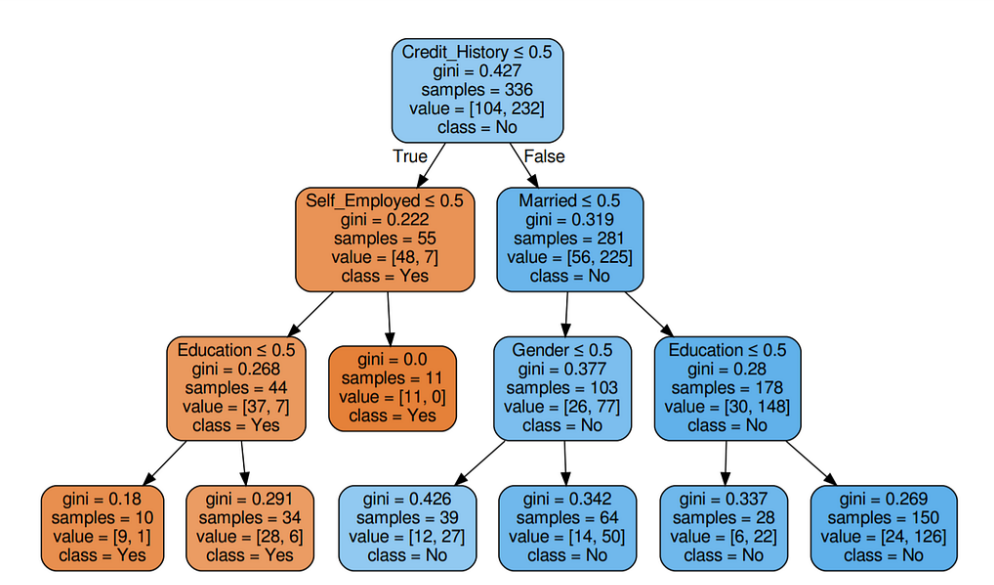


FIGURE 4.9 – Arbre de décision issu d'une base de données de défauts de paiement contenant 336 observations. Les variables utiliseées pour la segmentation sont : « Credit History », « Self Employed », « Married », « Education », « Gender ».

avec $q = \mathrm{N}_{\mathrm{A}_1}/\mathrm{A}_{\mathrm{A}_2}$ proportion des $x_i \in \mathrm{A}$ 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

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

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 :

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

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

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

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.

4.3.3 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 :

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

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

On définit ensuite un critère côut/complexité par

$$C_\alpha(T) = \sum_{m=1}^{M} N_m R_m(T) + m\alpha, \tag{35}$$

où $M = |T|$ 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 $C_\alpha(T)$ 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 $0 = \alpha_0 < ... < \alpha_M$ avec $M \leq |T|$ et une suite imbriquées de sous-arbres $(T_{\alpha_m})_m$ avec

$$T = T_{\alpha_0} \subseteq T_{\alpha_1} \subseteq ... \subseteq T_{\alpha_M} = \text{racine} \tag{36}$$

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

$$T_m \in \underset{T_i \subset T}{\arg\min}\, C_\alpha(T_i). \tag{37}$$

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.

4.3.4 Exemples en classification et régression.

Les figures Fig. 4.10 et Fig. 4.11 illustrent la méthode CART respectivement sur une tâche de classification (à nouveau les iris de Fisher) et une tâche de régression.

4.3.5 Remarques finales.

Les arbres CART forment une méthode de classification et de régression non linéaire (c.f. Fig. 4.12). 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.

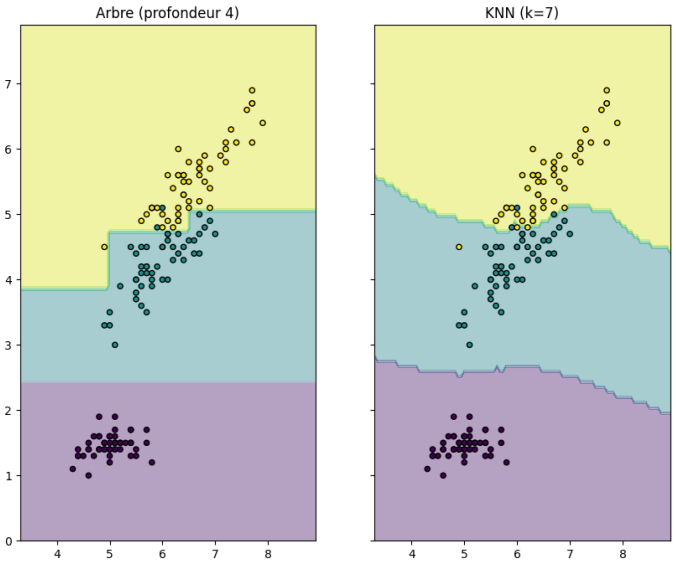


FIGURE 4.10 – Classification des trois espèces d'iris de Fisher par un arbre CART de profondeur 4 (à gauche) et par la méthode des k-ppv pour $k = 7$ (à droite). Noter le nombre et la position des iris mal classées dans les deux cas.

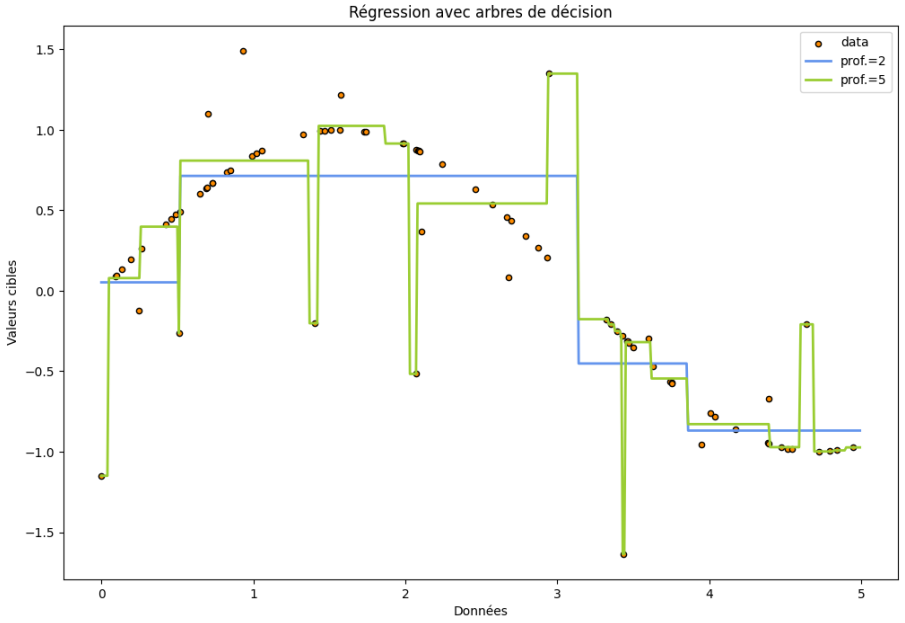


FIGURE 4.11 – Régression par arbre de décision. Un nuage de points (oranges) est généré à partir d'une sinusoïde perturbée par un bruit aléatoire. Il s'agit de reconstruire la courbe initiale à l'aide d'un arbre de décision de profondeur 2 (en bleu) ou de profondeur 5 (en vert)

- 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.

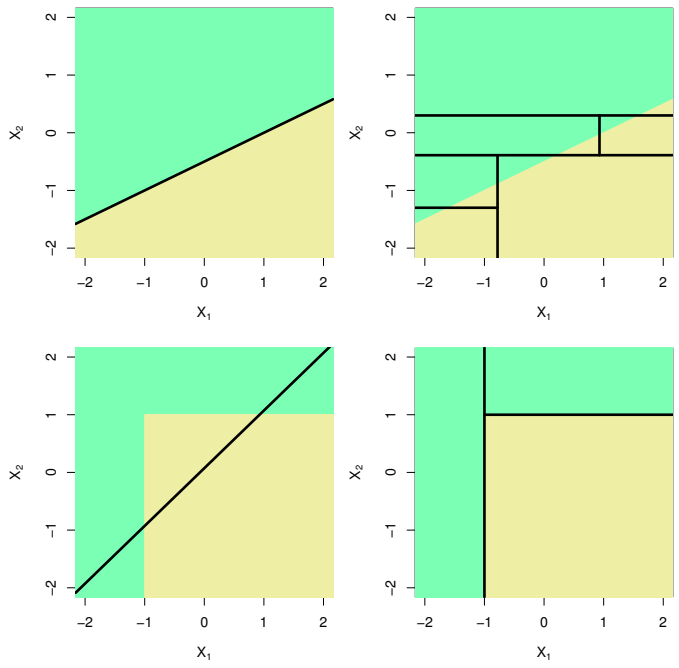


FIGURE 4.12 – Illustration de la non-linéarité des méthodes à base d'arbres. En haut : une tâche de classification binaire associée à deux régions linéairement séparables (une zone verte et une jaune) ; la méthode CART (colonne de droite) ne peut séparer que des régions à frontière horizontale et verticale. 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 5

Méthodes à base de convexification

5.1 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.

5.1.1 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.

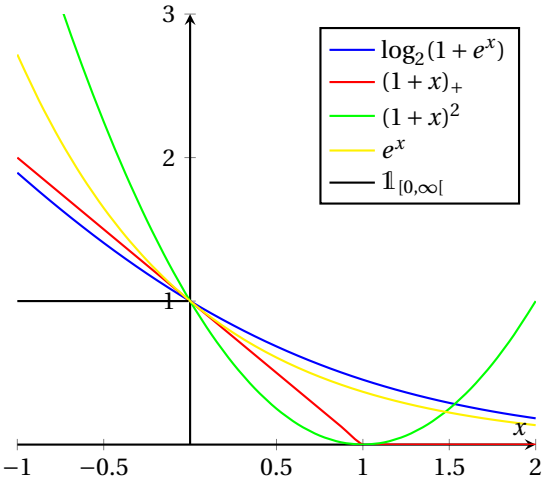


FIGURE 5.1 – Fonctions de perte convexe ϕ : courbes de $\phi(yh(x))$.

5.1.2 Convexification de la fonction perte

On définit le minimiseur du risque empirique convexifié par

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

$\widehat{h}_n$ 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 :

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

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

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

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	$\phi(x) = (1+x)^2$

5.1.3 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,

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

5.2 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.

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.

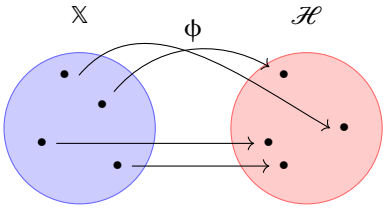


FIGURE 5.2 – $\mathscr{H}$ représente les données (features) issues de $\mathbb{X}$.

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.

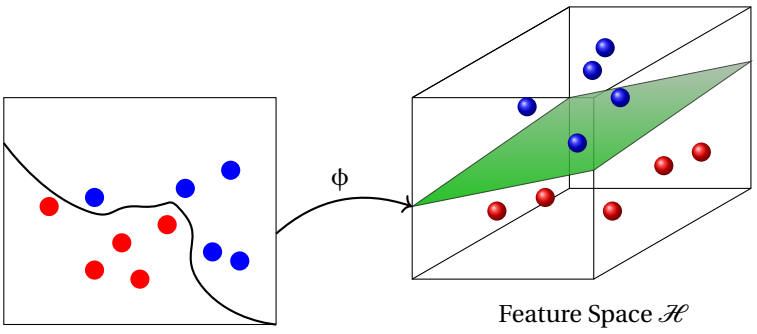


FIGURE 5.3 – On envoie les données initiales dans un espace (abstrait) de dimension plus grande, afin de pouvoir séparer plus facilement les classes.

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.

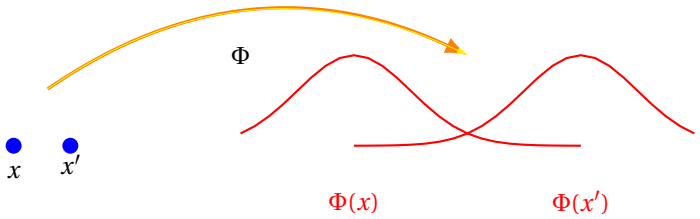


FIGURE 5.4 – La fonction de représentation ϕ transforme en fait un point de $\mathbb{X}$ en une fonction $K_x = \phi(x)$ de $\mathbb{X}$ dans $\mathbb{R}$ (considérée comme un élément de l'espace de Hilbert $\mathscr{H}$).

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

Exemples de noyaux usuels :

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

Théorème 14 — Théorème de représentation de Riesz

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

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

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

$$\mathrm{L}_x : \mathscr{H} \longrightarrow \mathbb{R} \tag{12}$$
$$h \longrightarrow \mathrm{L}_x(h) = h(x) \tag{13}$$

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

Définition 15

$\mathscr{H}$ 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

$$\forall x \in \mathbb{X}, \exists\, ! \, \mathrm{K}_x \in \mathscr{H} \text{ tel que } \mathrm{L}_x(h) = \langle \mathrm{K}_x, h \rangle = h(x), \ \forall h \in \mathscr{H}. \tag{14}$$

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

$$\mathrm{K}(x, y) = \langle \mathrm{K}_x, \mathrm{K}_y \rangle = \langle \phi(x), \phi(y) \rangle . \tag{15}$$

Autrement dit,

$$\mathrm{K}_x = \phi(x) \tag{16}$$

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 $\mathrm{K}_x(x')$ 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.

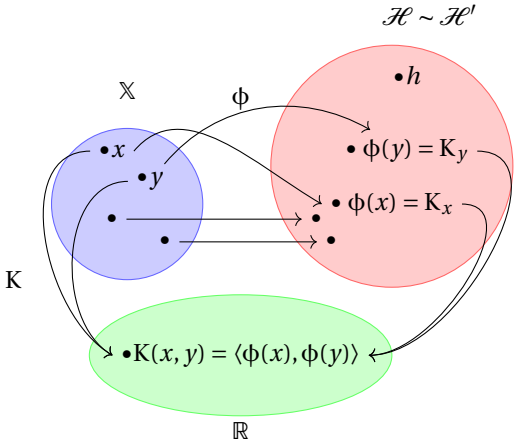


FIGURE 5.5 – Les espaces $\mathbb{X}$, $\mathscr{H}$, $\mathscr{H}'$ et $\mathbb{R}$, ainsi que les fonctions ϕ et K.

Théorème 17 du représentant

- Soit $K : \mathbb{X} \times \mathbb{X} \longrightarrow \mathbb{R}$ 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

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

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

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

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.

5.3 Machines à vecteurs de support (SVM)

5.3.1 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.

5.3.2 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 :

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

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. 5.6). 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.

