Table des matières

Chapitre I

Apprentissage Statistique

I.1 Introduction

Nous avons vu en introduction de ce cours que la statistique comprend deux pans :
- la statistique décisionnelle (ou inférencielle) qui utilise les bases de données pour prédire la valeur de variables non observées
- la statistique descriptive qui a pour but de décrire les liens existant entre les différentes variables observées.

En statistique décisionnelle, les deux types de modèles considérés sont les modèles paramétriques (qui basent leur prédiction sur un nombre de paramètres fini indépendant de la taille de la base de données) et les modèles non paramétriques.

L'apprentissage statistique est la branche non paramétique de la statistique décisionnelle qui s'intéresse aux bases de données composées de n couples, souvent appelés couples entrée-sortie, supposés *indépendants et identiquement distribués*. Le but d'un algorithme d'apprentissage statistique est de proposer pour toute nouvelle entrée une prédiction de la sortie associée à cette entrée.

Les procédures d'apprentissage statistique sont utiles lorsqu'une modélisation paramètrique de la loi générant les données n'est pas accessible ou lorsque la complexité du modèle est telle qu'elle empêche son utilisation pour la prédiction.

Ces méthodes sont devenues incontournables dans de nombreuses applications pratiques (classement et analyse d'images, reconnaissance d'objets, classement de documents textuels (par exemple : pourriel vs non pourriel), diagnostic médical, analyse de séquences génétiques ou de protéines, prédiction du rendement d'actifs financiers, interface cerveau-machine, ...).

I.2 Description formelle et exemples

I.2.1 Problématique

Nous observons une base de données composée de n couples $Z_1 = (X_1, Y_1), \ldots, Z_n = (X_n, Y_n)$ que nous supposons être des réalisations indépendantes d'une même loi $\mathbb{P}$ inconnue. Les $X_1, \ldots, X_n$ appartiennent à un espace $\mathcal{X}$ et s'appellent les entrées. Typiquement, $\mathcal{X} = \mathbb{R}^d$ pour un grand entier d. Les $Y_1, \ldots, Y_n$ appartiennent à un espace $\mathcal{Y}$, et s'appellent les sorties. Typiquement, $\mathcal{Y}$ est fini ou $\mathcal{Y}$ est un sous-ensemble de $\mathbb{R}$.

But de l'apprentissage statistique : prédire la sortie Y associée à toute nouvelle entrée X, où il est sous-entendu que la paire (X, Y) est une nouvelle réalisation de la loi $\mathbb{P}$, cette réalisation étant indépendante des réalisations précédemment observées.

Une *fonction de prédiction* est une fonction (mesurable) de $\mathcal{X}$ dans $\mathcal{Y}$. Dans ce chapitre, nous supposons que toutes les quantités que nous manipulons sont mesurables. L'ensemble de toutes les fonctions de prédiction est noté $\mathcal{F}(\mathcal{X}, \mathcal{Y})$. La base de données $Z_1, \ldots, Z_n$ est appelée *ensemble d'apprentissage*, et sera parfois notée Z_1^n. Un *algorithme d'apprentissage* est une fonction qui à tout ensemble d'apprentissage renvoie une fonction de prédiction, i.e. une fonction de l'union $\cup_{n\geq 1}\mathcal{Z}^n$ dans l'ensemble $\mathcal{F}(\mathcal{X}, \mathcal{Y})$, où $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$. C'est un estimateur de "la meilleure" fonction de prédiction, où le terme "meilleure" sera précisé ultérieurement.

Soit $\ell(y, y')$ la perte encourue lorsque la sortie réelle est y et la sortie prédite est y'. La fonction $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ est appelée *fonction de perte*.

Exemple du classement : $\ell(y, y') = \mathbf{1}_{y \neq y'}$ (i.e. $\ell(y, y') = 1$ si $y \neq y'$ et $\ell(y, y') = 0$ sinon). Un problème d'apprentissage pour lequel cette fonction de perte est utilisée est appelé problème de *classement* (ou plus couramment par anglicisme *classification*). L'ensemble $\mathcal{Y}$ considéré en classement est le plus souvent fini, voire même de cardinal deux en *classement binaire*.

Exemple de la régression L_p *:* $\mathcal{Y} = \mathbb{R}$ et $\ell(y, y') = |y - y'|^p$ où $p \geq 1$ est un réel fixé. Dans ce cas, on parle de régression L_p. La tâche d'apprentissage lorsque $p = 2$ est aussi appelée *régression aux moindres carrés*.

La qualité d'une fonction de prédiction $g : \mathcal{X} \to \mathcal{Y}$ est mesurée par son *risque* (ou *erreur de généralisation*) :

$$R(g) = \mathbb{E}\big[\ell\big(Y, g(X)\big)\big]. \tag{I.1}$$

Le risque est donc l'espérance par rapport à loi $\mathbb{P}$ de la perte encourue sur la donnée (X, Y) par la fonction de prédiction g. La qualité d'un algorithme d'apprentissage $\hat{g}_n$, construit à partir de Z_1^n, peut être mesurée par son risque moyen $\mathbb{E}R[\hat{g}_n]$, où il est sous-entendu que l'espérance est prise par rapport à la loi de l'ensemble d'apprentissage.

La "meilleure" fonction de prédiction est la (ou plus rigoureusement une) fonction de $\mathcal{F}(\mathcal{X}, \mathcal{Y})$ minimisant R. Une telle fonction n'existe pas nécessairement mais existe pour les fonctions de pertes usuelles (notamment celles que nous considérerons par la suite). Cette "meilleure" fonction sera appelée *fonction cible* ou *fonction oracle*.

Exemple du classement : $\ell(y, y') = \mathbf{1}_{y \neq y'}$. La fonction qui à une entrée x renvoie la sortie la plus probable (au sens de la distribution conditionnelle de Y sachant $X = x$: $\mathcal{L}(Y|X = x)$) est "la" fonction cible en classement.

Exemple de la régression aux moindres carrés : $\mathcal{Y} = \mathbb{R}$ et $\ell(y, y') = |y - y'|^2$. La fonction qui à une entrée x renvoie la sortie moyenne $\mathbb{E}(Y|X = x)$ est "la" fonction cible en régression aux moindres carrés.

I.2.2 Exemples

Dans ce paragraphe, nous proposons des exemples illustrant la problématique précédente.

Exemple I.1. La reconnaissance de caractères manuscrits est un des problèmes sur lequel les méthodes d'apprentissage ont permis des avancées fulgurantes. Le contexte est le suivant :

nous disposons d'une image numérisée d'un caractère manuscrit. Cette image est essentiellement un tableau de nombre réels indiquant l'intensité lumineuse en chacun des pixels. Nous souhaitons trouver la fonction qui à ce tableau de réels renvoie le caractère présent dans l'image. A l'heure actuelle, les meilleures méthodes pour trouver une telle fonction sont de nature statistique : elles reposent donc sur

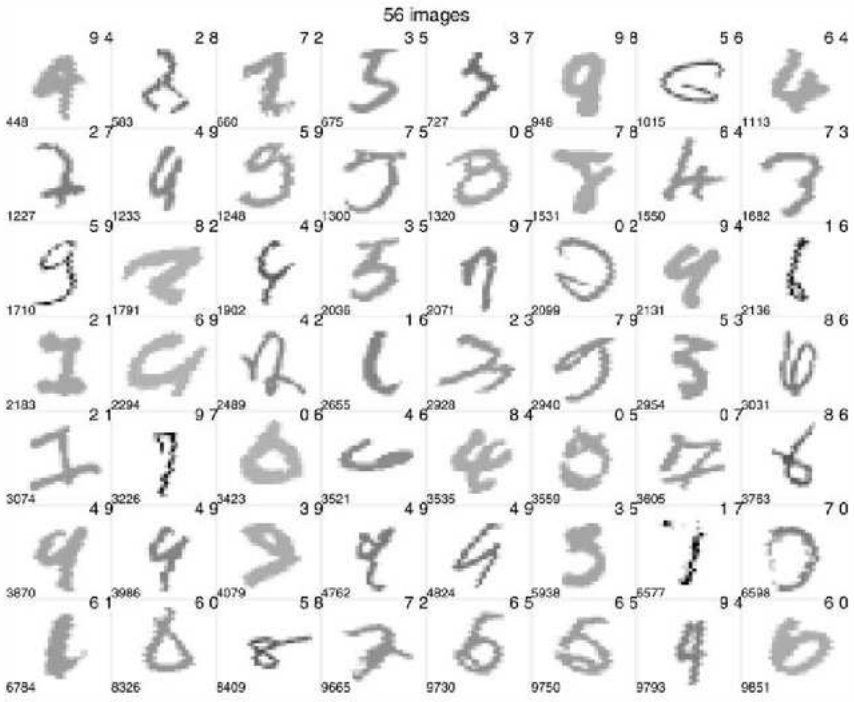

FIG. I.1 – Reconnaissance de chiffres manuscrits. Les 56 erreurs sur les 10 000 caractères de la base de test (MNIST/VSV2) d'un des meilleurs algorithmes de reconnaissance de caractères manuscrits [2, 7]. Le premier nombre en haut à droite indique la valeur prédite et le second indique la vraie valeur. Le nombre en bas à gauche est le numéro de l'image (de 1 à 10 000).

1. la constitution d'une base d'images de caractères où les images sont étiquetées par le caractère qu'elle contient. Un X_i correspond donc à une de ces images et un Y_i désigne le caractère que X_i contient.
2. l'utilisation de cette base pour proposer une estimation non paramétrique de la fonction cible.

Le taux d'erreur sur la reconnaissance de chiffres manuscrits (problème intéressant notamment les centres de tri postaux) sont de l'ordre de 0.5% pour les meilleurs algorithmes (voir figure I.1). ◊

Exemple I.2. Ces dernières années ont vu un essor de la recherche sur la manière d'interfacer le cerveau humain avec un ordinateur. Un des enjeux fondamentaux de ce domaine est de permettre aux personnes ayant perdu l'usage de leurs mains de communiquer avec un ordinateur.

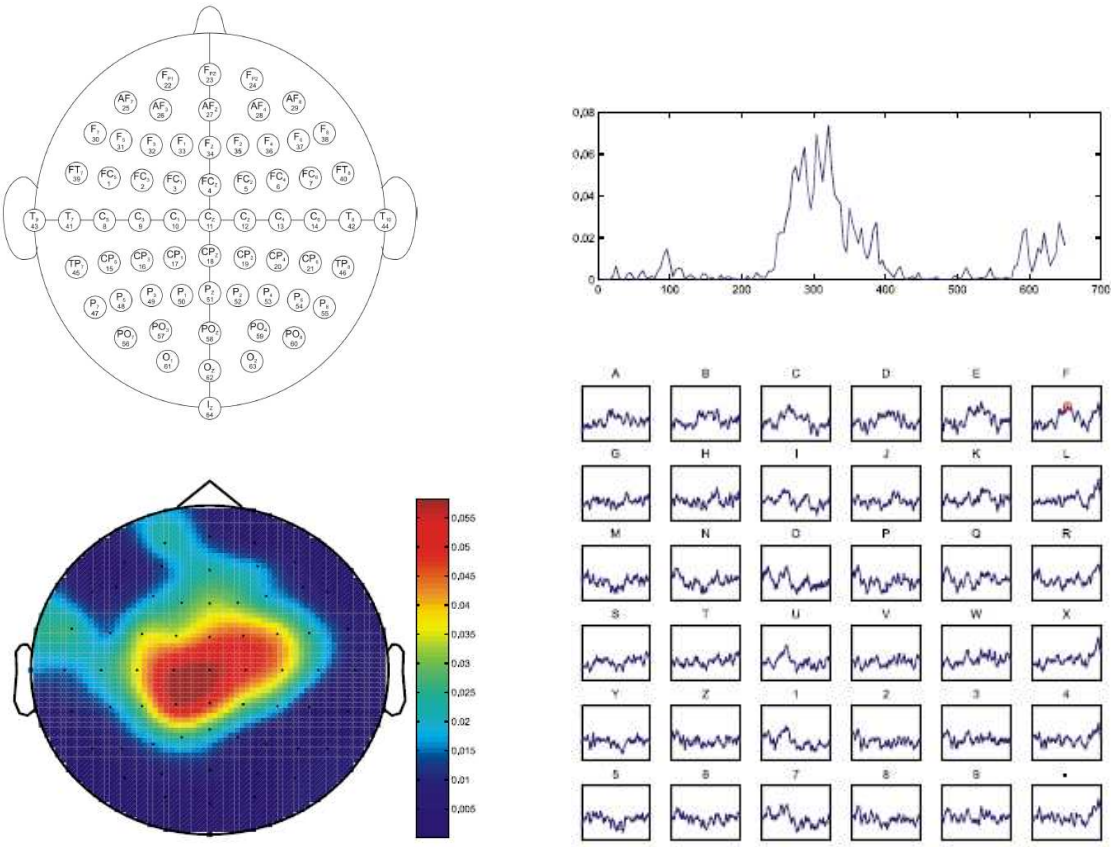

FIG. I.2 – *En haut à gauche :* exemple d'emplacement d'électrodes sur la surface du cerveau du sujet vu de haut. *En haut à droite :* exemple de signal électrique observé à une électrode durant les 700 ms suivant le stimulus. L'activité observée environ 300 ms après le stimulus est caractéristique de la réaction du sujet au stimulus. *En bas à gauche :* topographie (vue de haut) de la variance du signal due à la présence ou non de stimuli. Cette topographie montre que le signal est relativement peu localisé en un point précis de la surface du cerveau. *En bas à droite :* signaux électriques moyens observés à une électrode pour différents stimuli [1, 8].

Une des méthodes d'interface cerveau-machine consiste à placer des électrodes à la surface de leur cerveau (méthode non invasive) ou à l'intérieur (méthode invasive). Le but est de déduire de l'observation des signaux électriques les pensées et/ou volontés du sujet. A l'heure actuelle, ces méthodes reposent sur

- la constitution d'une base d'apprentissage : il est demandé à un (ou des) sujet(s) sous observation à penser à quelque chose de précis (par exemple, bouger la souris vers le haut, le bas, la gauche ou la droite ; autre exemple : à une des lettres de l'alphabet). La nature de cette pensée est une sortie Y_i associée à l'entrée X_i qui est le signal électrique observé pendant la seconde suivant la requête.
- l'apprentissage de la fonction cible à partir de cette base de signaux électriques étiquetés.

◊

Exemple I.3. Pour faire face aux fluctuations incontrôlées du marché, les banques proposent

aujourd'hui des produits dont les fluctuations sont indépendantes de la tendance (baissière ou haussière) du marché. Ces placements, dits de gestion alternative, reposent sur l'achat des actions qui vont croître le plus (ou du moins baisser le moins) et la vente pour le même montant des actions qui vont baisser le plus (ou croître le moins). La difficulté pour mettre en place ce type de produit est d'identifier ces actions "sur-performantes" et "sous-performantes".

Une méthode utilisée par les banques est

- de recenser un ensemble de paramètres caractérisant l'action. Ces paramètres proviennent autant des analystes techniques (qui étudie les courbes à travers des paramètres tels que la moyenne mobile, les seuils de résistance, ...) que des analystes financiers (qui étudie les paramètres de la société : chiffre d'affaires, bénéfices, indices de rentabilité, ...).
- de constituer une base de données où une entrée X_i est un vecteur des paramètres décrits ci-dessus et la sortie Y_i associée évalue la sur/sous-performance de l'action sur la période suivant l'observation de ces paramètres (typiquement de l'ordre d'une semaine)
- l'apprentissage de la fonction cible qui, à une date donnée, pour chaque action du marché, associe aux paramètres observés l'indice de sur/sous-performance de l'action.