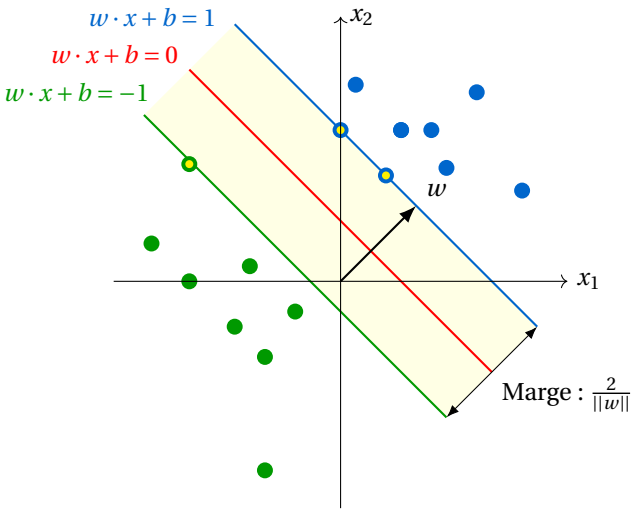


FIGURE 5.6 – Deux classes d'observations (en bleu et vert) linéairement séparables. En rouge, l'hyperplan séparateur de plus grande marge, en vert et bleu les hyperplans frontières de chaque classe avec les vecteurs supports (dont le centre est de couleur jaune). La largeur du tube coloriée en jaune 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 :

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

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

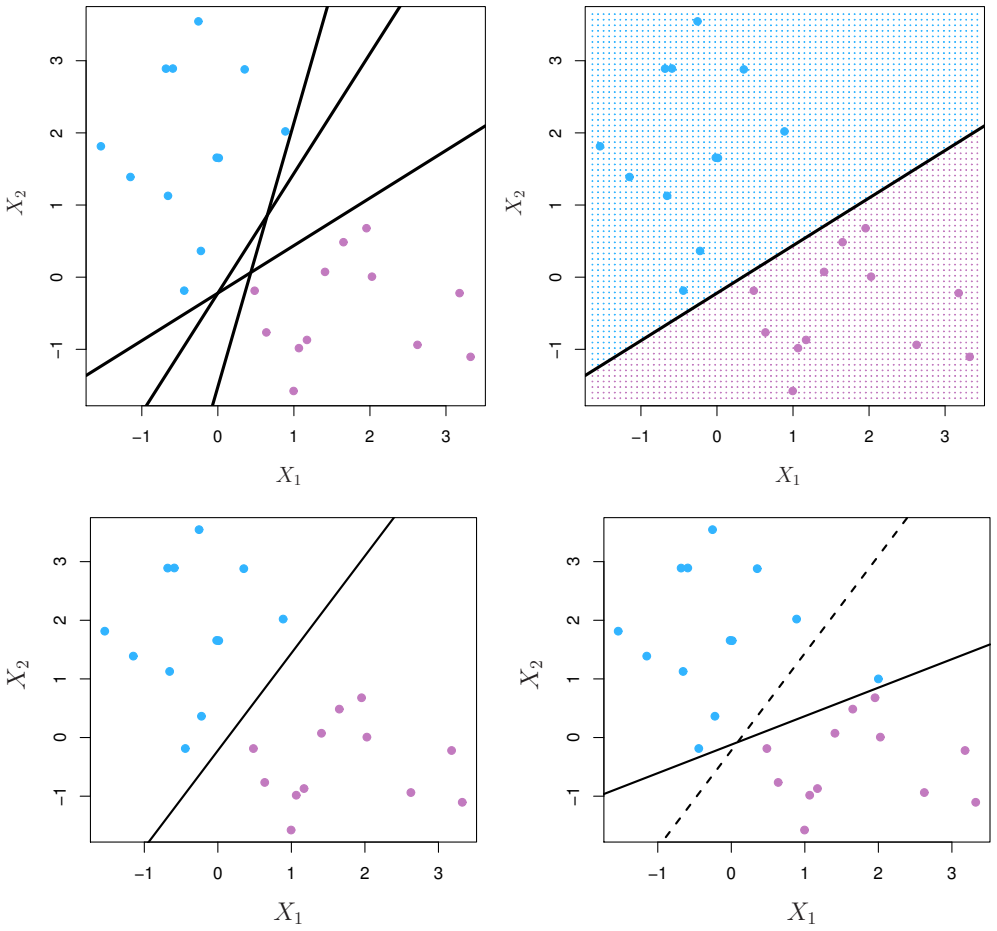


FIGURE 5.7 – Deux classes d'observations (bleu et rouge) mesurées par l'intermédiaire de deux variables x_1 et x_2. À gauche, trois exemples de plans séparateurs. À droite, le plan séparateur optimal, le plus éloigné des deux nuages. Crédit : Introduction to Machine Learning with Python. James, Witten, Hastie, Tibshirani. Figures du chapitre 9.

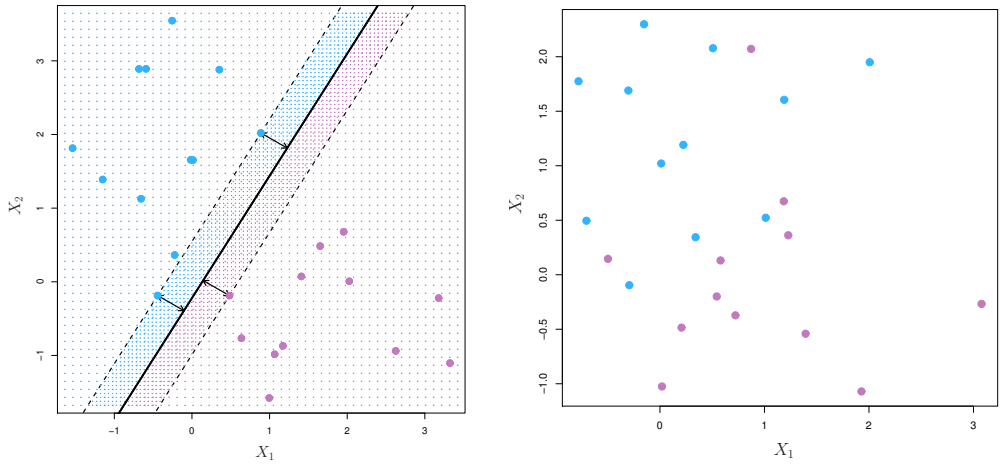


FIGURE 5.8 – Deux classes d'observations (bleu et rouge) mesurées par l'intermédiaire de deux variables x_1 et x_2. À gauche, le plan séparateur de plus grande marge avec les points les vecteurs de support à la frontière. À droite, un exemple de nuages non linéairement séparable à cause d'un point rouge ajouté par rapport au nuage précédent (trouver où...). Crédit : Introduction to Machine Learning with Python. James, Witten, Hastie, Tibshirani. Figures du chapitre 9.

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

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 $\mathrm{K}(x, y) = \langle \phi(x), \phi(y) \rangle$ et... on retrouve le premier point de vue relatif aux espaces de Hilbert.

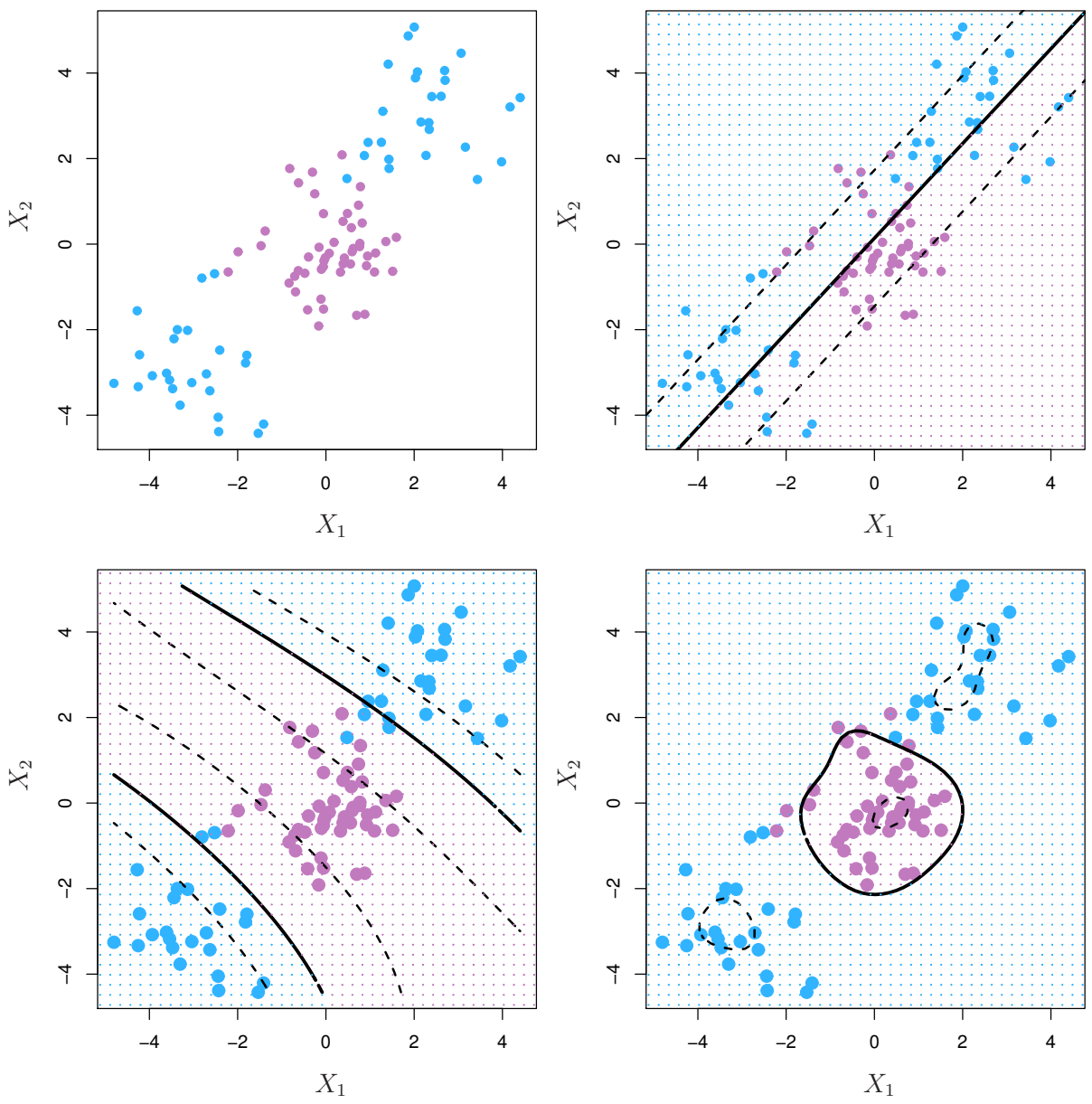


FIGURE 5.9 – Deux classes d'observations (bleu et rouge) mesurées par l'intermédiaire de deux variables x_1 et x_2. En haut, les données ne sont clairement plus linéairement séparables. En bas, après un changement de variables de x en $\phi(x)$ par une fonction non linéaire, il est à nouveau possible 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.

5.3.3 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.) :

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

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

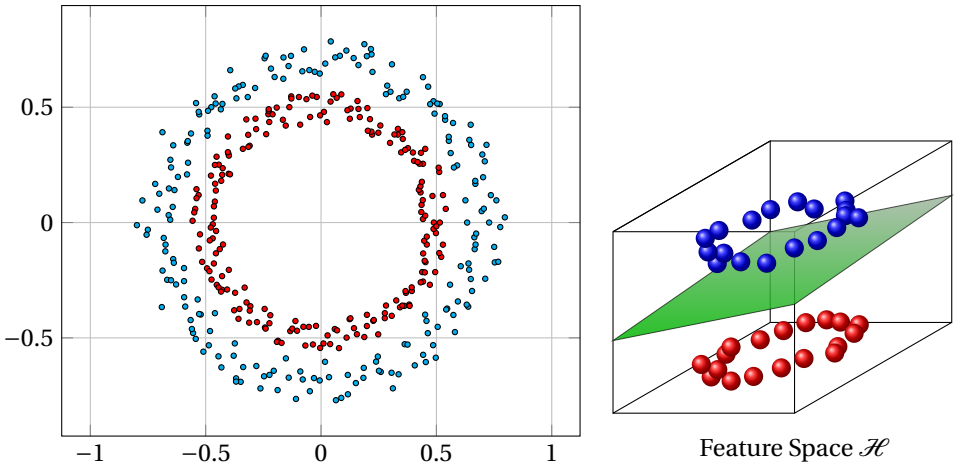


FIGURE 5.10 – Données non linéairement séparables en dimension 2, mais dont on se doute qu'elles sont à symétrie circulaire. En dimension 3, elles correspondent à deux nuages de hauteurs différentes qui sont linéairement séparables. $\phi(x,y) = (x,y,x^2+y^2)$ et $K(x,y) = \langle \phi(x), \phi(y) \rangle = x^T y + ||x||^2 ||y||^2$.

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. 5.11). 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.

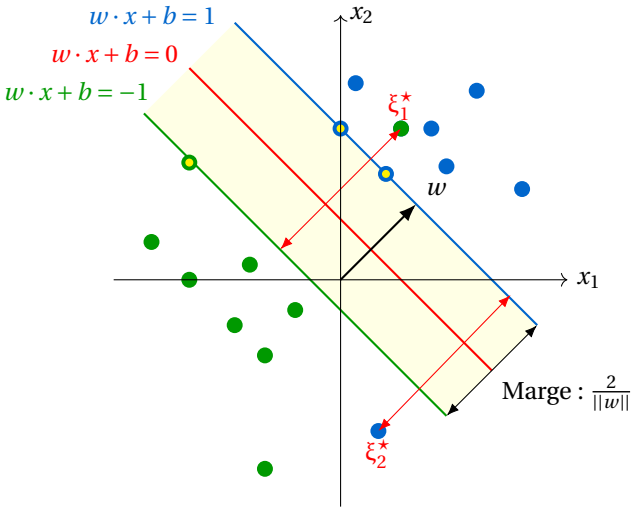


FIGURE 5.11 – Marge souple avec deux classes d'observations (en bleu et vert) non linéairement séparables. Deux points (l'un vert, l'autre bleu) sont de part et d'autre de la marge. La distance $\xi_1^\star$ et $\xi_2^\star$ à la frontière de leur région les caractérisent.

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.

5.3.4 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.

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

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

On a ainsi :

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

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 5.12). 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 :

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

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

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

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

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

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

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

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 :

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

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. 5.13 et Fig. 5.14 illustrent la régression à base de machines à vecteurs de support.

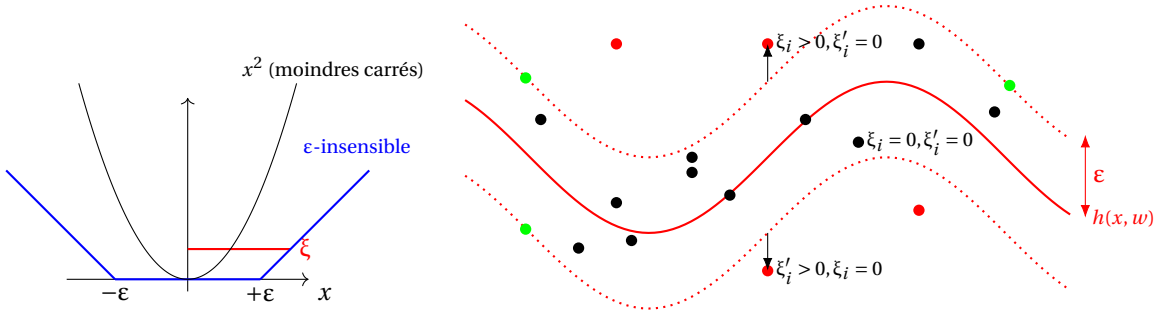


FIGURE 5.12 – À gauche : la fonction de perte ϵ-insensible comparée à la perte au sens des moindres carrées. À droite : dans la SVR, la fonction perte définie un ϵ-tube de part et d'autre de la fonction de régression h dans laquelle doivent se trouver tous les points de l'échantillon. Il est possible 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.