$\Diamond$

I.2.3 Lien entre classement binaire et régression aux moindres carrés

Dans cette section, nous considérons le problème de prédiction binaire, c'est-à-dire où la sortie ne peut prendre que deux valeurs. C'est en particulier la problématique des logiciels de lutte contre les pourriels (ou, par anglicisme, spam). Sans perte de généralité, nous pouvons considérer : $\mathcal{Y} = \{0; 1\}$. Le théorème suivant précise le lien entre classement binaire et régression aux moindres carrés dans le contexte de la prédiction binaire.

Considérons $\mathcal{Y} = \{0; 1\}$. Soit η^* la fonction cible en régression aux moindres carrés définie par $\eta^*(x) = \mathbb{E}(Y|X = x) = \mathbb{P}(Y = 1|X = x)$. Soit g^* la fonction cible en classement définie par

$$g^*(x) = \mathbf{1}_{\eta^*(x)\geq 1/2} = \begin{cases} 1 & \text{si } \mathbb{P}(Y = 1|X = x) \geq 1/2 \\ 0 & \text{sinon} \end{cases}$$

Pour toute fonction de régression $\eta : \mathcal{X} \to \mathbb{R}$, on définit la fonction de classement $g_\eta \triangleq \mathbf{1}_{\eta\geq 1/2}$.

Théorème I.4. *Nous avons*

$$R_{cla}(g_\eta) - R_{cla}(g^*) \leq 2\sqrt{R_{reg}(\eta) - R_{reg}(\eta^*)},$$

où R_{cla} et R_{reg} désignent respectivement les risques en classement et en régression aux moindres carrés : précisément $R_{cla}(g_\eta) = \mathbb{P}[Y \neq g_\eta(X)]$ et $R_{reg}(\eta) = \mathbb{E}\big[\big(Y - \eta(X)\big)\big]$.

Autrement dit, si η est une "bonne" fonction de régression, alors sa version seuillée g_η est une "bonne" fonction de classement.

I.2.4 Consistance universelle

Définition I.5. *Un algorithme d'apprentissage est dit* consistant par rapport à la loi *de (X, Y) si et seulement si*

$$\mathbb{E}R(\hat{g}_n) \underset{n\to+\infty}{\longrightarrow} R(g^*).$$

Un algorithme d'apprentissage est dit universellement consistant *si et seulement si il est consistant par rapport à toute loi possible du couple* (X, Y).

Théorème I.6. *Si un algorithme* $\hat{\eta}$ *est universellement consistant pour la régression aux moindres carrés à sorties dans* $[0;1]$ $\big($*i.e.* $\mathcal{Y} = [0;1]$, $\ell(y, y') = (y - y')^2\big)$, *alors l'algorithme* $\hat{g} = \mathbf{1}_{\hat{\eta} \geq 1/2}$ *est universellement consistant pour le problème de classement binaire à sorties dans* $\{0;1\}$ $\big($*i.e.* $\mathcal{Y} = \{0;1\}$, $\ell(y, y') = \mathbf{1}_{y \neq y'}\big)$.

Démonstration. Le point de départ consiste à remarquer que si $\hat{\eta}$ est universellement consistant pour la régression aux moindres carrés à sorties dans $[0;1]$, alors $\hat{\eta}$ est en particulier consistant par rapport à toute distribution telle que les sorties sont dans $\{0;1\}$ avec probabilité un (i.e. presque sûrement). Le résultat annoncé est une conséquence directe du théorème I.4 et du théorème de Jensen. □

I.3 Les algorithmes d'apprentissage et leur consistance

Dans ce paragraphe[1], nous considérons (sauf mention contraire) le problème de régression aux moindres carrés défini par $\mathcal{X} = \mathbb{R}^d$, $\mathcal{Y} = [-B; B]$ pour $B > 0$ et $\ell(y, y') = (y - y')^2$. Une fonction cible est alors $\eta^* : x \mapsto \mathbb{E}(Y|X = x)$.

I.3.1 Algorithmes par moyennage locale

Idée principale : Pour estimer $\mathbb{E}(Y|X = x)$, moyenner les Y_i des X_i proches de x.

Cette idée nous amène à nous intéresser aux algorithmes d'apprentissage de la forme

$$\hat{\eta} : x \mapsto \textstyle\sum_{i=1}^n W_i(x) Y_i,$$

où les poids réels $W_i(x)$ vont être des fonctions bien choisies de $x, n, X_1, \ldots, X_n$.

Exemple 1 : Algorithme par partition : soit $\{A_1, A_2, \ldots\}$ une partition finie ou dénombrable de $\mathcal{X}$ (i.e. $\cup_j A_j = \mathcal{X}$ et $A_j \cap A_k = \emptyset$ pour $j \neq k$). Soit $A(x)$ l'élément de la partition qui contient x. L'algorithme par partition considère les poids

$$W_i(x) = \frac{\mathbf{1}_{X_i \in A(x)}}{\sum_{l=1}^n \mathbf{1}_{X_l \in A(x)}}, \tag{I.2}$$

où nous utilisons la convention $\frac{0}{0} = 0$.

Exemple 2 : Algorithme par noyau (ou *estimateur de Nadaraya-Watson*) : Soient $K : \mathbb{R}^d \to \mathbb{R}_+$ et $h > 0$ un paramètre (dit de largeur du noyau). L'algorithme par noyau considère les poids

$$W_i(x) = \frac{K(\frac{x - X_i}{h})}{\sum_{l=1}^n K(\frac{x - X_l}{h})}, \tag{I.3}$$

où nous utilisons toujours la convention $\frac{0}{0} = 0$. Les deux noyaux les plus courants sont le noyau fenêtre $K(x) = \mathbf{1}_{\|x\| \leq 1}$ et le noyau gaussien $e^{-\|x\|^2}$, avec $\|\cdot\|$ la norme euclidienne.

Exemple 3 : L'*algorithme des k-plus proches voisins* (k-p.p.v.) considère les poids

$$W_i(x) = \begin{cases} \frac{1}{k} & \text{si } X_i \text{ fait partie des } k\text{-p.p.v. de } x \text{ dans } X_1, \ldots, X_n \\ 0 & \text{sinon} \end{cases}. \tag{I.4}$$

[1] Ce paragraphe s'inspire largement des six premiers chapitres du livre [4].

L'algorithme des k-plus proches voisins

Pour définir parfaitement l'algorithme des k-p.p.v., il faut préciser ce qui se passe lorsque le point x à classer est à égale distance de plusieurs points de l'ensemble d'apprentissage. La règle la plus simple consiste à traiter ces problèmes d'égalité de distances par tirage au sort : pour une distance d et $x \in \mathbb{R}^d$, si l'ensemble $E = \big\{i \in \{1,\dots,n\} : \|X_i - x\| = d\big\}$ est de cardinal $|E| \geq 2$, alors les points $(X_i)_{i\in E}$ sont ordonnés en tirant une permutation aléatoire (suivant la loi uniforme sur l'ensemble des $|E|!$ permutations de E).

Pour cette gestion des égalités éventuelles de distances, nous avons

Théorème I.7. *Si $k_n \underset{n\to+\infty}{\longrightarrow} +\infty$ et $k_n/n \underset{n\to+\infty}{\longrightarrow} 0$, l'algorithme des k_n-p.p.v. est universellement consistant.*

- L'algorithme des k-p.p.v. nécessite de conserver en mémoire tous les points de la base d'apprentissage, d'où un stockage coûteux (en $\mathrm{O}(n)$).
- Une implémentation naïve de l'algorithme repose sur le calcul des distances entre le point pour lequel la prédiction est recherchée et les points de l'ensemble d'apprentissage, d'où un algorithme en $\mathrm{O}(n)$. Des méthodes basées sur la construction d'un arbre associé à l'ensemble d'apprentissage ont un coût moyen en $\mathrm{O}(\log n)$. Cela nécessite néanmoins la construction de l'arbre, ce qui a en général un coût en $\mathrm{O}(n \log n)$ (voir par exemple 'kd-tree' sur `en.wikipedia.org`). Une implémentation de cette méthode est accessible en ligne sous le nom d'¡¡approximate nearest neighbour¿¿ (`www.cs.umd.edu/~mount/ANN/`) qui permet également de rechercher les plus proches voisins de manière approchée (ce qui mène à un gain de temps considérable). Une astuce pour améliorer la recherche approchée des plus proches voisins est de construire plusieurs arbres de recherche (si possible les plus différents possibles).
- Le choix du paramètre k peut être effectué par une méthode dite de validation croisée que nous décrivons ci-dessous. Cette méthode, couramment utilisée pour trouver les 1 ou 2 paramètres de réglage d'un algorithme d'apprentissage, repose sur l'estimation du risque moyen $\mathbb{E}R(\hat{g})$ d'un algorithme par

$$\tfrac{1}{n}\sum_{j=1}^{p}\sum_{(x,y)\in B_j}\ell\big[y,\hat{g}(\cup_{k\neq j}B_k)(x)\big] \tag{I.5}$$

  où p est l'ordre de la validation croisée et B_1,...,B_p est une partition équilibrée (i.e. $n/p-1 < |B_j| < n/p+1$) de l'ensemble d'apprentissage. Moins formellement, il faut couper l'ensemble d'apprentissage en p parties, entraîner l'algorithme sur $p-1$ de ces parties, regarder la perte qu'encourt cet algorithme sur les données de la p-ème partie, et faire cela p fois (correspondant aux p parties pouvant être laissées de côté lors de l'apprentissage). Pour $p = n$, l'estimateur du risque est appelé *erreur ¡¡laisser-un-de-côté¿¿*.
  Pour les k-p.p.v., chaque k définit un algorithme. Choisir k par validation croisée d'ordre p signifie choisir le k qui minimise l'erreur de validation croisée donnée par (I.5). En pratique, on prend k d'ordre 5 à 10 suivant les contraintes de temps de calcul.

Algorithme par noyau

Le théorème suivant donne un résultat de consistance universelle pour l'algorithme par noyau.

Théorème I.8. *On note par $\mathcal{B}(0,u)$ la boule euclidienne de $\mathbb{R}^d$ de centre 0 et de rayon $u>0$. Si il existe $0<r\leq R$ et $b>0$ tels que*

$$\forall u\in\mathbb{R}^d \quad b\mathbf{1}_{\mathcal{B}(0,r)}\leq K(u)\leq \mathbf{1}_{\mathcal{B}(0,R)}$$

et si $h_n \underset{n\to+\infty}{\longrightarrow} 0$ et $nh_n^d \underset{n\to+\infty}{\longrightarrow} +\infty$, alors l'algorithme par noyau défini par

$$\hat{g}(x)=\sum_{i=1}^n \left(\frac{K(\frac{x-X_i}{h})}{\sum_{l=1}^n K(\frac{x-X_l}{h})}\right)Y_i$$

(avec la convention $\frac{0}{0}=0$) est universellement consistant.

Remarque I.9. C'est un algorithme très employé par les praticiens, notamment avec le noyau gaussien. Il est néanmoins à utiliser avec précaution pour des dimensions de l'espace d'entrées supérieures à 10. ◊

Algorithme par partition

Tout d'abord, rappelons que cet l'algorithme repose sur une partition de $\mathbb{R}^d$ $A_1, A_2, \ldots$ finie ou dénombrable et que pour tout $x\in\mathbb{R}^d$, nous notons $A(x)$ l'élément de la partition qui contient le point x. En pratique, cette partition est prise d'autant plus fine que la taille n de l'nesemble d'apprentissage est grande. Le théorème suivant indique une bonne manière de choisir la partition en fonction de n.

Théorème I.10. *On note encore par $\mathcal{B}(0,u)$ la boule euclidienne de $\mathbb{R}^d$ de centre 0 et de rayon $u>0$. Le diamètre de A_j est noté $\mathrm{Diam}(A_j)=\sup_{x_1,x_2\in A_j}\|x_1-x_2\|$. Si pour tout $R>0$*

$$\begin{cases} \max\limits_{j:A_j\cap\mathcal{B}(0,R)\neq\emptyset} \mathrm{Diam}(A_j) \underset{n\to+\infty}{\longrightarrow} 0 \\ \frac{|\{j:A_j\cap\mathcal{B}(0,R)\neq\emptyset\}|}{n} \underset{n\to+\infty}{\longrightarrow} 0 \end{cases}$$

alors l'algorithme par partition définie par pour tout $x\in\mathbb{R}^d$

$$\hat{g}(x)=\sum_{i=1}^n \left(\frac{\mathbf{1}_{X_i\in A(x)}}{\sum_{l=1}^n \mathbf{1}_{X_l\in A(x)}}\right)Y_i$$

(avec la convention $\frac{0}{0}=0$) est universellement consistant.

Remarque I.11. Pour une grille (régulière) de $\mathbb{R}^d$ de pas H_n, les hypothèses du théorème sont simplement $H_n \underset{n\to+\infty}{\longrightarrow} 0$ et $nH_n^d \underset{n\to+\infty}{\longrightarrow} +\infty$. En pratique, contrairement à l'algorithme par noyau gaussien, il y a un effet escalier (bien souvent non désiré) au bord des éléments de la partition, puisque la sortie prédite pour une entrée x change brusquement lorsque x change d'élément de la partition. ◊

Remarque I.12. Dans les arbres de décision, les éléments de la partition sont définis par les réponses à un ensemble de questions, ces questions étant éventuellement choisies en fonction de l'ensemble d'apprentissage observé. Chaque noeud d'un arbre de décision correspond à un ensemble de l'espace $\mathbb{R}^d$ des entrées. Les contraintes sur les parties de $\mathbb{R}^d$ associées aux noeuds sont les suivantes :

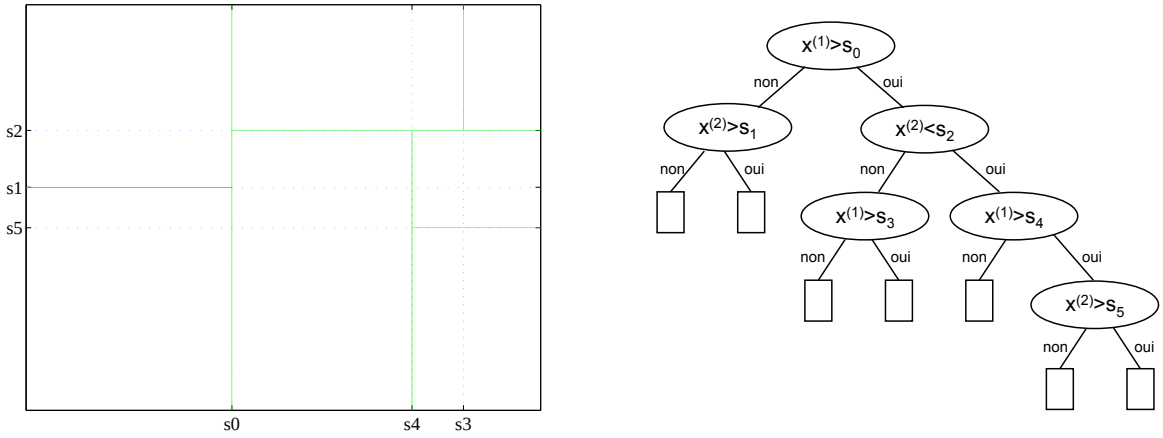

FIG. I.3 – *A gauche :* Exemple de partition provenant d'un arbre de décision. *A droite :* Arbre de décision correspond à cette partition.