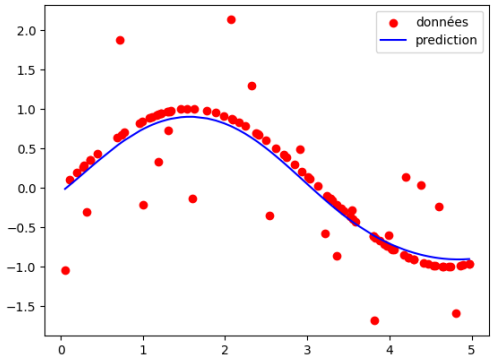


FIGURE 5.13 – Un nuage de points (en rouge) générés autour d'une sinusoïde perturbée par un bruit aléatoire et la courbe de régression SVR (en bleu).

5.3.5 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).

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

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

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 5.15 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.

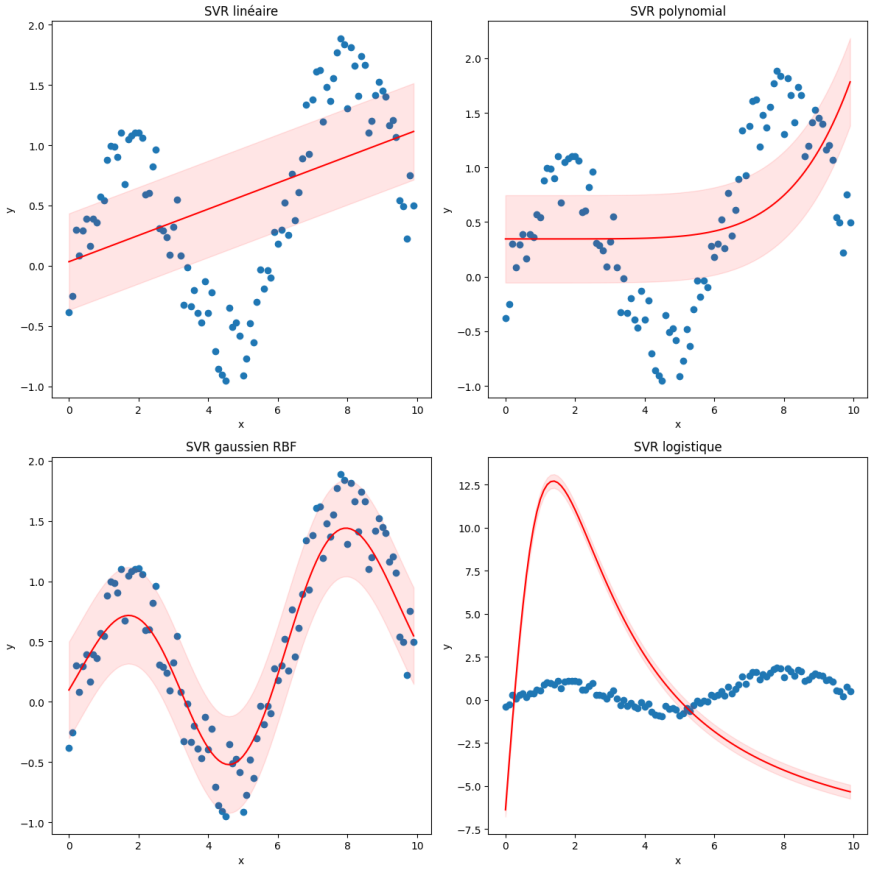


FIGURE 5.14 – Un nuage de points (en bleu) générés autour d'une sinusoïde perturbée par un bruit aléatoire et les courbes de régressions SVR (en rouge, avec les ϵ-tubes en rose, pour $\epsilon = 0.5$) pour 4 types de noyaux : linéaire, polynomial, gaussien et logistique. Noter les performances des noyaux gaussiens.

- 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).
- 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.

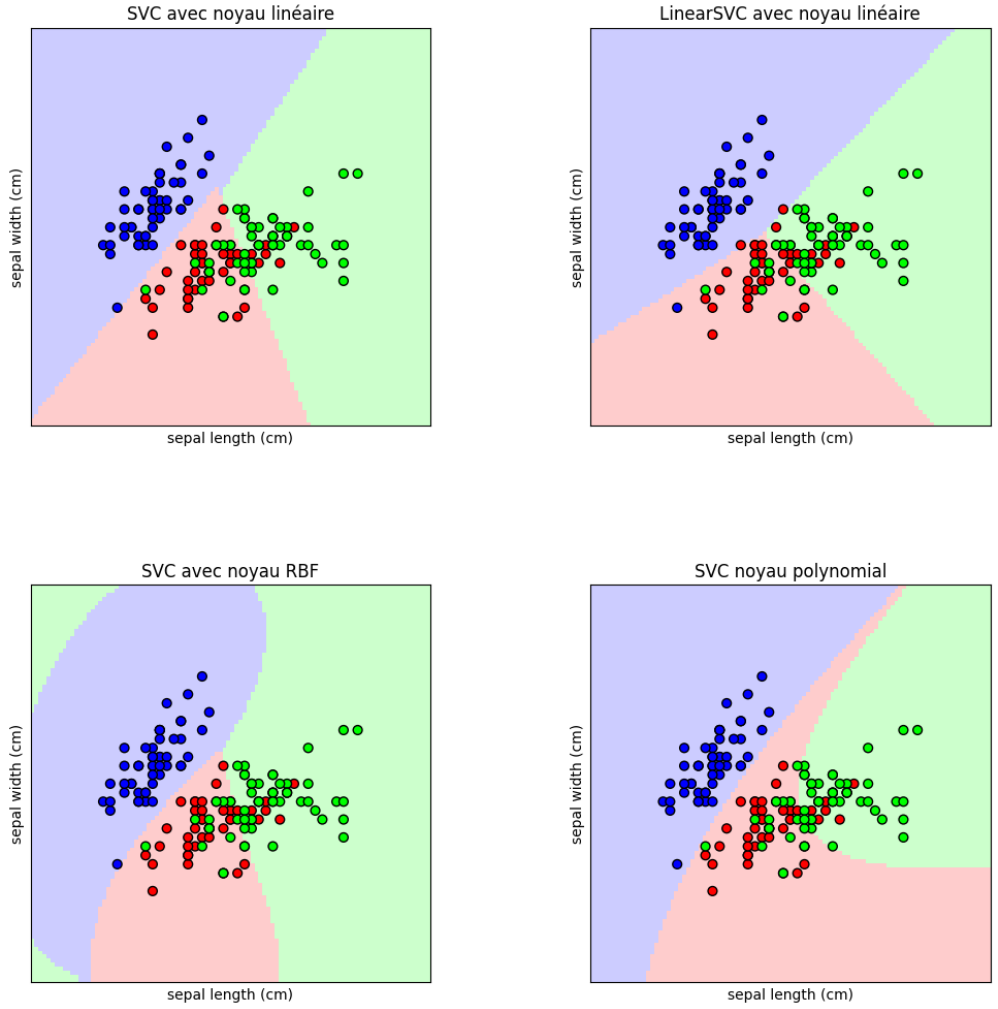


FIGURE 5.15 – Utilisation des SVM pour classer les 3 espèces d'iris de Fisher. 4 noyaux différents sont proposés. Les régions dédiées à chaque espèce sont représentées dans le plan caractérisé par la longueur et la largeur du sépale.

5.4 Boosting

5.4.1 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. Le Gradient Boosting est une interprétation du Boosting comme descente de gradient dans un espace fonctionnel.

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

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

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.

5.4.2 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 5.16 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 $w = (w_i(m))_i$:

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

2 • Calculer l'erreur normalisée :

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

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

4 • Mettre à jour les poids des observations :

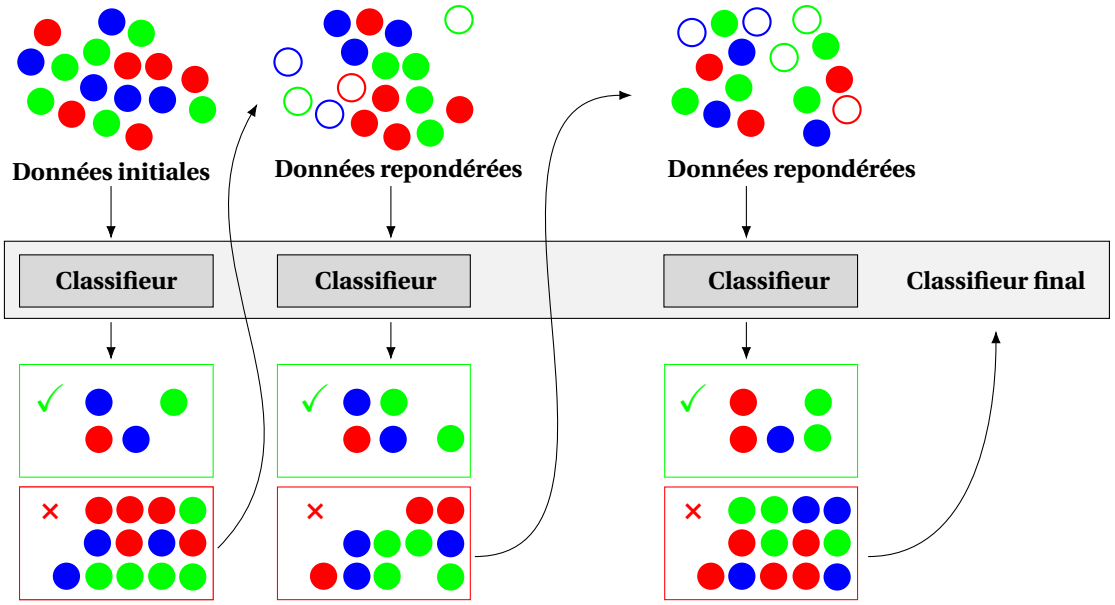


FIGURE 5.16 – Schéma traditionnel présentant la technique de Boosting : un premier classifieur faible est entraîné sur l'ensemble des données initiales. Il pondère de façon différentes les données bien (rectangle vert) et mal classées (rectangle rouge) en favorisant ces dernières afin que le classifieur 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.

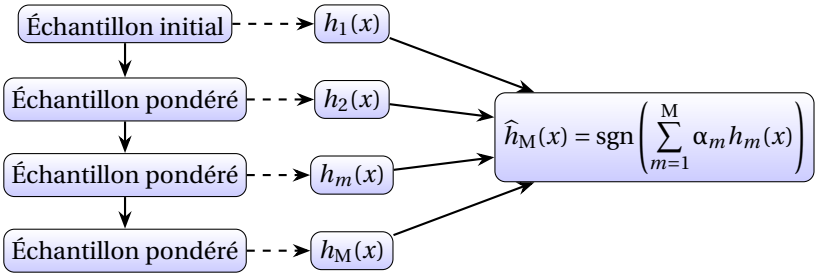


FIGURE 5.17 – Forme du classifieur Boosting.

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

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

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

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

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

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

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

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

Démonstration. 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,

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

$$= \underset{\alpha>0}{\arg\min} \left((e^{\alpha} - e^{-\alpha})\frac{1}{n}\sum_{i=1}^{n} w_i \mathbb{1}_{[Y_i \neq h(X_i)]} + \frac{1}{n}e^{-\alpha} \right).$$

$\forall \alpha \geq 0$, 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

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

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

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

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

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 : $\epsilon_m = 1/2 - \gamma_n$ avec $\gamma_n > 0$.

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

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

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 :

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

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 $\mathrm{M} = \mathrm{M}_n = n^{1-\epsilon}$, $\epsilon \in]0,1[$, 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

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

et de la sortie qui est donnée par

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

5.4.3 Gradient Boosting

Nous reprenons 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 $y_i = \pm 1$. Le risque empirique est

$$\mathrm{R}_n(h,x,y) = \frac{1}{n}\sum_{i=1}^{n} l(h(x_i),y_i) \tag{67}$$

L'algorithme de Gradient Boosting utilise une méthode semblable à la méthode de Newton-Raphson pour minimiser le risque empirique (lorsque celui-ci est strictement convexe). On fixe un paramètre λ et l'on construit une suite récurrente de la forme

$$x_m = x_{m-1} - \lambda l'(x_{m-1}). \tag{68}$$

Se pose le problème de la notation du vecteur gradient. Que signifie $\nabla_x \mathrm{R}_n$? Ce que l'on trouve partout est la formule suivante :

$$\nabla_x \mathrm{R}_n(h_m) = \left(\frac{\partial l(y_i,h(x_i))}{\partial h(x_i)}\Big|_{h(x_i)=h_m(x_i)}\right)^{\mathrm{T}}_{i=1,..,n} \tag{69}$$

Cette notation est ambiguë et peu claire à cause de la dérivation par rapport à $h(x_i)$, qui n'est pas une variable. En fait, R_n est une fonction de deux variables vectorielles x et y et on ne peut donc la dériver que par rapport aux composantes de ces deux variables. Voici la notation que nous adopterons : partant de $\frac{\partial l}{\partial x}(x,y)$ on pose

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

Il s'agit du gradient de la perte empirique R_n (qui est différentiable) en tant que fonction de sa variable x, évalué aux points d'abscisses $h_m(x_i)$. Le gradient est le vecteur des dérivées partielles en x_i que l'on évalue ensuite aux points de coordonnées $(h_m(x_i), y_i)$. Cette mise au point étant effectuée, nous présentons le principe de Gradient Boosting.

Comme on l'a dit, Adaboost est un cas particulier d'algorithme de descente de gradient, dans lequel on cherche $\sum_{m=1}^{M} \alpha_m h_m$ minimisant $R_n(h) = \sum_{i=1}^{n} l(h(x_i), y_i)/n$. Puisqu'il n'existe pas de solution explicite, on cherche donc une solution approchée et l'idée est d'utiliser un algorithme de descente de gradient de type Newton-Raphson. On parle parfois de gradient fonctionnel car on cherche une fonction minimisante et non un point ou un vecteur.