- la racine est associée à $\mathbb{R}^d$
- si $A \subset \mathbb{R}^d$ est l'ensemble associé à un noeud et si $A^{(1)}, \ldots, A^{(k)}$ sont les parties associées aux fils de ce noeud, alors

$$A^{(1)} \cup \cdots \cup A^{(k)} = A \qquad \text{et} \qquad \forall 1 \leq i < j \leq k \quad A^{(i)} \cap A^{(j)} = \emptyset,$$

autrement dit $A^{(1)}, \ldots, A^{(k)}$ est une partition de A.

Dans un arbre de décision binaire, tout noeud possède zéro ou deux fils. Donc tout noeud peut être vu comme une question sur l'entrée x dont la réponse est ¡¡oui¿¿ ou ¡¡non¿¿. Les questions typiques sur x sont : la i-ème composante de x est elle plus grande qu'un certain seuil (cf. figure I.3) ? De quel côté x est-il par rapport à un certain hyperplan de $\mathbb{R}^d$? x appartient-il à un certain hyperrectangle ?

L'arbre de décision est une variante de l'algorithme par partition qui est très utilisée notamment en raison de sa simplicité d'interprétation, de la rapidité de l'algorithme et de sa capacité à être mis à jour de manière dynamique (les branches de l'arbre peuvent être développées au fur et à mesure que de nouvelles données viennent compléter l'ensemble d'apprentissage). ◊

I.3.2 Algorithmes par minimisation du risque empirique

Principe de la minimisation du risque empirique

Rappelons tout d'abord que le risque d'une fonction de prédiction $g : \mathcal{X} \to \mathcal{Y}$ est défini par

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

Le but d'un algorithme d'apprentissage est de trouver une fonction de prédiction dont le risque est aussi faible que possible (autrement dit aussi proche que possible du risque des fonctions cibles).

La distribution $\mathbb{P}$ générant les données étant inconnue, le risque R et les fonctions cibles sont inconnus. Néanmoins, le risque $R(g)$ peut être estimé par son équivalent empirique

$$r(g) = \frac{1}{n} \sum_{i=1}^{n} \ell[Y_i, g(X_i)].$$

Si nous supposons $\mathbb{E}\big\{\ell[Y,g(X)]\big\}^2 < +\infty$, alors la L.F.G.N. et le T.C.L. permettent d'affirmer

$$r(g) \xrightarrow[n\to+\infty]{\text{p.s.}} R(g)$$
$$\sqrt{n}\big[r(g) - R(g)\big] \xrightarrow[n\to+\infty]{\mathcal{L}} \mathcal{N}\big(0, \operatorname{Var}\ell[Y,g(X)]\big).$$

Pour toute fonction de prédiction g, la variable aléatoire $r(g)$ effectue donc des déviations en $\mathrm{O}(1/\sqrt{n})$ autour de sa moyenne $R(g)$.

Puisque nous cherchons une fonction qui minimise le risque R et puisque ce risque est approché par le risque empirique r, il est naturel de considérer l'algorithme d'apprentissage, dit de *minimisation du risque empirique*, défini par

$$\hat{g}_{\text{MRE}} \in \operatorname*{argmin}_{g\in\mathcal{G}} r(g), \tag{I.6}$$

où $\mathcal{G}$ est un sous-ensemble de $\mathcal{F}(\mathcal{X},\mathcal{Y})$.

Prendre $\mathcal{G} = \mathcal{F}(\mathcal{X},\mathcal{Y})$ n'est pas une bonne idée. Tout d'abord, cela entraîne un problème de choix puisqu'en général, pour tout ensemble d'apprentissage, il existe une infinité de fonctions de prédiction minimisant le risque empirique (voir figure I.4). Par ailleurs et surtout, si on prend l'algorithme du plus proche voisin comme minimiseur du risque empirique (en régression aux moindres carrés ou en classement), alors on peut montrer que cet algorithme est ¡¡loin¿¿ d'être universellement consistant.

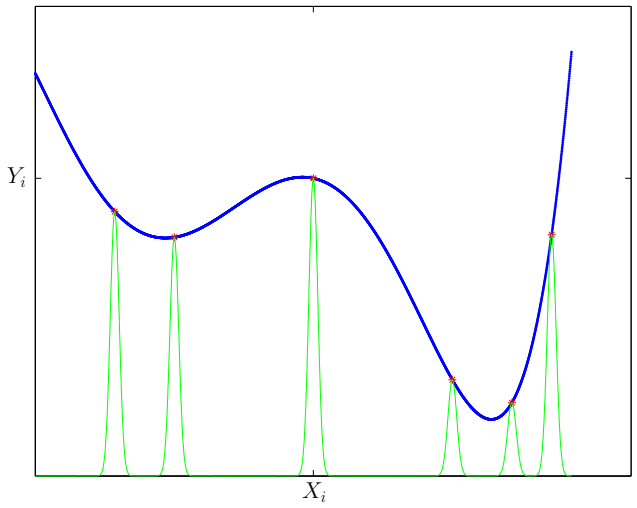

FIG. I.4 – Surapprentissage en régression aux moindres carrés : les couples entrées-sorties sont représentés par des croix. Les deux courbes minimisent le risque empirique $\frac{1}{n}\sum_{i=1}^{n}[Y_i - g(X_i)]^2$ (puisqu'elles ont toutes les deux un risque empirique nulle). La courbe fine semble apprendre par coeur la valeur des sorties associées aux entrées de l'ensemble d'apprentissage. On dit qu'elle "surapprend". Au contraire, la courbe épaisse explique plus simplement les données.

Prendre $\mathcal{G} = \mathcal{F}(\mathcal{X},\mathcal{Y})$ mène en général à un *surapprentissage* dans la mesure où l'algorithme résultant a un risque empirique qui peut être très inférieure à son risque réel (même lorsque la taille de l'ensemble d'apprentissage tend vers l'infini).

En pratique, il faut prendre $\mathcal{G}$ suffisamment grand pour pouvoir raisonnablement approcher toute fonction tout en ne le prenant pas trop grand pour éviter que l'algorithme ¡¡surapprenne¿¿. La ¡¡grandeur¿¿ de l'ensemble $\mathcal{G}$ est appelée *capacité* ou *complexité*. Un autre point de vue consiste à rajouter à $r(g)$ une pénalisation, quand par exemple, la fonction g est trop irrégulière. Ces deux approches sont en fait proches l'une de l'autre. L'approche par pénalisation sera adoptée pour les S.V.M. (Section I.3.2).

Soit $\tilde{g}$ une fonction minimisant le risque sur $\mathcal{G}$:

$$\tilde{g} \in \operatorname*{argmin}_{g \in \mathcal{G}} R(g). \tag{I.7}$$

On suppose le minimum atteint pour simplifier l'exposé. D'après l'inégalité

$$R(\hat{g}_{\mathrm{MRE}}) \geq R(\tilde{g}) \geq R(g^*),$$

L'excès de risque de $\hat{g}_{\mathrm{MRE}}$ se décompose en deux termes positifs, appelés *erreur d'estimation* et *erreur d'approximation* (ou *biais*) :

$$R(\hat{g}_{\mathrm{MRE}}) - R(g^*) = \underbrace{R(\hat{g}_{\mathrm{MRE}}) - R(\tilde{g})}_{\text{erreur d'estimation}} + \underbrace{R(\tilde{g}) - R(g^*)}_{\text{erreur d'approximation}} ,$$

Plus $\mathcal{G}$ est grand, plus l'erreur d'approximation est faible mais plus l'erreur d'estimation est en général grande. Il y a donc un compromis à trouver dans le choix de $\mathcal{G}$. Ce compromis est souvent appelé *dilemme ¡¡biais-variance¿¿* , où le terme variance provient du lien entre l'erreur d'estimation et la variabilité de l'ensemble d'apprentissage que nous avons supposé dans notre formalisme être une réalisation de variables aléatoires i.i.d..