À une itération donnée, on a un classifieur h_{m-1} à améliorer. On lui ajoute g de telle sorte que $R_n(h_{m-1} + \alpha g)$ diminue au maximum. D'après la méthode de Newton-Raphson, il faut prendre l'opposé du gradient : $g = -\nabla_x R_n(h_{m-1})$. On pose donc :

$$h_m(x) = h_{m-1}(x) - \alpha \nabla_x R_n(h_{m-1}) = \left(h_{m-1}(x_i) - \frac{\lambda}{n} \frac{\partial l}{\partial x_i}(h_{m-1}(x_i), y_i) \right)_i$$

Il y a deux problèmes :

- La récurrence donne h_m uniquement aux points x_i.
- Le gradient g 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

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

On cherche la fonction $h \in \mathscr{H}$ la plus proche de g : 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 $m = 1, ..., M$, 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 $\mathscr{H}$ que l'on transforme en classifieur binaire :

$$\widehat{h}_M(x) = \operatorname{sgn}\left(\sum_{m=1}^{M} \alpha_m h_m(x) \right). \tag{71}$$

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

$$\widehat{\mathrm{M}} = \arg\min_{\mathrm{M}} \frac{1}{n} \sum_{i=1}^{n} l(\mathrm{Y}_i, \widehat{h}_{\mathrm{M}}(\mathrm{X}_i)), \tag{72}$$

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

$$h^\star(x) = \frac{1}{2} \ln\left(\frac{\mathbb{P}[\mathrm{Y}=1|\mathrm{X}=x]}{\mathbb{P}[\mathrm{Y}=-1|\mathrm{X}=x]}\right). \tag{73}$$

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

$$p(x) = \frac{e^{x^{\mathrm{T}}\beta}}{1+e^{x^{\mathrm{T}}\beta}} \tag{74}$$

où β est le vecteur des coefficients d'une régression linéaire. Si, à la place de cette définition, on pose

$$p(x) = \frac{e^{h(x)}}{1+e^{h(x)}} \tag{75}$$

pour une fonction vectorielle h, alors l'expression de la log-vraisemblance permet de définir la fonction de perte à minimiser :

$$l(x,y) = \ln\left(1+\exp(-2yh(x))\right). \tag{76}$$

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.

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

$$p_k(x) = \frac{e^{h_k(x)}}{\sum_{i=1}^{\mathrm{K}} e^{h_i(x)}}, \tag{77}$$

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

$$\mathrm{L}(y, h(x)) = -\sum_{k=1}^{\mathrm{K}} \mathbb{1}_{[y=k]} \ln p_k(x) \tag{78}$$

$$\frac{\partial \mathrm{L}}{\partial x} \mathrm{L}(y,x)|_{x=h_k} = \mathbb{1}_{[y=k]} - p_k(x) \tag{79}$$

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.

5.4.4 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.

5.4.5 Boosting pour la régression : L^2-Boosting

En régression, on utilise simplement l'algorithme de Gradient Boosting avec la fonction de perte donnée par

$$l(y, h(x)) = \frac{1}{2}(y - h(x))^2. \tag{80}$$

dans un cadre où $\mathbb{Y} = \mathbb{R}$.

À l'issue des M itérations, l'estimateur $\widehat{h}_M$ est un estimateur du minimiseur du risque empirique :

$$h^\star = \mathbb{E}[Y|X = x] = \arg\min_h \mathbb{E}\left[(Y - h(X))^2\right] \tag{81}$$

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. 5.18 et Fig. 5.19 illustrent les méthodes de Boosting pour la régression.

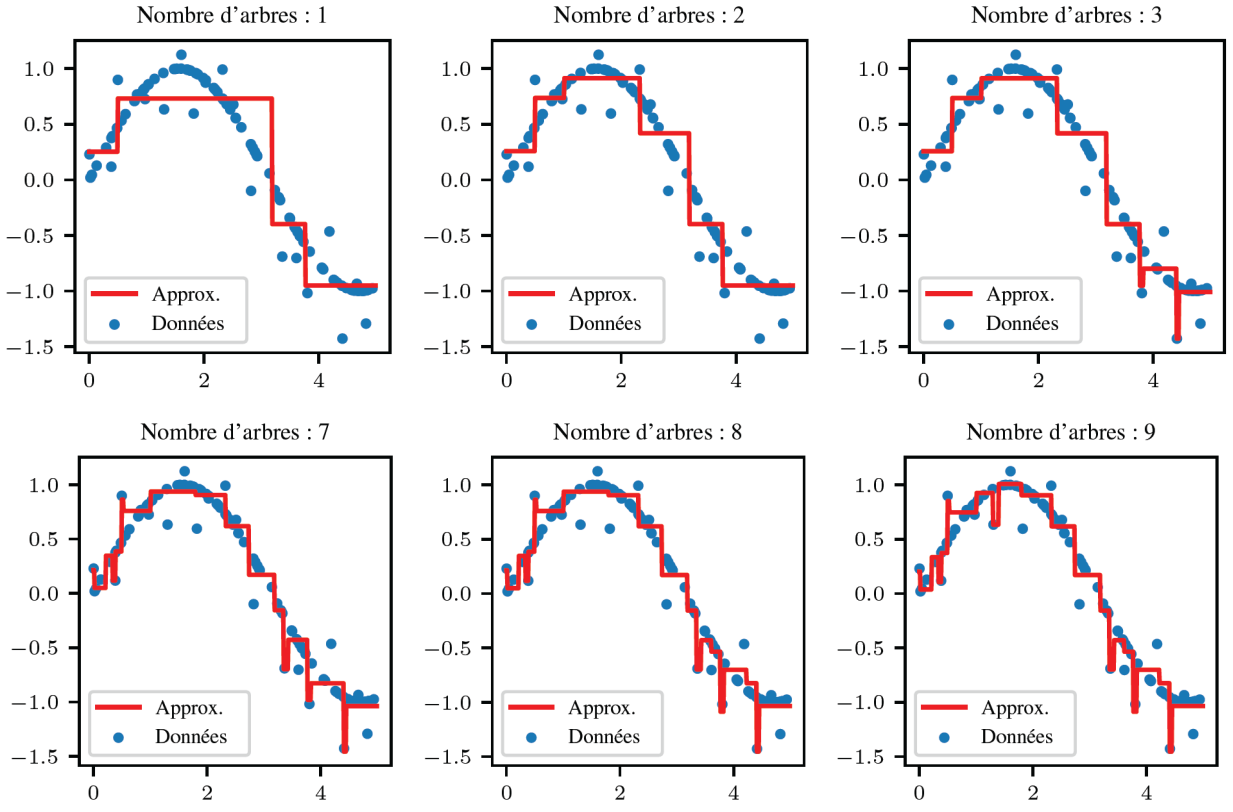


FIGURE 5.18 – Performances d'Ababoost dans une tâche de régression. Chaque classifieur faible est ici un arbre de profondeur 2.

5.4.6 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.

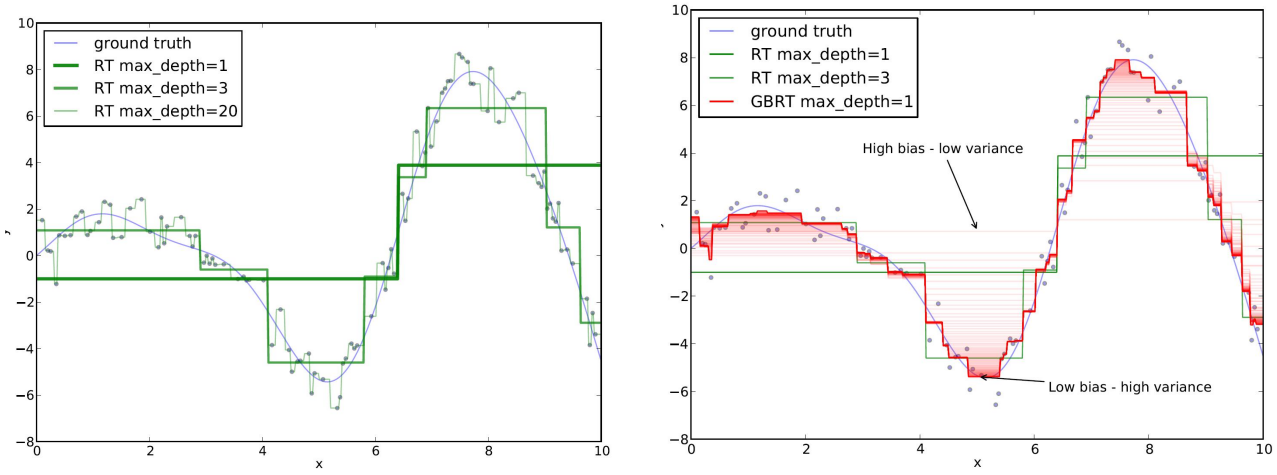


FIGURE 5.19 – Performances de Gradient Boosting dans une tâche de régression. À gauche, des arbres de décision, seuls, en régression (profondeur 1, 3, 20). À droite, une exécution de Gradient Boosting avec des arbres de profondeur 1.

— 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 6

Méthodes à base d'agrégation

6.1 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.

6.2 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 6.1 – 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.

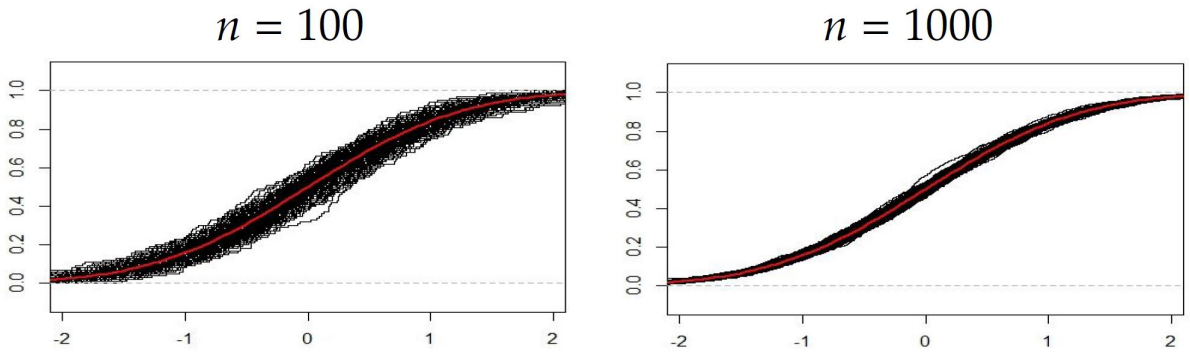


FIGURE 6.2 – Convergence uniforme des fonctions de distributions empiriques (en noir) d'échantillons de taille n vers la fonction de répartition théorique (en rouge) de la loi initiale à densité. Crédit : Arthur Charpentier.

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 $X \sim \mathcal{E}$ 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 $\mathscr{D}_n$.

$$\begin{cases} \theta_1^\star = \mathrm{T}(\mathrm{X}_{11}^\star, ..., \mathrm{X}_{1n}^\star) \\ ... \\ \theta_b^\star = \mathrm{T}(\mathrm{X}_{b1}^\star, ..., \mathrm{X}_{bn}^\star) \\ ... \\ \theta_\mathrm{B}^\star = \mathrm{T}(\mathrm{X}_{\mathrm{B}1}^\star, ..., \mathrm{X}_{\mathrm{B}n}^\star) \end{cases}$$

Les vecteurs $\theta_b^\star$ pour $b = 1, ..., \mathrm{B}$, sous $\mathbb{P}_n$, sont des réplications bootstrap de $\widehat{\theta} = \mathrm{T}(\mathrm{X}_1, ..., \mathrm{X}_n)$.

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...

6.3 Bagging : Bootstrap Agregating

6.3.1 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 $b = 1, ..., \mathrm{B}$, 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 $g_1, ..., g_\mathrm{B}$. Pour une tâche de régression posons comme modèle : $\mathrm{Y} = g(\mathrm{X}) + \epsilon$ avec $g(x) = \mathbb{E}[\mathrm{Y}|\mathrm{X} = x]$.

Pour chaque échantillon bootstrap $\mathscr{D}_{nb}$ issu de l'échantillon initial, on ajuste un régresseur $g_b(x)$. Le régresseur bagging est

$$\widehat{g}_\mathrm{B}(x) = \frac{1}{\mathrm{B}} \sum_{b=1}^{\mathrm{B}} g_b(x) \tag{3}$$

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 $b = 1, ..., \mathrm{B}$:

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

$$\widehat{g}_\mathrm{B}(x) = \frac{1}{\mathrm{B}} \sum_{b=1}^{\mathrm{B}} g_b(x) \tag{4}$$

$g_b(x)$ 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.

6.3.2 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.

6.3.3 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 $(\mathrm{X}_i, \mathrm{Y}_i)$ 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

$$\widehat{\mathrm{Y}}_i = \frac{1}{|\mathbb{J}_i|}\sum_{b\in\mathbb{J}_i} \mathrm{T}(\mathrm{X}_i, \mathscr{D}_{nb}).$$

L'erreur de prédiction OOB est : $\frac{1}{n}\sum_{i=1}^n (\widehat{\mathrm{Y}}_i - \mathrm{Y}_i)^2$.

La probabilité d'erreur OOB est : $\frac{1}{n}\sum_{i=1}^n \mathbb{1}_{[\widehat{\mathrm{Y}}_i \neq \mathrm{Y}_i]}$.

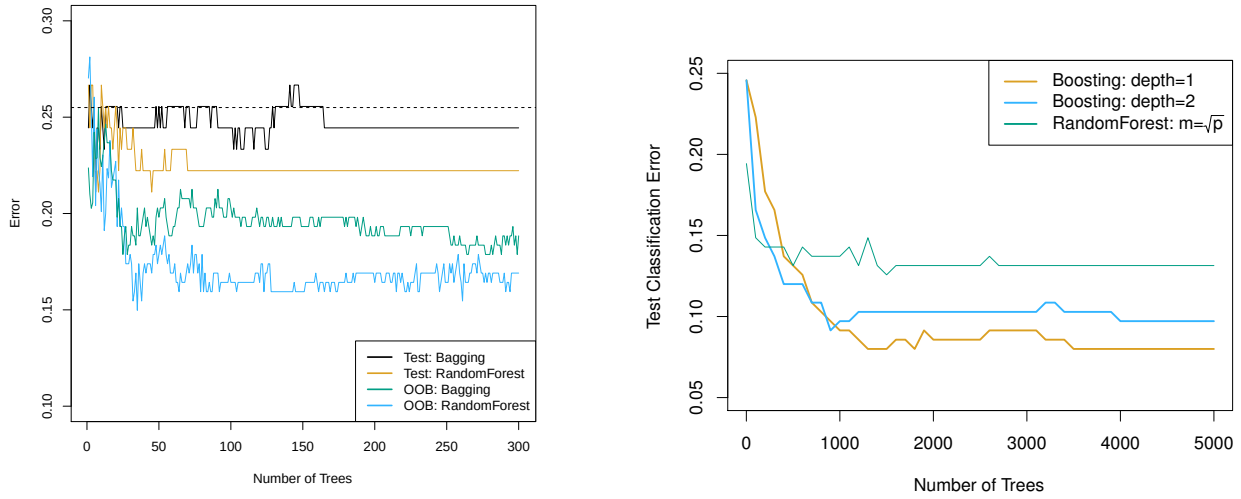


FIGURE 6.3 – À gauche : illustration des performances du Bagging et de forêts aléatoires. À droite : comparaison des performances en Boosting et Bagging en fonction du nombre d'arbres. Crédit : Introduction to Machine Learning with Python. James, Witten, Hastie, Tibshirani. Figures du chapitre 8.

6.4 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).

6.5 Stacking

6.5.1 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.

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.

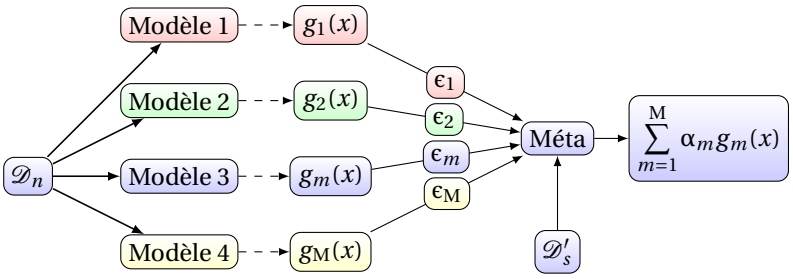


FIGURE 6.4 – Illustration du Stacking. Chaque classifieur niv.1 g_m fournit au méta-modèle son erreur de régression / classification ϵ_m. Le méta-modèle initialise les poids α_m à partir de ϵ_m. Il met à jour les poids lors de son entraînement sur $\mathscr{D}'_s$.

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.

6.6 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.

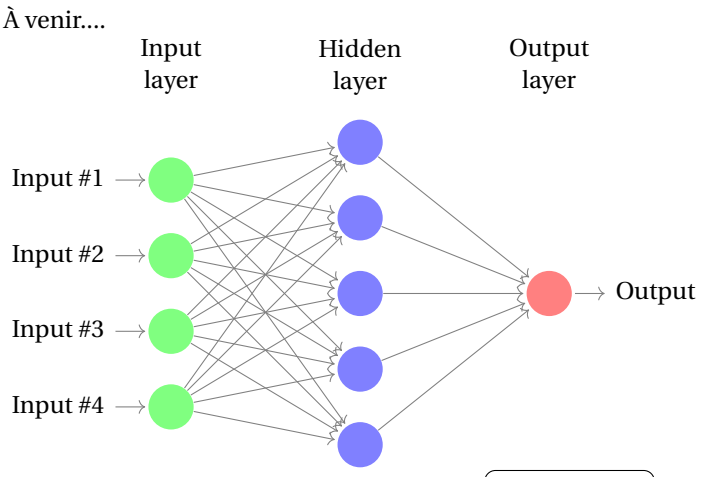
— + Chapitre 5 (bootstrap p.212) Intro to Statistical Learning with Python.

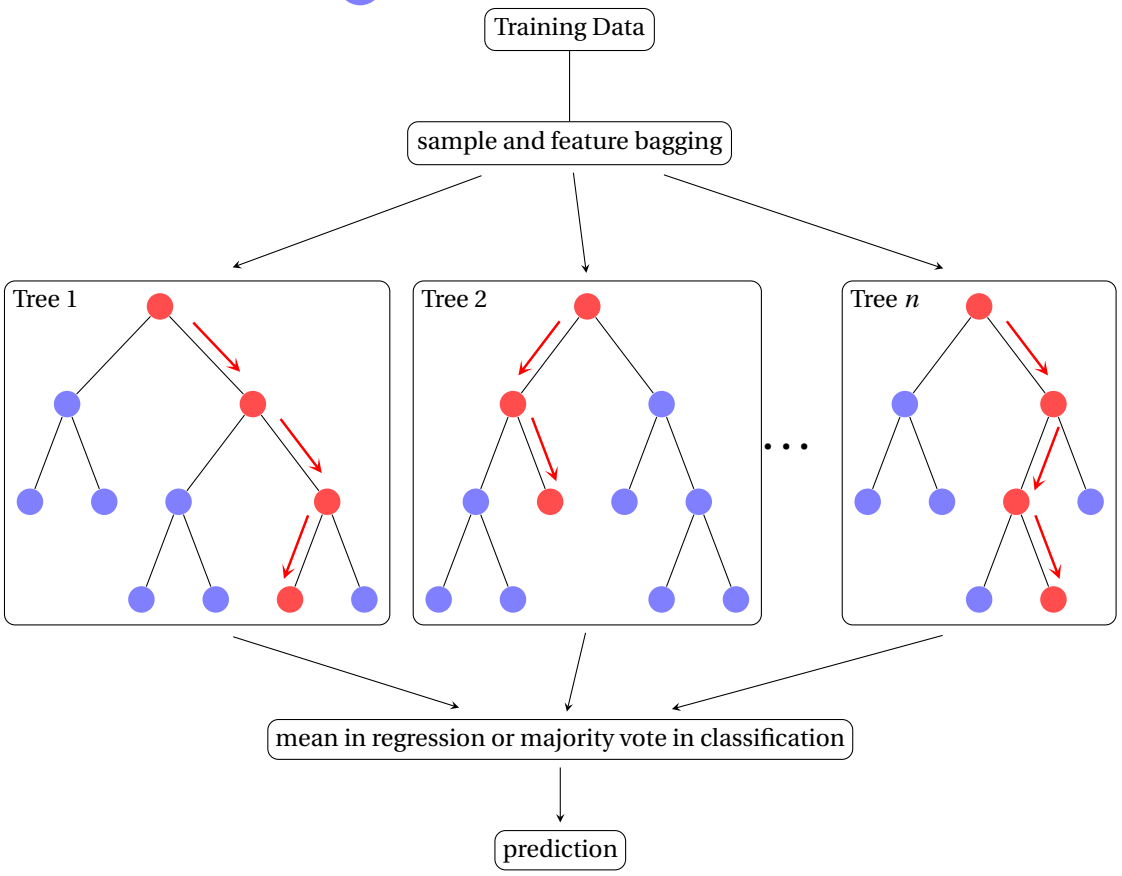
— + Chapitre 8 (Bagging et random forests p.343) Intro to Statistical Learning with Python.
— + Chapitre 7 (Bagging p.88) du polycopié de Frédéric Sur.

Chapitre 7

Les réseaux de neurones

À venir....





Chapitre 8

Brève introduction aux méthodes de l'apprentissage non supervisé

À venir....

8.1 Formalisation et principales méthodes

L'apprentissage non supervisé se distingue de l'apprentissage supervisé par son objectif fondamental : extraire des structures, des motifs ou des regroupements cachés dans des données sans utiliser d'étiquettes prédéfinies ; regrouper ce qui se ressemble (« ce qui se ressemble s'assemble »), éloigner ce qui est vraiment différent. Il intervient dans des contextes où aucune information de sortie n'est disponible. Il s'agit donc de laisser les données « parler d'elles-mêmes », de découvrir leur organisation intrinsèque, leur géométrie, leurs regroupements ou leur distribution, souvent en grande dimension.

Il est nécessaire de définir une notion de cluster (dans une partition), c'est à dire un groupe de « données similaires » et de définir une bonne notion de similarité.

L'apprentissage non supervisé est utile dans des situations où les données sont brutes, n'ont pas été annotées ou catégorisées. On cherche alors à explorer, identifier des tendances, des segments ou des structures latentes sans avoir d'hypothèse initiale forte.

Les données peuvent ne pas avoir été étiquetées pour différents motifs :

- Pas de temps ou d'argent.
- Pas de spécialiste pour étiquetter.
- Trop de catégories.
- Impossible à étiquetter.

Le but est alors de réduire la dimension des données pour faciliter leur visualisation ou leur traitement.

L'apprentissage non supervisé recouvre plusieurs approches classiques :

- Clustering (regroupement) : consiste à répartir les observations en groupes (les clusters) homogènes selon une notion de similarité.
- k-means, clustering hiérarchique, DBSCAN, spectral clustering, etc.
- Réduction de dimension : pour représenter les données dans un espace de dimension réduite tout en préservant leur structure (distance, variance, etc.).
- Analyse en composantes principales (ACP, PCA), algorithme de visualisation t-SNE, autoencodeurs.
- Isolation Forest.
- Analyse de densité et de distribution : estimer la structure probabiliste des données, souvent utilisée dans la génération de données ou la modélisation de phénomènes latents.
- Modèles de mélange gaussien (GMM), algorithme EM, modèles génératifs.
- Apprentissage de représentations (unsupervised representation learning) : apprentissage de caractéristiques ou d'encodages utiles pour d'autres tâches, par exemple via des réseaux de neurones non supervisés ou autoencodeurs.

L'absence de vérité terrain (« ground truth ») complique considérablement l'évaluation des résultats. Évaluation : Il n'y a souvent pas de critère objectif unique. On doit recourir à des mesures internes (inertie, silhouette), à la visualisation, ou à des comparaisons *a posteriori* avec des labels si ceux-ci sont disponibles. Paramétrage : De nombreuses méthodes nécessitent des paramètres à choisir sans supervision (nombre de clusters, voisinage, seuils...). Sensibilité aux données : Le prétraitement (normalisation, codage des variables, gestion des valeurs manquantes) influence fortement les résultats. Interprétabilité : Les structures extraites doivent souvent être interprétées par un expert humain, en tenant compte du contexte métier.

Domaines d'application possibles :

- Segmentation de clientèle en marketing,
- Regroupement de documents ou de contenus multimédias,
- Analyse exploratoire en biologie ou en sociologie,
- Compression de données,
- Détection de fraude ou d'attaques dans les systèmes,
- Prétraitement pour l'apprentissage supervisé (via apprentissage auto-supervisé, clustering préalable, etc.).

Voici un premier exemple intéressant, proposé par Nicolas Baskiotis, qui illustre bien la difficulté des méthodes non supervisées. Dans la figure Fig. 8.1, quel est le bon partitionnement ?

Réponse : aucun ! L'échantillon est aléatoire, de loi uniforme.

8.2 La notion de similarité

L'apprentissage non supervisé est un problème de similarité. Il existe différentes approches :

- Géométrique : basée sur la connectivité, centroïde (k-moyennes, CAH).
- Graphes (spectral clustering).
- Distribution de probabilités latentes (estimation de densités).
- Modèles bayésiens.
- Apprentissage génératif (réseaux de neurones).

Les méthodes à base de partitionnement :

- k-moyennes, DBSCAN, Mean Shift.
- Hard : une donnée appartient à un unique groupe.
- Soft : probabilité d'appartenance à un groupe.
- Nombre de classes k inconnu *a priori*.
- Similarité intra-groupe et dissimilarité inter-groupe.
- La malédiction de la dimension n'est jamais loin.

Formalisation mathématique

- Échantillon $\mathscr{D} = \{X_1, ..., X_n\}$ avec $X_i \in \mathbb{R}^d$.
- Partition π_k sur $\mathscr{D}$: $\mathscr{D}_1$,..., $\mathscr{D}_k$.
- Critère de similarité d (distance) sur $\mathbb{R}^d$ ou $\mathbb{X}$.
- Critère de similarité D sur les sous-ensembles de $\mathscr{D}$.
- Clustering : à k fixé, trouver $\pi_k^\star = \arg\min_\pi \phi(\pi)$
- ϕ est une fonction des distances d et D.

Distances sur $\mathbb{R}^d$ et sur $\mathscr{P}(\mathscr{D})$

$$d_p(x,y) = \left(\sum_{i=1}^{d} |x_i - y_i|^p \right)^{1/p} \tag{1}$$

— $p = 2$ distance euclidienne.

— $p = 1$ distance de Manhattan.

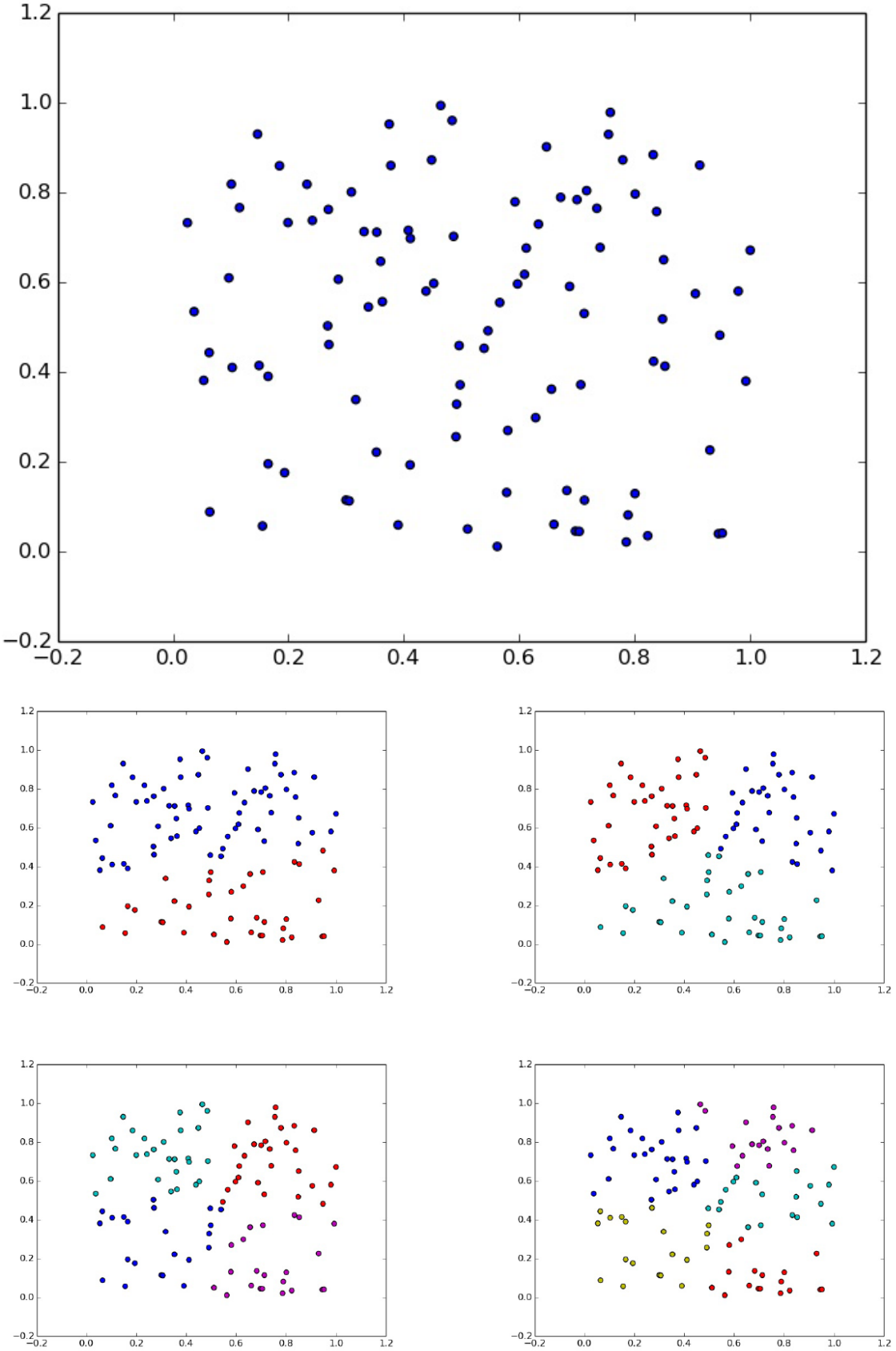


FIGURE 8.1 – En haut : un nuage de points à classifier. En bas : 4 propositions différentes de clustering en 2,3 et 4 classes. Crédit : Nicolas Baskiotis.

— $p \to 0$ distance de Hamming.
— p quelconque distance de Minkowski.

$$D(A,B) =$$

— PPV (simple linkage) : $\min\{d(x,y), x \in A, y \in B\}$
— Diamètre max (complete linkage) : $\max\{d(x,y), x \in A, y \in B\}$
— Moyenne (average linkage) : $\frac{1}{|A|.|B|}\sum_{x,y} d(x,y)$
— Ward : $\frac{|A|.|B|}{|A|+|B|}||m_A - m_B||^2$
— Barycentres : $d(A,B) = d(m_A, m_B)$

$A, B \in \{\mathscr{D}_1, ..., \mathscr{D}_k\}$ et $m_A = \sum_{x \in A} 1/|A|$ barycentre (centroïde) de A.