Réseaux de neurones

Les réseaux de neurones sont nés de la volonté de modéliser le fonctionnement du cerveau. Un neurone biologique reçoit des stimuli de ses voisins, et produit un signal lorsque son seuil d'activation est dépassé. La modélisation de Mc Culloch et Pitts (1943) considère que le neurone fait une combinaison linéaire de ses entrées, d'où la fonction d'activation

$$g(x) = \mathbf{1}_{\sum_{j=1}^{d} a_j x^{(j)} + a_0 > 0},$$

où les x_j sont les stimuli envoyés par les neurones voisins et $-a_0$ est le seuil d'activation.

Définition I.13. *Une sigmoïde σ est une fonction croissante telle que*

$$\begin{cases} \sigma(x) & \underset{x \to -\infty}{\longrightarrow} & 0 \\ \sigma(x) & \underset{x \to +\infty}{\longrightarrow} & 1 \end{cases}$$

Voici des exemples de sigmoïdes.

1. $\sigma(x) = \mathbf{1}_{x \geq 0}$
2. $\sigma(x) = \frac{1}{1+\exp(-x)}$
3. $\sigma(x) = \frac{1}{2} + \frac{1}{\pi} \arctan x$

Définition I.14. – *Un* neurone *(artificiel) est une fonction définie sur* $\mathbb{R}^d$ *par*

$$g(x) = \sigma\big(\textstyle\sum_{j=1}^{d} a_j x^{(j)} + a_0\big) = \sigma(a \cdot \tilde{x})$$

où $a = (a_0, \cdots, a_d)^t$, $\tilde{x} = (1, x^{(1)}, \cdots, x^{(d)})^t$ *et* σ *est une sigmoïde.*

– *Un* réseau de neurones à une couche cachée *est une fonction* $f : \mathbb{R}^d \to \mathbb{R}$ *définie par*

$$f(x) = \textstyle\sum_{j=1}^{k} c_j \sigma(a_j \cdot \tilde{x}) + c_0$$

où $\tilde{x} = (1, x^{(1)}, \cdots, x^{(d)})^t$. *La sigmoïde* σ, *le nombre de neurones* k *et les paramètres* $a_1, \ldots, a_k \in \mathbb{R}^{d+1}$, $c_0, c_1, \ldots, c_k \in \mathbb{R}$ *caractérisent le réseau.*

Le théorème suivant donne un résultat de consistance universelle pour l'algorithme de minimisation du risque empirique sur un ensemble de réseaux de neurones à une couche cachée.

Théorème I.15. *Soient* (k_n) *une suite d'entiers et* (β_n) *une suite de réels. Soit* $\mathcal{F}_n$ *l'ensemble des réseaux de neurones à une couche cachée tels que* $k \leq k_n$ *et* $\sum_{i=0}^{k} |c_i| \leq \beta_n$. *L'algorithme* $\hat{f}$ *qui produit pour l'ensemble d'apprentissage* Z_1^n *la fonction de prédiction de* $\mathcal{F}_n$ *minimisant l'erreur quadratique empirique, i.e.*

$$\hat{f} \in \operatorname*{argmin}_{f \in \mathcal{F}_n} \textstyle\sum_{i=1}^{n} [Y_i - f(X_i)]^2,$$

est universellement consistant si $k_n \to +\infty$, $\beta_n \to +\infty$ *et* $\frac{k_n \beta_n^4 \log(k_n \beta_n^2)}{n} \underset{n \to +\infty}{\longrightarrow} 0$.

La minimisation de l'erreur quadratique empirique est un problème d'optimisation non convexe en raison de la sigmoïde. Par conséquent, il n'existe pas en général d'algorithme permettant d'obtenir systématique le minimum global. Il faut donc avoir recours à des heuristiques pour trouver les meilleurs minima locaux (et éventuellement) un minimum global.

L'algorithme des réseaux de neurones est une méthode d'apprentissage puissante ayant notamment donné d'excellents résultats sur les problèmes de reconnaissance de visages ([3]) et de reconnaissance de chiffres manuscrits ([9]). Des conseils sur la manière d'implémenter cet algorithme sont donnés dans [6, 5, 9].

Machines à Vecteurs Supports ou Séparateurs à Vastes Marges (S.V.M.)

Pour simplifier l'exposé, nous présentons cet algorithme dans le cadre du classement binaire avec $\mathcal{X} = \mathbb{R}^d$, $\mathcal{Y} = \{-1; +1\}$ et $\ell(y, y') = \mathbf{1}_{y \neq y'}$. Une fonction cible est alors

$$g^* : x \mapsto \begin{cases} +1 & \text{si } P(Y = +1 | X = x) \geq 1/2 \\ -1 & \text{sinon} \end{cases}$$

Soit $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ une fonction symétrique. Soit $\mathcal{H}$ l'espace vectoriel engendré par les fonctions

$$k(x, \cdot) : x' \mapsto k(x, x').$$

On suppose que

$$\langle \textstyle\sum_{1 \leq j \leq J} \alpha_j k(x_j, \cdot), \sum_{1 \leq k \leq K} \alpha'_k k(x'_k, \cdot) \rangle_{\mathcal{H}} = \sum_{j,k} \alpha_j \alpha'_k k(x_j, x'_k) \tag{I.8}$$

définit un produit scalaire sur $\mathcal{H}$, où $J, K \in \mathbb{N}$, α, α' sont des vecteurs quelconques de $\mathbb{R}^J$, et $x_1, \ldots, x_J$ et $x'_1, \ldots, x'_K$ sont respectivement des J-uplet et K-uplet d'éléments de $\mathcal{X}$.

C'est en particulier le cas pour les trois exemples suivants

- le noyau linéaire $k(x, x') = \langle x, x' \rangle_{\mathbb{R}^d}$,
- les noyaux polynômiaux $k(x, x') = \big(1 + \langle x, x' \rangle_{\mathbb{R}^d}\big)^p$ pour $p \in \mathbb{N}^*$,
- les noyaux gaussiens $k(x, x') = e^{-\|x-x'\|^2/(2\sigma^2)}$ pour $\sigma > 0$.

Plus généralement, (I.8) définit un produit scalaire dès que k est semi-définie positive (i.e. pour tout $J \in \mathbb{N}$, $\alpha \in \mathbb{R}^J$ et tout J-uplet $x_1, \ldots, x_J$ de $\mathcal{X}$, $\sum_{1 \le j,k \le J} \alpha_j \alpha_k k(x_j, x_k) \ge 0$).

L'espace $\mathcal{H}$ muni du produit scalaire $\langle\ ,\ \rangle_{\mathcal{H}}$ est dit à *noyau reproduisant* car pour tout $f \in \mathcal{H}$, nous avons $\langle f, k(x, \cdot) \rangle_{\mathcal{H}} = f(x)$. La fonction k est appelée fonction *noyau*.

Définition I.16. *Soit "sgn" la fonction signe : $sgn(x) = \mathbf{1}_{x \ge 0} - \mathbf{1}_{x<0}$. Notons u_+ la partie positive d'un réel u : $u_+ = \max(u, 0)$. Une machine à vecteur support de noyau k et de paramètre $C > 0$ est un algorithme d'apprentissage qui produit la fonction de prédiction $x \mapsto sgn\big\{\hat{h}(x) + \hat{b}\big\}$, où $(\hat{h}, \hat{b})$ est une solution du problème*

$$\min_{b \in \mathbb{R}, h \in \mathcal{H}} \ C \sum_{i=1}^n \big(1 - Y_i[h(X_i) + b]\big)_+ + \tfrac{1}{2}\|h\|^2_{\mathcal{H}} \qquad (\mathcal{P})$$

Si on impose $b = 0$ (et on ne minimise donc que sur $\mathcal{H}$), on parle de S.V.M. sans terme constant.