8.3 Classification ascendante hiérarchique

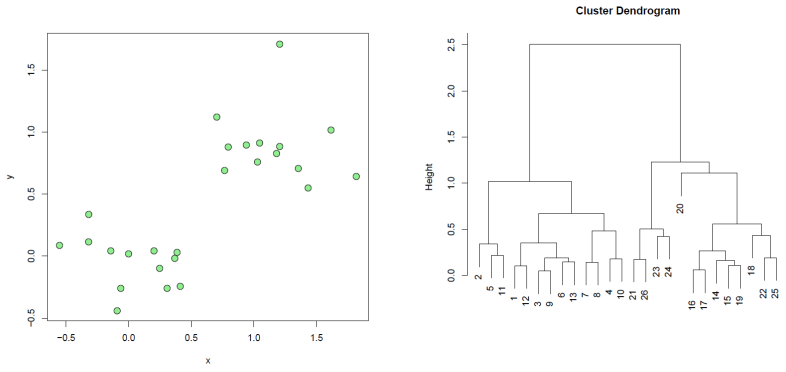
Classification ascendante hiérarchique : principe
Algorithme glouton :

— Fusionner les partitions les plus semblables selon D.
— Construire des clusters de plus en plus larges.
— S'arrêter quand il reste un unique cluster.
— ⇒ Arbre de partitionnement binaire : dendrogramme.

CAH : ce n'est pas une méthode de classification, mais de paritition nement (non supervisé) !

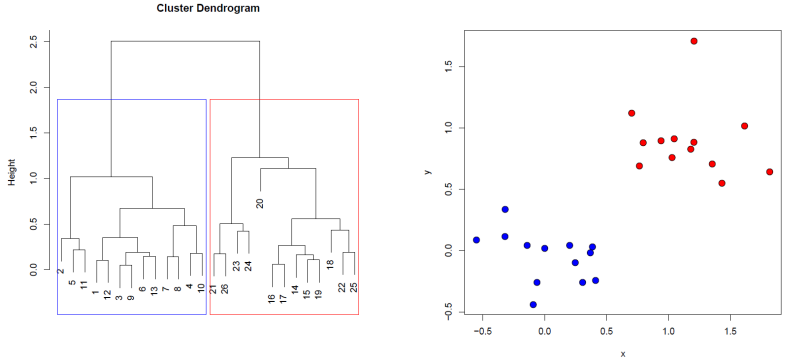
Selon le choix de D, le dendrogramme est plus ou moins équilibré.

Le choix de k est également important... et subjectif.
CAH : exemple 1 (J. Salmon, N. Verzelen) -1-



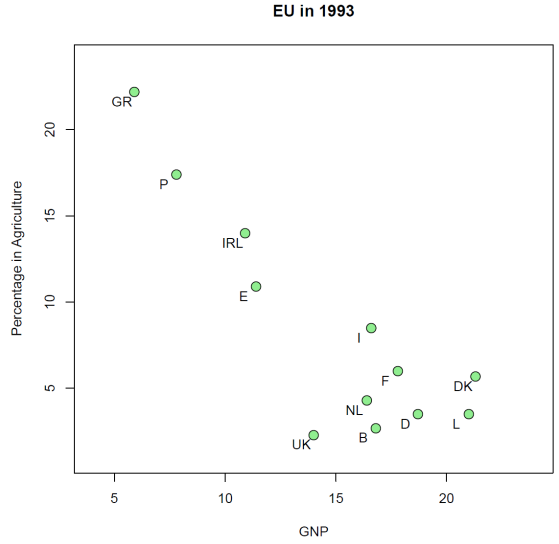
(Figure : J. Salmon, N. Verzelen)

Height sur l'axe (Oy) : distance entre les clusters.
CAH : exemple 1 -2-



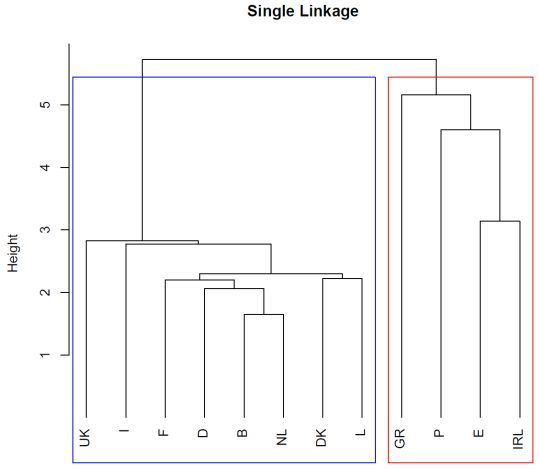
(Figure : J. Salmon, N. Verzelen)

CAH : exemple 2 (J. Salmon, N. Verzelen) -1-



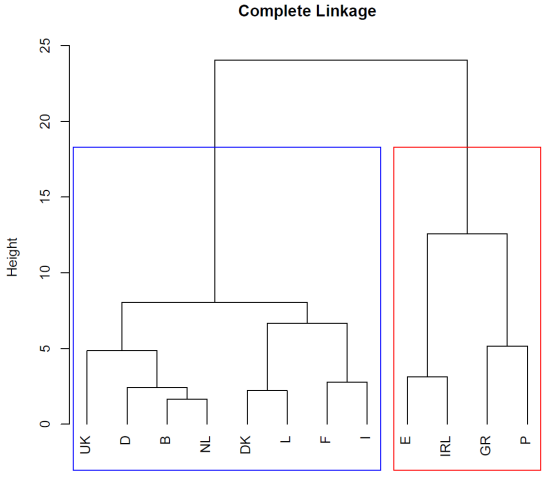
(Figure : J. Salmon, N. Verzelen)

CAH : exemple 2 -2-



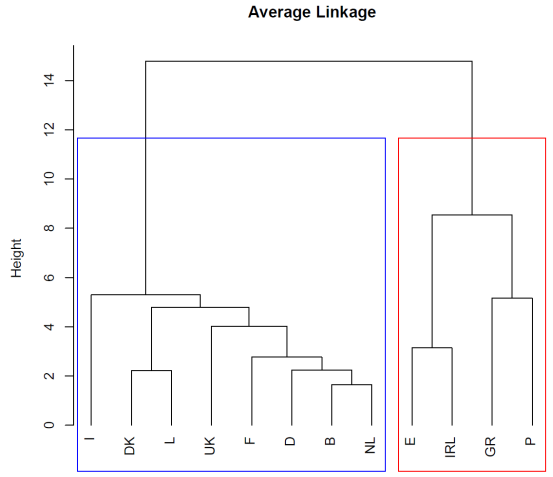
(Figure : J. Salmon, N. Verzelen)

CAH : exemple 2 -3-



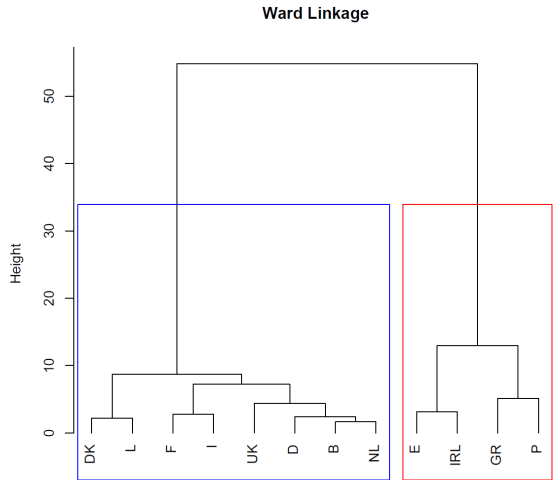
(Figure : J. Salmon, N. Verzelen)

CAH : exemple 2 -4-



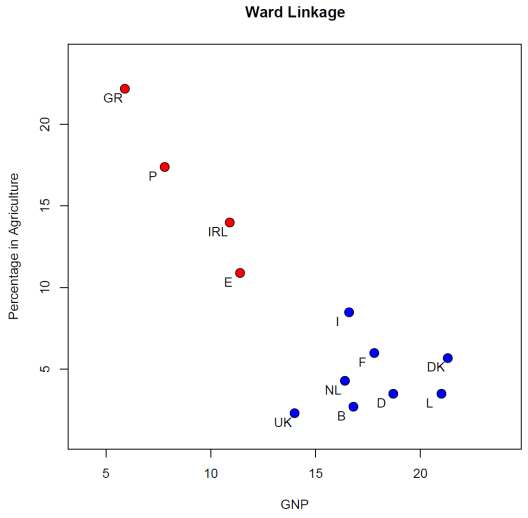
(Figure : J. Salmon, N. Verzelen)

CAH : exemple 2 -5-

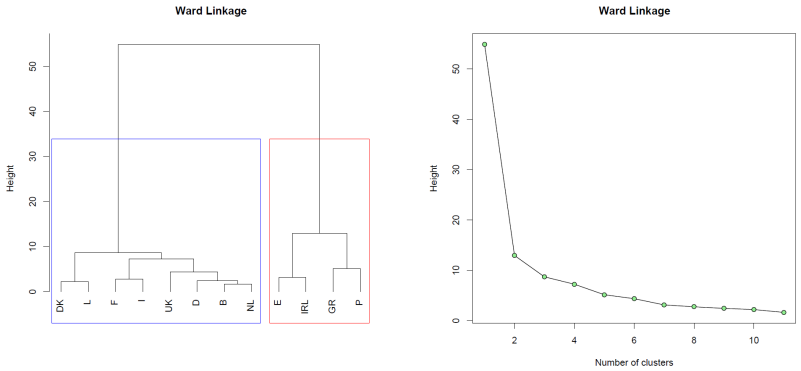


(Figure : J. Salmon, N. Verzelen)

CAH : exemple 2 -5-



(Figure : J. Salmon, N. Verzelen)



(Figure : J. Salmon, N. Verzelen)

Choix de k : méthode du « coude ». CAH : complexité $\mathcal{O}(n^3)$ en implémentation naïve. (n itérations sur matrice $n \times n$.).

Meilleurs algorithmes en $\mathcal{O}(n^2 \ln n)$ voire $\mathcal{O}(n^2)$

8.4 Méthode des k-moyennes

Algorithme des k-moyennes
Construit la partition qui minimise la distance intra-cluster (inertie) :

$$\varepsilon(\pi_k) = \sum_{i=1}^{k} \sum_{x_j \in \mathcal{D}_i} ||x_j - m_i||^2, \tag{2}$$

avec m_i barycentre (ou centroïde) du cluster (ou groupe) i :

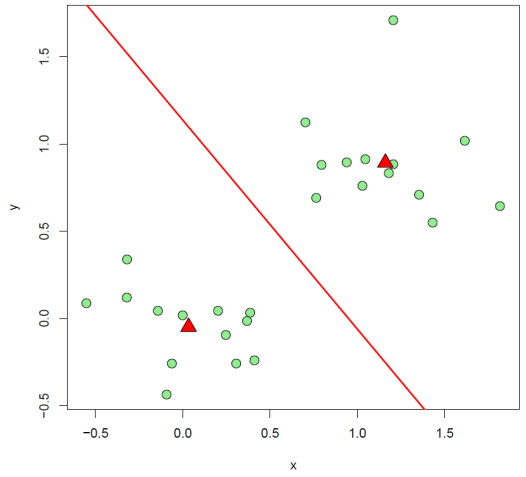
$$m_i = \frac{1}{|\mathcal{D}_i|} \sum_{x_j \in \mathcal{D}_i} x_j \tag{3}$$

L'algorithme construit :

$$\widehat{\varepsilon}_k \in \underset{\pi_k = \{\mathcal{D}_1, \ldots, \mathcal{D}_k\}}{\arg\min} \ \varepsilon(\pi_k).$$

Problème NP-difficile ⇒ obligation d'une méthode de résolution approchée.

k-means ≠ k-nn!!!!!
CAH -4-



(Figure : J. Salmon, N. Verzelen)
Résolution approchée
Formation cluster : chaque donnée affectée au centroïde le + proche.
Algorithme de Lloyd (1957)

— Affecter chaque point au cluster de plus proche centre m_i.

— Ré-estimer les centres selon la nouvelle répartition.

— Itérer jusqu'à convergence

Complexité : $\mathcal{O}(n(k+1))$.

Converge vers un minimum local seulement.

⇒ En pratique, on lance plusieurs fois l'algo avec ≠ initialisations.
Géométrie des classes
Les centres induisent une partition de Voronoi de $\mathbb{R}^d$.

$$V_i = \{x \in \mathbb{R}^d : ||x - m_i|| \leq \min_{k \neq i} ||x - m_k||\}$$

V_i est une cellule de Voronoi (convexe).
Convergence
Si la loi $\mathbb{P}$ est connue, on peut définir k centroïdes optimaux minimisant l'inertie.

Soit $\varepsilon_k^\star$ qui minimise l'inertie. Alors $\varepsilon_{k+1}^\star \leq \varepsilon_k^\star$.

Mais l'algorithme de Lloyd ne fournit pas forcément ce min.

$$\varepsilon(m_1, ..., m_k) = \mathbb{E}\left[\min_{i=1..k} ||X - m_k||^2\right]$$

Théorème Si ε a un min unique en $(m_1^\star, ..., m_k^\star)$ alors la suite des estimateurs minimisant ε_n converge p.s. vers...
Heursitique pour choisir k : méthode du coude (Elbow). Quand la décroissance devient moins franche.

Pour éviter les minima locaux, on recommence plusieurs dois l'algo.

Modèles génératifs
Modèle probabiliste : stochastic parrot (Shannon, Mathematical Theory of Communication, 1948).

Approche deep-learning : 2015
Chat-GPT (openAI) : 2022
Coût énergétique des LLM.
Chat-GPT : « écris-moi un cours d'apprentissage statistique pour les mastères spécialisés Data-Science »
Meow generator

8.5 Spectral clustering

Partitionnement spectral

— Les méthodes géométriques (dont k-means) ne trouvent que des clusters en boule.

— Ne tiennent pas compte d'une éventuelle structure.

— Même problème pour les estimations de densité.

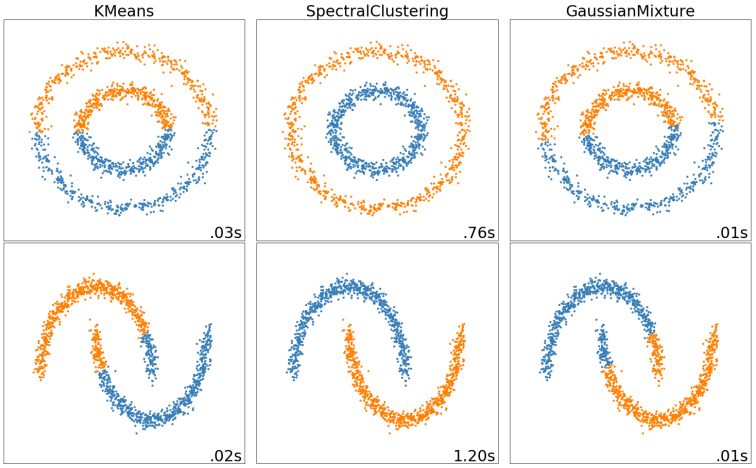
⇒ Spectral clustering :

— On projète les données sur les nœuds d'un graphe pondéré.

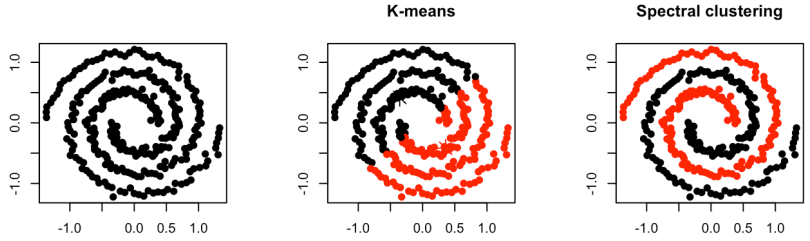
— Les arêtes modélisent la similarité entre les données.

— Le poids de chaque arête est proportionnel à la distance (dissimilarité) entre données.

Données avec structure latente -1-



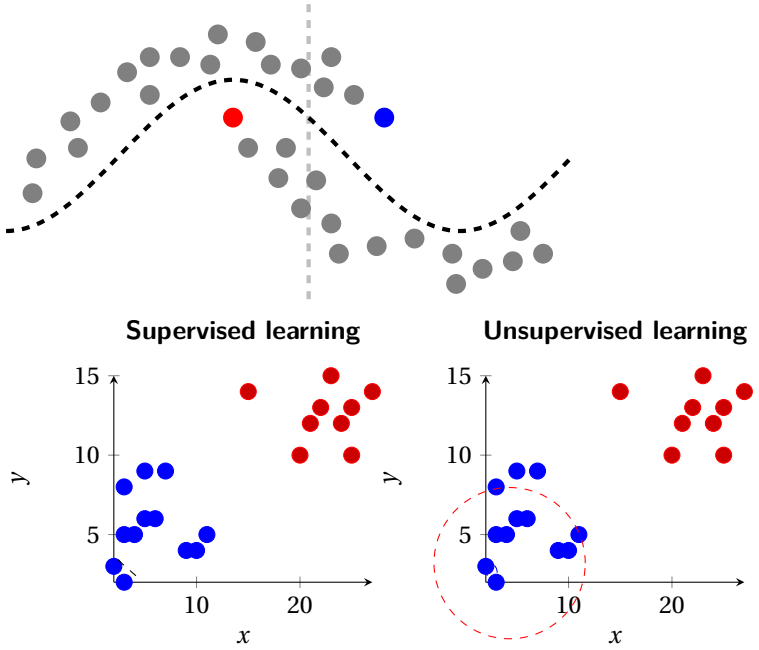
Données avec structure latente -2-



Chapitre 9

Quelques aspects de l'apprentissage semi-supervisé

À venir....



9.1 Introduction et problématique

L'apprentissage semi-supervisé forme une famille de méthodes qui se situent à l'interface entre l'apprentissage supervisé et l'apprentissage non supervisé. Son objectif est de tirer parti d'un petit ensemble de données étiquetées au milieu d'un grand nombre de données non annotées, pour améliorer la performance d'un modèle prédictif. L'idée est d'utiliser les données non annotées pour compléter l'apprentissage supervisé.

Ce type d'apprentissage est souvent associé au concept d'apprentissage transductif : méthode qui s'effectue sur les données d'apprentissage dans le but de faire des prédictions sur les observations de la base de test et uniquement celles-ci. On cherche alors à minimiser une erreur moyenne sur la base de test et pas forcément l'erreur de généralisation.

L'apprentissage semi-supervisé s'inscrit dans un contexte où les données sont difficiles ou coûteuses à étiqueter, alors que les données non annotées sont abondantes, faciles à collecter, mais inexploitables directement pour des tâches de prédiction précises. Ce cadre est particulièrement pertinent dans les situations où :

- L'étiquetage est coûteux, chronophage ou nécessite une expertise (données médicales, vidéos, documents juridiques, classification de pages internet),
- Un grand volume de données non étiquetées est disponible à faible coût (textes, images, capteurs, logs, etc.),

- On souhaite réduire la dépendance aux données étiquetées, tout en bénéficiant des avantages de l'apprentissage supervisé.

Cadre formel.

Considérons un petit échantillon étiqueté L = $\{(x_i, y_i)_{i=1,..,l}\}$, où chaque $x_i \in \mathbb{R}^d$ et $y_i \in \mathbb{Y}$, et un échantillon non étiqueté U = $\{(x_i, y_i)_{i=l+1,..,l+u}\}$.

L'objectif est d'apprendre un modèle capable de généraliser à de nouvelles données en exploitant l'information disponible dans L et U.

L'apprentissage semi-supervisé repose sur certaines hypothèses clefs quant à la structure des données, parmi lesquelles :

- Hypothèses sur la densité ou les modèles statistiques sous jacents.
- Hypothèse de continuité : des points proches dans l'espace des caractéristiques ont probablement la même étiquette.
- Hypothèse de regroupement (cluster assumption) : les données forment des groupes bien séparés, et les étiquettes sont constantes à l'intérieur de chaque groupe.
- Hypothèse de la marge : la frontière de décision optimale doit passer dans les régions de faible densité des données non étiquetées.

Ces hypothèses sont essentielles pour justifier l'utilisation d'informations non étiquetées dans la construction du modèle.

Application possibles :

— Vision par ordinateur : classification d'images, détection d'objets, segmentation.

— Traitement du langage naturel : classification de texte, analyse de sentiment, résumé automatique.

— Médecine : détection de maladies sur imagerie avec peu de données annotées.

— Cyber-sécurité : détection d'anomalies ou de menaces avec peu d'exemples labellisés.

— Industrie : maintenance prédictive, surveillance de systèmes.

Les performances de l'apprentissage semi-supervisé sont fortement dépendantes de la qualité des labels prédits (risque d'amplifier les erreurs si le modèle se trompe dès le départ). Il est par ailleurs difficile de mesurer les performances pendant l'apprentissage si peu de labels sont disponibles. Le choix des hyperparamètres : est crucial mais difficiles à optimiser sans validation supervisée. Les données non étiquetées peuvent être bruitées, hors distribution ou redondantes, ce qui amène à des étiquetages pas toujours robustes.

9.2 Principales approches

Il existe plusieurs grandes familles de méthodes semi-supervisées, dont voici une liste non exhaustive (nous ne rentrerons pas dans le détail de ces méthodes, exceptée les réseaux neuronaux sur graphe) :

1. Auto-apprentissage (self-training) : une première version du modèle est entraînée sur les données étiquetées. Il est ensuite utilisé pour prédire les étiquettes des données non étiquetées. Les prédictions étiquetées avec un haut de degré de confiance sont ajoutées à la base d'apprentissage, comme si elles étaient correctes. Le modèle est réentraîné sur ce nouvel ensemble enrichi et la procédure est répétée jusqu'à satisfaire un critère d'arrêt. Dans cette méthode, le classifieur utilise ses propres prédictions pour apprendre.

2. Co-apprentissage (co-training) : deux modèles sont entraînés sur des sous-ensembles de variables disjoints (des « features » différentes) de l'échantillon de données. Chacun fournit à l'autre des étiquettes sur les données non supervisées. C'est une méthode adaptée aux données pour lesquelles les features peuvent être divisées de façon indépendantes. Cette méthode peut être vue comme un auto-apprentissage croisé effectué sur deux classifieurs.

3. Apprentissage actif : au lieu d'exploiter les données non étiquetées, on annote de façon active uniquement les données qui apporteront le plus d'information.

4. Méthodes à base de graphes (GNN) : on construit un graphe de similarité entre tous les points (étiquetés et non étiquetés) de l'échantillon de données. Chaque nœud du graphe représente une donnée et deux nœuds sont reliés par une arête s'ils sont considérés comme proches relativement à une mesure de similarité (ou une distance

entre les variables statistiques constituant le vecteur des features). L'information sur les labels est ensuite propagée à travers le graphe. Cette méthode est basée sur l'idée (hypothèse d'« homophilie ») que des nœuds voisins connectés ont probablement les mêmes étiquettes.

5. Méthodes basées sur la régularisation : de façon générale, les régions denses du jeu de données sont considérées comme des classes bien représentées (un cluster bien déterminé) que l'on va conserver. Les modèles sont entraînés avec une fonction de coût qui favorise la régularité sur les données non annotées (mesurées par la consistance entre prédictions voisines) et qui pénalise les frontières de décision pour ne pas couper de région de grande densité. Par exemple : entropy minimization, Virtual Adversarial Training (VAT), etc.

6. Méthodes basées sur l'apprentissage profond : pseudo-étiquetage, MixMatch, FixMatch, Mean Teacher, et d'autres algorithmes très efficaces utilisent des réseaux neuronaux et diverses techniques de régularisation ou d'augmentation de données.

7. Laplacien SVM ou séparateur semi-supervisé à vaste marge (S^3VM) : On ajoute deux contraintes au problème d'optimisation traditionnel définissant les SVM, afin de maintenir les données non étiquetées à l'extérieur de la marge, tout en minimisant l'erreur de classification. De façon plus générale, les méthodes de « Manifold Learning » utilise de l'information sur la structure géométrique de la distribution marginale des données non étiquetées. L'hypothèse sous jacente est que les données vivent dans une sous-variété régulière (un sous espace vectoriel, par exemple) de plus petite dimension. La frontière de décision doit alors respecter la forme de cette sous-variété.

8. T-SVM.

9. Méthodes de mélange : les différentes classes sont mélangées, par exemple par des gaussiennes. On utilise ensuite un algorithme de type EM (Expectation Maximization) et on s'assure que le modèle s'adapte bien à la fois aux données étiquetées et à celles qui ne le sont pas.

10. Clustering semi-supervisé : on dispose d'un superviseur qui peut produire quelques données annotées ou des informations sous la forme de contraintes entre les données. On procède en apprenant une métrique avec laquelle on va utiliser un algorithme de clustering standard. On peut aussi utiliser un algorithme travaillant directement sur les contraintes.

De façon générale, beaucoup d'algorithmes reposent sur des fonctions de distance entre les données. Les performances dépendent alors directement de la qualité de cette métrique. En grande dimension, les données ont tendance à être équirépartie dans l'espace, il est donc important de choisir une métrique qui ne soit pas trop « géométrique ».

9.3 Brève introduction aux réseaux neuronaux sur graphes (GNN)

Un réseau neuronal sur graphe (Graph Neural Network, GNN) est une classe de modèles d'apprentissage profond géométrique spécialement conçus pour les données structurées en graphes [**?**, **?**, **?**, **?**, **?**]. Les GNN atteignent d'excellentes performances dans de nombreuses tâches liées aux graphes, telles que la classification de nœuds ou de graphes, la prédiction de liens, le regroupement de graphes, l'apprentissage semi-supervisé ou encore la réduction de dimension.

Les GNN utilisent un algorithme de propagation de messages comme règle de mise à jour : chaque nœud reçoit de l'information uniquement de ses voisins directs, et met à jour itérativement ses vecteurs de features, selon un principe proche de l'algorithme de propagation de croyance proposé par Pearl [**?**]. En agrégeant l'information issue du voisinage local, les GNN intègrent les données des nœuds, les poids des arêtes et la topologie du graphe dans le processus d'apprentissage.

Une couche dans un GNN correspond, pour un nœud donné, à son voisinage direct (les nœuds auxquels il est relié). L'empilement des couches permet une mise à jour itérative du processus de propagation de messages : un GNN à l couches effectue l itérations sur un même graphe, où la l-ième itération apporte à chaque nœud l'information provenant de son voisinage à l sauts (cf. Fig. 9.1). La nécessité de disposer de l itérations pour permettre à deux nœuds distants de l de communiquer est connue sous le nom de problème du rayon (problem radius) [**?**].

La formulation mathématique d'un GNN est définie par son graphe sous-jacent $G = (V, E)$, la matrice des vecteurs de features des nœuds $X \in \mathbb{R}^{n \times d}$, dans laquelle $x_v \in \mathbb{R}^d$ est la ligne de X correspondant au nœud $v \in V$, et par la règle de propagation :

$$\begin{cases} h_v^0 = x_v, \\ h_v^{l+1} = \phi_l\left(h_v^l, \sum\limits_{u \sim v} \hat{A}_{uv} \psi_l\left(h_u^l\right)\right). \end{cases} \tag{1}$$

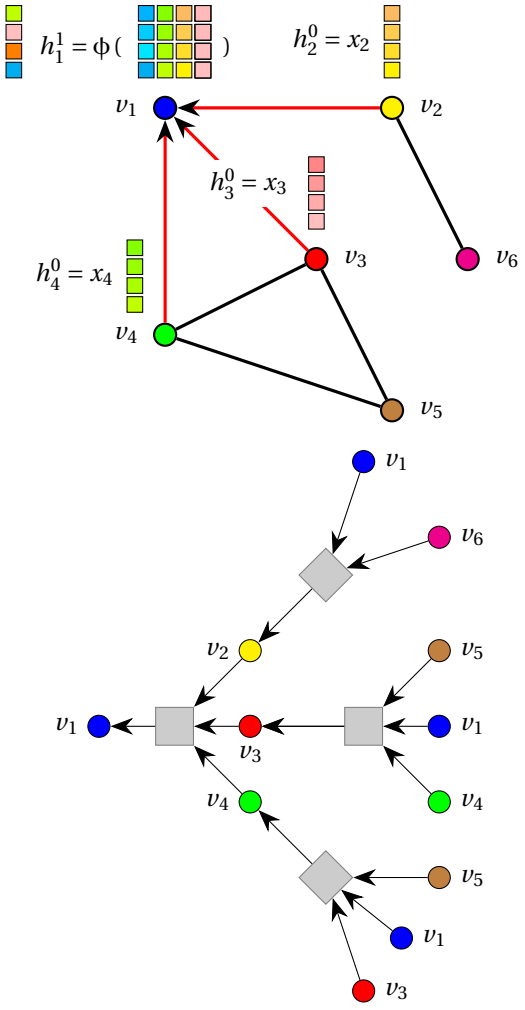


FIGURE 9.1 – Structure typique d'un GCN. En haut : le nœud cible est v_1, et l'on se situe juste avant la première itération. Le voisinage à 1 saut de v_1 (constitué de v_2, v_3, v_4) envoie leurs vecteurs de features initiaux ($h_{20} = x_2$, $h_{30} = x_3$, $h_{40} = x_4$) vers v_1 au cours de la première itération. Ces vecteurs sont alors agrégés avec $h_{10} = x_1$ à l'aide de la fonction ϕ. Le résultat est le vecteur de features mis à jour h_{11}, qui remplace x_1 et sera ensuite envoyé à tous les voisins de v_1 lors de l'itération 2. En bas : arbre correspondant au voisinage à 2 sauts de v_1. La première couche agrège les messages en provenance du voisinage à 1 saut, ce qui correspond à l'opération décrite dans le schéma de gauche, où les vecteurs de features de v_2, v_3, v_4 sont transmis et agrégés au niveau de v_1. Le même processus est appliqué à tous les nœuds, en particulier à v_2, v_3, v_4. La seconde couche agrège les messages provenant du voisinage à 2 sauts de v_1, c'est-à-dire les nœuds situés à distance 2 de v_1. Les messages mis à jour dans ces nœuds parviennent au nœud initial au cours de la deuxième itération. Les carrés gris représentent le processus d'agrégation, modélisé par les fonctions ϕ_1. Les nœuds du graphe sous-jacent coïncident avec les nœuds du réseau de neurones : 1 couche de GNN = 1 itération de l'algorithme de propagation de messages.