Remarque I.17. La machine à vecteur support de noyau linéaire $k(x, x') = \langle x, x' \rangle$ et de paramètre $C > 0$ est un algorithme d'apprentissage qui produit la fonction de prédiction $x \mapsto \mathrm{sgn}\big\{\langle w, x \rangle_{\mathbb{R}^d} + b\big\}$, où $w \in \mathbb{R}^d$ et $b \in \mathbb{R}$ minimisent

$$C \sum_{i=1}^n \big(1 - Y_i[\langle w, X_i \rangle_{\mathbb{R}^d} + b]\big)_+ + \tfrac{1}{2}\|w\|^2_{\mathbb{R}^d}.$$

$\Diamond$

Le théorème suivant donne un résultat de consistance universelle pour les S.V.M. à noyau gaussien.

Théorème I.18. *Soit $\mathcal{X}$ un compact de $\mathbb{R}^d$ et soit $\sigma > 0$. La S.V.M. (avec ou sans terme constant) de noyau gaussien $k : (x, x') \mapsto e^{-\|x-x'\|^2/(2\sigma^2)}$ et de paramètre $C = n^{\beta-1}$ avec $0 < \beta < 1/d$ est universellement consistante.*

Concentrons-nous à présent sur la résolution du problème d'optimisation $(\mathcal{P})$ et sur l'interprétation de la solution proposée. Des techniques d'optimisation classiques montrent que le problème d'optimisation $(\mathcal{P})$ sur un e.v. de dimension infinie se ramène (par passage à son problème dual) à un problème d'optimisation sur un e.v. de dimension n qui est quadratique et avec des contraintes linéaires simples.

Le théorème suivant explique comment calculer la solution de $(\mathcal{P})$ en pratique.

Théorème I.19. *Il existe (h, b) solution de $(\mathcal{P})$. De plus, h est unique et s'écrit*

$$h = \sum_{j=1}^n \alpha_j Y_j k(X_j, \cdot), \qquad (I.9)$$

où α est un vecteur de $\mathbb{R}^n$ qui maximise

$$\sum_{i=1}^n \alpha_i - \tfrac{1}{2} \sum_{1 \le i,j \le n} \alpha_i \alpha_j Y_i Y_j k(X_i, X_j) \qquad (\mathcal{D})$$

sous les contraintes : $\sum_{i=1}^n \alpha_i Y_i = 0$ *et* $\forall i, 0 \leq \alpha_i \leq C$*. Par ailleurs, si il existe une composante de* α *telle que* $0 < \alpha_i < C$*, alors* $Y_i - h(X_i)$ *est constant sur l'ensemble des* i *tel que* $0 < \alpha_i < C$*. Soit* b *la valeur commune. Le couple* (h, b) *est solution de* $(\mathcal{P})$*.*

Idées de la preuve. Pour l'unicité du h solution de $(\mathcal{P})$, il faut montrer tout d'abord que h appartient nécessairement l'espace vectoriel engendrée par $k(X_1, \cdot), \ldots, k(X_n, \cdot)$, puis utiliser la stricte convexité de $h \mapsto \|h\|_{\mathcal{H}}^2$. (Attention, l'unicité de h n'implique pas en général celle des α_i.)

Pour obtenir $(\mathcal{D})$, la première étape consiste à remarquer que $(\mathcal{P})$ est égal à

$$\min_{\substack{b\in\mathbb{R}, h\in\mathcal{H} \\ \xi_i\geq 0 \\ \xi_i \geq 1 - Y_i[h(X_i)+b]}} \quad C\sum_{i=1}^n \xi_i + \tfrac{1}{2}\|h\|_{\mathcal{H}}^2,$$

ce qui ramène le problème convexe $(\mathcal{P})$ à un problème quadratique avec contraintes linéaires. La suite de la preuve consiste à écrire le dual de ce problème et à utiliser les conditions de Karush-Kuhn-Tucker. □

Remarque I.20. La fonction de prédiction produite par une S.V.M. de noyau k peut donc s'écrire $x \mapsto \operatorname{sgn}\big\{\sum_{i=1}^n \alpha_i' k(X_i, x) + b\big\}$ où $\alpha_1', \ldots, \alpha_n'$ et b sont des paramètres judicieusement choisis. Une S.V.M. de noyau k réalise donc une (éventuellement partielle) séparation linéaire des données $X_1, \ldots, X_n$ représentées respectivement par $k(X_1, \cdot), \ldots, k(X_n, \cdot)$ dans l'espace vectoriel $\mathcal{H}$. Les α_i utilisés dans (I.9) et obtenus par résolution de $(\mathcal{D})$ sont intrinsèquement liés à la position de $k(X_i, \cdot)$ par rapport à l'hyperplan $h(x) + b = 0$ (voir figure I.5). Connaître α_i et Y_i permet de situer $k(X_i, \cdot)$ par rapport à la frontière de décision (d'équation $h(x) + b = 0$) sans avoir besoin de calculer $h(X_i) + b$. Cela permet une interprétation rapide des résultats d'une S.V.M.. Notons en particulier que

1. tout point mal classé X_i vérifie nécessairement $\alpha_i = C$ (réciproque fausse),
2. tout point X_i tel que $0 < \alpha_i < C$ se trouve sur les courbes d'équations respectives $h(x) + b = -1$ et $h(x) + b = +1$,
3. tout point X_i tel que $\alpha_i = 0$ n'influe pas sur la fonction de prédiction produite par une S.V.M. au sens où en retirant ces points de l'ensemble d'apprentissage et en recalculant la solution de $(\mathcal{D})$ pour ce nouvel ensemble d'apprentissage, nous retrouvons la même fonction de prédiction : $x \mapsto \operatorname{sgn}[h(x) + b]$.

◊

Dans le contexte du classement binaire, l'algorithme des réseaux de neurones décrits dans le Théorème I.15 peut être également utilisé (par exemple en seuillant à 0 la fonction obtenue par minimisation du risque empirique quadratique : $\hat{g}(x) = +1$ si $\hat{f}(x) \geq 0$ et $\hat{g}(x) = -1$ sinon). L'inconvénient de cet algorithme d'apprentissage est qu'il doit résoudre un problème d'optimisation non convexe. Rien n'empêche en pratique l'algorithme de minimisation de tomber dans un minimum local. En conséquence, suivant l'implémentation et les heuristiques utilisées pour minimiser le risque empirique, des résultats très inégaux peuvent être observés.

L'algorithme des S.V.M. est en général privilégié par les praticiens car il possède les mêmes propriétés théoriques que l'algorithme des réseaux de neurones sans en avoir les inconvénients pratiques : les S.V.M. résolvent un problème d'optimisation convexe dont la solution est le

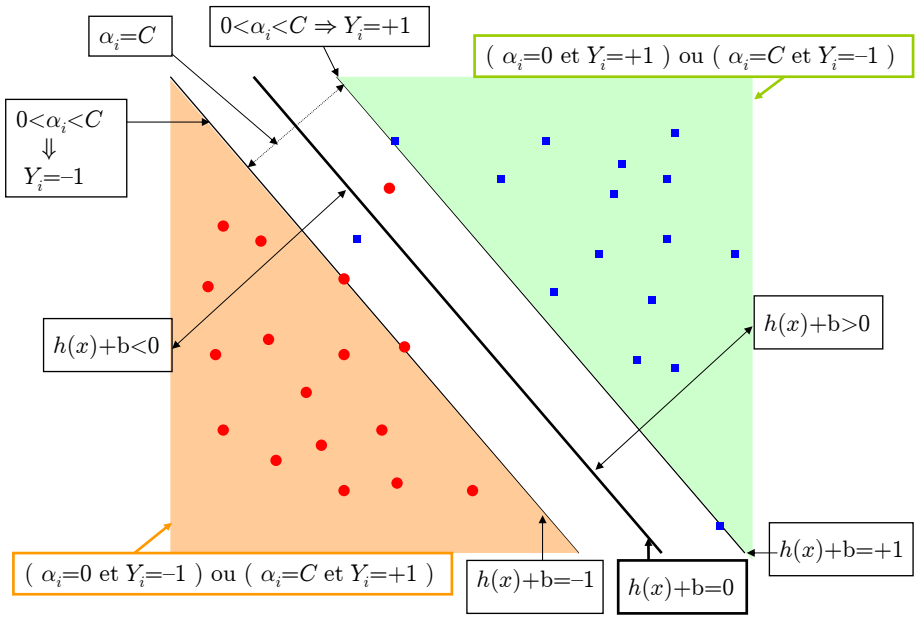

FIG. I.5 – Interprétation géométrique des S.V.M.. Le plan du dessin représente l'espace vectoriel (de fonctions de $\mathcal{X}$ dans $\mathbb{R}$) engendré par les fonctions $k(X_1,\cdot),\ldots,k(X_n,\cdot)$. Les coefficients α_i sont ceux obtenues par résolution de $(\mathcal{D})$ et le couple (h,b) est la solution présentée dans le Théorème I.19 : $h=\sum_{j=1}^{n}\alpha_j Y_j k(X_j,\cdot)$. Les points ronds et carrés représentent les points de l'ensemble d'apprentissage (par le biais des fonctions $k(X_i,\cdot)$ qui leur sont associées).

plus souvent unique et facile à obtenir. De nombreux logiciels implémentant les S.V.M. dans différents langages informatiques sont accessibles en ligne.

Choix du paramètre C et du noyau k. Pour $\mathcal{X}=\mathbb{R}^d$, les noyaux les plus populaires sont

- le noyau linéaire $(x,x')\mapsto\langle x,x'\rangle_{\mathbb{R}^d}$,
- les noyaux polynômiaux $(x,x')\mapsto\left(1+\langle x,x'\rangle_{\mathbb{R}^d}\right)^p$ pour $p\in\mathbb{N}^*$,
- les noyaux gaussiens $(x,x')\mapsto e^{-\|x-x'\|^2/(2\sigma^2)}$ pour $\sigma>0$.

Attention, seul le noyau gaussien peut mener à un algorithme universellement consistant. Il doit donc être en général préféré en pratique.

Pour choisir C, la méthode la plus employée est de faire une validation croisée. Pour limiter le temps de calcul, il faut prendre tout d'abord une grille géométrique grossière (par exemple, $\{10^5/n,\ 10^4/n,\ 10^3/n,\ 10^2/n,\ 10/n\}$), puis prendre une grille géométrique plus fine autour du C ayant donné la plus faible erreur de validation croisée (cf. (I.5)), à condition que ce C ne soit pas sur les bords de la grille grossière.

Pour choisir la largeur σ du noyau gaussien, le principe est le même que pour choisir C sauf que l'intervalle de recherche est $[\sigma_{\min};\sigma_{\max}]$, où $\sigma_{\min}$ est la médiane de l'ensemble des distances d'un point de l'ensemble d'apprentissage à son plus proche voisin et $\sigma_{\max}$ est le diamètre de $\{X_1,\ldots,X_n\}$ pour la norme euclidienne sur $\mathbb{R}^d$, i.e. $\sigma_{\max}=\max_{i,j}\|X_i-X_j\|_{\mathbb{R}^d}$.

I.4 Au delà de la consistance universelle

Les résultats de consistance universelle sont des résultats asymptotiques. Ils disent juste que si nous avons suffisamment de données (et un ordinateur suffisamment puissant pour les traiter) alors tout algorithme universellement consistant produira une prédiction très proche de la meilleure prédiction possible.

Les résultats de consistance universelle ne disent pas le nombre de données nécessaires pour avoir une garantie du type $\mathbb{E}R(\hat{g}) \leq R(g^*) + \epsilon$ pour $\epsilon > 0$ fixé. Pour que ce nombre existe, il faudrait avoir un résultat de consistance universelle uniforme, i.e.

$$\lim_{n\to+\infty} \sup_{\mathbb{P}} \left\{\mathbb{E}R(\hat{g}) - R(g^*)\right\} = 0,$$

la consistance universelle n'affirmant que

$$\sup_{\mathbb{P}} \lim_{n\to+\infty} \left\{\mathbb{E}R(\hat{g}) - R(g^*)\right\} = 0.$$

En général, ce nombre n'existe pas d'après le théorème suivant.

Théorème I.21. *Si $|\mathcal{X}| = +\infty$, il n'existe pas d'algorithme d'apprentissage uniformément universellement consistant ni en régression aux moindres carrés ($\ell(y, y') = (y - y')^2$) ni en classement ($\ell(y, y') = \mathbf{1}_{y\neq y'}$).*

L'absence d'algorithme universellement uniformément consistant nous amène à définir un ¡¡bon¿¿ algorithme d'apprentissage comme étant un algorithme universellement consistant et ayant une propriété de convergence uniforme sur une classe de probabilités paraissant pertinente pour le problème à traiter. Plus précisément, si $\mathcal{P}$ est un ensemble de probabilités sur $\mathcal{Z}$ dans laquelle nous pensons que $\mathbb{P}$ est, nous souhaitons que le bon algorithme satisfasse

$$\lim_{n\to+\infty} \sup_{\mathbb{P}\in\mathcal{P}} \left\{\mathbb{E}R(\hat{g}) - R(g^*)\right\} = 0$$

et également avoir une suite $\sup_{\mathbb{P}\in\mathcal{P}} \left\{\mathbb{E}R(\hat{g}) - R(g^*)\right\}$ décroissant le plus vite possible vers 0 pour que peu de données soient nécessaires à l'algorithme pour prédire efficacement. L'ensemble $\mathcal{P}$ doit être pensé comme une modélisation de notre a priori, et il en résulte un a priori implicite sur la fonction cible. L'obtention d'algorithmes incorporant un a priori et étant efficace lorsque l'a priori est correct est au coeur de la recherche actuelle en apprentissage statistique.

Bibliographie

[1] B. Blankertz, K.-R. Mller, G. Curio, T. Vaughan, G. Schalk, J. Wolpaw, A. Schlgl, C. Neuper, G. Pfurtscheller, T. Hinterberger, M. Schrder, and N. Birbaumer. Bci competition 2003 : Progress and perspectives in detection and discrimination of eeg single trials. *IEEE Transactions on Biomedical Engineering*, 51(6) :1044–1051, 2004.

[2] D. DeCoste and B. Schölkopf. Training invariant support vector machines. *Machine Learning*, 46 :161–190, 2002.

[3] C. Garcia and M. Delakis. Convolutional face finder : A neural architecture for fast and robust face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(11) :1408–1423, 2004. article non accessible en ligne, me le demander.

[4] L. Györfi, M. Kohler, A. Krzyżak, and H. Walk. *A Distribution-Free Theory of Nonparametric Regression*. Springer, 2004.

[5] Y. LeCun, 2005. `http://www.cs.nyu.edu/~yann/2005f-G22-2565-001/diglib/lecture09-optim.djvu`, requires the djvu viewer `http://djvu.org/download/`.

[6] Y. LeCun, L. Bottou, G. Orr, and K. Muller. Efficient backprop, `http://yann.lecun.com/exdb/publis/pdf/lecun-98b.pdf`. In G. Orr and M. K., editors, *Neural Networks : Tricks of the trade*. Springer, 1998.

[7] Y. LeCun and C. Cortes. MNIST page. `http://yann.lecun.com/exdb/mnist/`.

[8] G. Schalk, D. McFarland, T. Hinterberger, N. Birbaumer, and J. Wolpaw. BCI2000 : a general-purpose brain-computer interface system. *IEEE Transactions on Biomedical Engineering*, 51(6) :1034–1043, 2004.

[9] P. Simard, D. Steinkraus, and J. Platt. Best practice for convolutional neural networks applied to visual document analysis, `http://research.microsoft.com/~patrice/PDF/fugu9.pdf`. *International Conference on Document Analysis and Recogntion (ICDAR), IEEE Computer Society*, pages 958–962, 2003.