Pour chaque nœud v, $h_v^l \in \mathbb{R}^d$ est le vecteur de features de v mis à jour à l'iétration l, ϕ_l et ψ_l sont les fonctions à apprendre, qui sont supposées différentiables. $\hat{A} = A + I$ est la matrice d'adjacence modifiée de G, qui inclut des boucles. Les boucles permettent à un nœud d'intégrer sa propre information à celle de ses voisins.and are known to prevent the phenomenon of over-smoothing described below.

For the sake of simplicity, we present our results on a typical Graph Convolutional Network (GCN) [**?**] whose simple formal expression is given in matrix form :

$$H_{l+1} = \sigma\left(D^{-1/2}\hat{A}D^{-1/2}H_l W_l\right), \tag{2}$$

with $H_l = \left(h_v^l\right)_v \in \mathbb{R}^{n\times d}$, $D = \mathrm{diag}(d_{ii})$, $d_{ii} = \sum_j \hat{A}_{ij}$, W_l is the trainable weight matrix. $\sigma = \phi_l$ is an activation function (ReLU, sigmoid, etc.) and $\psi_l = \mathrm{Id}$.

Over-smoothing, over-squashing and under-reaching are three issues that impair the performance of a GNN when using message passing algorithms. Over-smoothing occurs when node features quickly converge toward a

common average and become indistinguishable [**?**]. Under-reaching happens when the network is not deep enough to convey information from distant nodes due to the problem of radius [**?**]. To prevent over-smoothing, the depth of the network should not be too great, while the opposite is necessary to prevent under-reaching. Although these two phenomena are now well understood, less is known about over-squashing, which measures the difficulty of propagating information between distant nodes, often due to bottlenecks between nodes in certain parts of the graph [**?**]. While under-reaching is solely due to the neighborhood radius, over-squashing is linked to the topology and connectivity of the entire graph and seems to occur mainly in tasks that depend on long-range interactions [**?**].

To prevent over-squashing, the most common technique is local rewiring, which involves adding, removing or changing weights of edges selected to optimize certain properties related to the graph topology, thereby reducing bottlenecks. The property to optimize can be graph curvature [**?**], algebraic connectivity [**?**], commute time distance [**?**], effective resistance [**?**, **?**, **?**] or the use of specific graphs such as expanders [**?**] or complete graphs [**?**].

9.3.1 Application to GNNs : learning on the sparsified graph

As an application of our sparsification methods, we propose to learn on the sparsified graph of a GNN before any training or learning process. The objective is to reduce the computational resources required to store the graph and to speed up the training and processing of the data, while preserving connectivity. We aim to study the extent to which performance is affected by the sparsification process and how connectivity optimization limits this effect and the occurrence of over-squashing.

Our approach differs from previous work in several key aspects :

- We do not improve the connectivity of the graph by locally adding edges, but at the contrary we aim to preserve it in a simplified version of the graph.
- We adopt a global strategy, optimal within greedy methods, to preserve the graph's connectivity while significantly simplifying it by deleting a substantial number of edges (typically up to 30% or 50%).
- Our method is completely independent of the GNN architecture, which does not need to be modified; we simply replace the initial graph with its sparsifier before any operation are performed.

Our application benefits from the low complexity of our algorithms : to the best of our knowledge, a quadratic complexity in the number of operations is the lowest existing complexity for a deterministic sparsification algorithm optimizing connectivity in GNN. The greedy total resistance (GTR) algorithm [**?**] has a complexity of $O(n^3)$ and the first-order spectral rewiring (FoSR) algoritm [**?**] has a complexity of $O(kn^2)$ operations; the stochastic discrete Ricci flow (SDRF) [**?**] and the Random local edge flip (RLEF) algorithms [**?**] also have higher complexity. [**?**] proposed an implementation in $O(nm \ln n)$ operations, similar to the Spielman-Teng random algorithm, making it suitable only for weighted graphs.

9.3.2 Experiments

The following simulations demonstrate the performance of several GCNs on classic graph datasets run on a node classification task : the three Planetoid citation networks Citeseer, Pubmed and Cora, whose size is given in Table. 9.1. The nodes represent documents with bag-of-words feature vectors, and the edges are citation links between research papers. The different classes correspond to research fields and the model is trained to predict missing labels.

TABLEAU 9.1 – Citation network datasets.

Name	#nodes	#edges	#features	#classes
Cora	2708	5278	1433	7
CiteSeer	3327	4552	3703	6
PubMed	19717	44324	500	3

Given a target number k of edges ($n \leq k \leq m$), for each dataset, a GCN architecture is built around five different graphs : the original dataset's underlying graph and four other graphs sparsified to k edges. The sparsified graphs include two generated by our algorithms GSMVDIV and GSMVCORR, one with edges sampled uniformly at random and one sparsified using the Spielman-Teng algorithm [**?**]. All GCNs are trained and run on a node classification

task. The accuracy curves for training and testing data are then assessed for different values of the parameters, such as the number and dimension of the hidden layers.

TABLEAU 9.2 – Models of the simulations.

Name	Underlying graph
Gi	Base : initial graph
GSMVDIV	GSMVDIV
GSMVCORR	GSMVCORR
Spielman-Teng [**?**]	Spielman-Teng
Random	Random

Name of the different models of GNN and their underlying graph.

TABLEAU 9.3 – Performance for the Cora dataset.

#layers	dim HL	Gi	GSMVCORR	Spielman-Teng	Random
4	64	82,1±0,5	81,2±0,6	80,2±0,4	78,5±0,5
	16	80,0±0,9	79,5±0,6	77,1±0,7	76,4±0,9
	8	74,2±2,1	74,4±0,8	71,3±2,0	71,9±1,5
8	64	82,9±0,5	80,6±0,4	78,5±0,4	77,6±0,3
	16	78,6±0,8	78,5±0,6	74,4±0,8	73,5±0,7
	8	77,6±1,0	75,9±2,0	72,7±1,5	73,1±1,5
12	64	82,6±0,8	80,6±0,4	79,5±0,5	77,8±0,6
	16	80,7±1,2	79,3±0,8	78,7±1,0	75,4±1,5
	8	76,1±2,1	75,9±1,7	73,9±1,9	69,1±2,4
24	64	82,5±0,8	78,7±2,4	74,7±3,5	71,7±3,3
	16	60,3±6,5	58,4±4,7	47,3±8,1	44,8±8,1
	8	48,9±8,1	48,6±6,4	40,4±6,2	39,7±7,5

Accuary of the test set in node classification task using the (largest connected composant of) Cora dataset, for a GCN model with 4 different underlying graphs, as a function of the number and dimension of hidden layers. The best performance among the 3 sparsified graphs is highlighted in red. Each accuracy value represents the average of 10 independent samples, with the standard deviation indicated as ±.

Regarding Cora and Citeseer datasets and in most cases, PubMed, the model GSMVCORR outperforms the Spielman-Teng algorithm and always outperforms the randomly sparsified graph. The performance of GSMVCORR remains close to that of the initial model Gi, thus demonstrating that the sparsification does not significantly affect the classification task.

Tables 9.3, 9.4 and 9.5 present the performance metrics for respectively the Cora, CiteSeer and PubMed datasets. The difference in performance between Gi and GSMVCORR remains small, regardless of the number and the dimension of the layers, even for a significant percentage of edges deleted. As the dimension of the hidden layers decreases and the number of layers increases, the performance gap between the group Gi, GSMVCORR and the two models Spielman-Teng and Random widens. Over-squashing, known to appear with a high number of layers of small dimension, might be mitigated by the maximization of connectivity, potentially explaining the strong performance of our algorithms. Figure 9.3 illustrates these points for the Cora dataset by showing the accuracy curves for all graphs as a function of the number of layers.

The performance improvements are significant only for homophilic graphs, where the neighbors of a node provide relevant information about its state. In the case of heterophilic graphs, the topology does not contribute as effectively.

TABLEAU 9.4 – Performance for the CiteSeer dataset.

#layers	dim HL	Gi	GSMVCORR	Spielman-Teng	Random
4	64	74,6±0,5	74,6±0,4	73,1±0,5	72,5±0,6
	16	75,5±0,7	75,9±0,3	74,7±0,7	73,4±0,5
	8	75,9±1,2	75,8±1,1	75,1±0,4	74,4±0,6
8	64	73,9±0,5	73,3±0,2	72,9±0,5	71,8±0,4
	16	74,3±0,7	74,3±0,5	73,2±0,6	72,5±0,6
	8	73,4±0,9	74,1±1,1	72,9±0,7	72,1±0,6
12	64	73,5±0,7	73,2±0,4	73,5±0,5	72,5±0,6
	16	73,3±1,0	73,4±0,8	72,7±0,8	72,4±0,8
	8	71,4±1,3	71,2±0,8	71,1±1,1	69,5±1,0
24	64	72,9±1,0	72,4±0,8	72,1±0,7	69,9±1,7
	16	48,8±1,2	46,5±7,0	47,7±8,0	38,0±8,7
	8	36,9±9,3	32,3±3,7	29,9±6,3	28,3±3,4

Accuary of the test set in node classification task using the (largest connected composant of) CiteSeer dataset, for a GCN model with 4 different underlying graphs, as a function of the number and dimension of hidden layers. The best performance among the 3 sparsified graphs is highlighted in red. Each accuracy value represents the average of 10 independent samples, with the standard deviation indicated as ±.

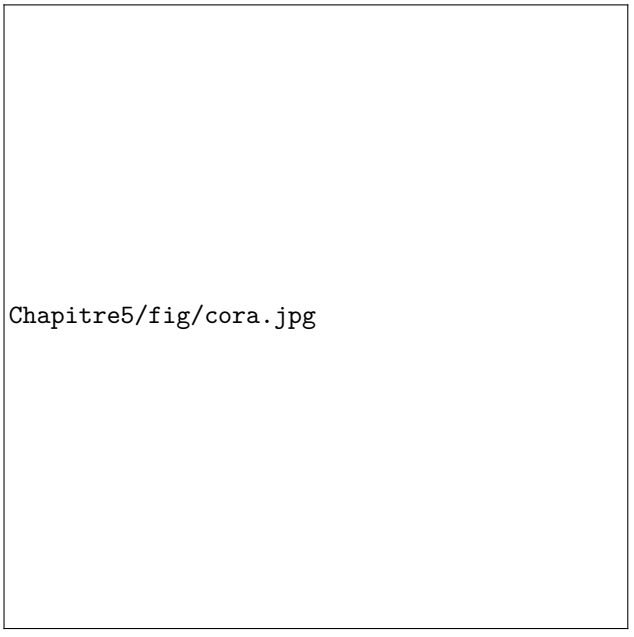


FIGURE 9.2 – Biggest connected component of Cora dataset. Nodes color correspond to the different features value.

Our algorithms have better performance for almost all parameter configurations, along with lower complexity. The connectivity is thus a crucial property to preserve and optimizing connectivity proves to be an effective solution to prevent over-squashing phenomenon.

Datasets configuration

For each dataset, the training set is extended to half of the data, the other half being the test set (using a random state parameter of 42). If the graph is not connected, the largest connected component is selected and the simulations are run on this subgraph.

The GNN architecture is a classical GCN as defined in [**?**], with some minor modifications : a linear classifier is

TABLEAU 9.5 – Performance for the PubMed dataset.

#layers	dim HL	Gi	GSMVCORR	Spielman-Teng	Random
4	64	88,9±0,2	88,7±0,1	88,3±0,3	88,1±0,1
	16	88,5±0,1	88,4±0,2	88,3±0,1	87,7±0,2
	8	87,3±0,1	87,7±0,3	87,4±0,1	86,7±0,2
8	64	88,4±0,1	88,3±0,3	87,8±0,6	87,0±0,2
	16	88,0±0,3	87,5±0,1	87,7±0,1	87,3±0,2
	8	87,5±0,1	86,7±0,5	87,3±0,2	86,9±0,3
12	64	88,2±0,3	88,0±0,2	87,8±0,1	86,8±0,2
	16	87,6±0,1	87,4±0,1	87,4±0,2	86,8±0,1
	8	86,9±0,6	86,8±0,7	87,0±0,1	86,8±0,4
24	64	86,2±0,3	85,0±0,5	85,0±0,5	83,9±0,1
	16	83,1±0,8	82,9±1,0	82,6±2,0	77,6±0,8
	8	69,1±3,9	73,5±4,0	59,7±9,8	56,8±6,5

Accuary of test set in the node classification task using the (biggest connected composant of) PubMed dataset, for a GCN model with 4 different underlying graphs, as a function of the number and dimension of hidden layers. The best performance among the 3 sparsified graphs is highlighted in red. Each accuracy value represents the average of 10 independent samples, with the standard deviation indicated as ±.

Chapitre5/fig/Shadecurveb.png

FIGURE 9.3 – Accuary of the test set in a node classification task, using the (largest connected component of the) Cora dataset, with a GCN architecture, for different underlying graphs, as a function of the number of layers. The initial graph has $n = 2485$ nodes and 5069 edges. Blue : Gi, red : GSMVCORR (GCorr), green : Spielman-Teng (ST), yellow : Random. The shaded confidence interval represents ± standard deviation over 10 independent samples. Each hidden layer has a dimension of only 8 and the number of edges in the sparsified graphs is 2500 (50% of the initial graph). Note that the sparsified graph is nearly a spanning tree.

defined as the first layer, a sequence of GCNConv forms the hidden layers, which end with another linear classifier. The output of each layer, after applying the non-linear activation function, is added with to the input vector (or the residual after the first GCNConv layer) to serve as a residual connection preventing over-smoothing.

Some parameters are fixed throughout all simulations and their values are provided in Tab. 9.6.

The dimension of each hidden layer varies from 8 to 64, and the number of layers ranges from 3 to 27.

TABLEAU 9.6 – Fixed parameters.

Parameter	Value
Learning rate	5×10^{-3}
Weight decay	5×10^{-4}
Dropout	0,5
Epochs	200

9.4 Conclusion

In this work, we proposed two greedy algorithms to sparsify a graph while preserving its connectivity. These algorithms emerge from a geometric interpretation of the volume of the graph Laplacian matrix and modify the sprectum of the initial graph to optimize its robustness.

Both algorithms are deterministic, adapted to unweighted graphs and of reasonable quadratic complexity in time. We provided a detailed implementation of several variants and empirically demonstrated that they perform better than state of the art sparsification algorithms.

We proposed an application to GNNs that simplifies the architecture and accelerates the processes executed on the network, without significantly degrading performance. Our method seems to be an effective way to limit the over-squashing phenomenon, but only for homophilic graphs where the neighbors of a node bring meaningful information.

Chapitre 10

Apprentissage statistique et économétrie

À venir....

Bibliographie

[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.