Table des matières

Première partie

Introduction

Chapitre 1

Présentation du domaine de l'apprentissage automatique

1.1 Qu'est-ce que l'apprentissage automatique

La faculté d'apprendre de ses expériences passées et de s'adapter est une caractéristique essentielle des formes de vies supérieures. Elle est essentielle à l'être humain dans les premières étapes de la vie pour apprendre des choses aussi fondamentales que reconnaître une voix, un visage familier, apprendre à comprendre ce qui est dit, à marcher et à parler.

L'apprentissage automatique[1] est une tentative de comprendre et reproduire cette faculté d'apprentissage dans des systèmes artificiels. Il s'agit, très schématiquement, de concevoir des algorithmes capables, à partir d'un nombre important

[1] Anglais : *Machine Learning*

d'exemples (les *données* correspondant à "l'expérience passée"), d'en assimiler la nature afin de pouvoir appliquer ce qu'ils ont ainsi appris aux cas futurs.

1.2 Situation historique multi-disciplinaire

Traditionnellement, on considère que le domaine de l'apprentissage automatique sub-symbolique est né vers la fin des années 50, comme une branche dissidente de l'Intelligence Artificielle classique, avec la publication des travaux de Rosenblatt sur le Perceptron [75].

Historiquement, c'est là le fruit de la rencontre de l'Intelligence Artificielle et des neuro-sciences. Ce qu'on a alors appelé la branche "connexioniste" de l'I.A. ambitionnait de parvenir à créer des machines capables d'intelligence en tentant de mimiquer le fonctionnement des systèmes nerveux biologiques, ou tout du moins en s'inspirant fortement des connaissances sur les réseaux de neurones biologiques, et présentait un départ radical de l'approche symbolique de logique "Aristotélicienne" adoptée par l'I.A. classique. Ainsi ont été développés les réseaux de neurones artificiels.

Dès sa naissance, le domaine était donc résolument inter-disciplinaire. Au cours des 45 années qui ont suivi, ce caractère n'a fait que s'accentuer, et si l'attrait pour la stricte inspiration biologique s'est beaucoup estompé (certains diront malheureusement), c'est avant tout parce que des connexions profondes ont été développées avec d'autres disciplines. En effet la formalisation du domaine, son mûrissement, la compréhension théorique accrue des problèmes impliqués, se sont accompagnés d'un rapprochement avec des disciplines ayant de solides fondations

mathématiques et théoriques telles que la théorie de l'information et le traitement du signal, l'optimisation non-linéaire, mais surtout et de façon prépondérante ces dernières années avec le point de vue statistique.

1.2.1 L'apprentissage automatique par rapport aux statistiques classiques

Du point de vue du problème de l'apprentissage, on peut diviser les statistiques classiques en deux branches :

– Les statistiques paramétriques, dont le cadre suppose que l'on connaît la forme du *vrai* modèle qui a généré les données, ignorant seulement ses paramètres, et où il s'agit d'estimer au mieux les paramètres du dit modèle à partir d'un échantillon de données fini.
– Les statistiques non paramétriques (k plus proches voisins, fenêtres de Parzen, . . . Voir la section 5.2). Là, la plupart des études statistiques s'intéressent aux propriétés de convergence et consistance de l'estimateur quand le nombre d'exemples tend vers l'infini.

Les recherches en apprentissage automatique se sont quant à elles concentrées davantage sur des problèmes réels complexes, où il serait absurde de croire que l'on puisse disposer du *vrai* modèle, et où l'on est également loin d'avoir une quantité illimitée de données. Bien que les statistiques classiques se soient un peu intéressées à ces questions, depuis l'avènement de l'informatique ce champ d'investigation a surtout été exploré par la communauté de l'apprentissage automatique. Par ses origines dans des domaines moins frappés de rigueur et de formalisme mathématique (la neuro-biologie et l'électronique/informatique), les recherches en

intelligence artificielle sub-symbolique ont pris un chemin davantage empirique, se satisfaisant très bien de produire des “monstres” mathématiques comme les réseaux de neurones, du moment qu’ils fonctionnaient et donnaient de bons résultats ! Dans la mesure où les modèles utilisés étaient plus complexes, les questions de sélection de modèle et du contrôle de leur capacité se sont imposées naturellement avec force.

Mais on voit que, bien plus qu’une différence de fond entre les deux domaines, ce qui les sépare est une différence de culture et d’emphase : les études statistiques classiques se sont souvent auto-limitées à des modèles se prêtant bien à une analyse mathématique (modèles assez simples, en faible dimension). En comparaison, la recherche en intelligence artificielle était résolument engagée sur la voie de la complexité, avec pour seule limite la capacité du matériel informatique, et poussée par le besoin de mettre au point des systèmes répondant aux problèmes concrets du moment.

Néanmoins, avec le temps, le domaine de l’apprentissage automatique a mûri, s’est formalisé, théorisé, et s’est ainsi inéluctablement rapproché des statistiques, au point d’être rebaptisé *apprentissage statistique*. Pour autant, bien que s’étant considérablement réduit, le fossé culturel n’est pas totalement comblé, notamment en ce qui concerne les conventions quant aux façons de procéder et à la terminologie. Nous espérons donc que le lecteur davantage familier avec le formalisme statistique saura pardonner l’approche certainement moins rigoureuse de l’auteur.

1.3 Les tâches de l'apprentissage

On peut séparer les *tâches* de l'apprentissage automatique en trois grandes familles :

– apprentissage supervisé,
– apprentissage non-supervisé,
– apprentissage par renforcement.

1.3.1 L'apprentissage supervisé

La formulation du problème de l'apprentissage supervisé est simple : on dispose d'un nombre fini d'exemples d'une tâche à réaliser, sous forme de paires *(entrée, sortie désirée)*, et on souhaite obtenir, d'une manière automatique, un système capable de trouver de façon relativement fiable la sortie correspondant à toute nouvelle entrée qui pourrait lui être présentée.

On distingue en général trois types de problèmes auxquels l'apprentissage supervisé est appliqué. Ces tâches diffèrent essentiellement par la nature des paires *(entrée, sortie)* qui y sont associées :

Classification

Dans les problèmes de classification, l'entrée correspond à une instance d'une classe, et la sortie qui y est associée indique la classe. Par exemple pour un problème de reconnaissance de visage, l'entrée serait l'image bitmap d'une personne

telle que fournie par une caméra, et la sortie indiquerait de quelle personne il s'agit (parmi l'ensemble de personnes que l'on souhaite voir le système reconnaître).

Régression

Dans les problèmes de régression, l'entrée n'est pas associée à une classe, mais dans le cas général, à une ou plusieurs valeurs réelles (un vecteur). Par exemple, pour une expérience de biochimie, on pourrait vouloir prédire le taux de réaction d'un organisme en fonction des taux de différentes substances qui lui sont administrées.

Séries temporelles

Dans les problèmes de séries temporelles, il s'agit typiquement de prédire les valeurs futures d'une certaine quantité connaissant ses valeurs passées ainsi que d'autres informations. Par exemple le rendement d'une action en bourse... Une différence importante avec les problèmes de régression ou de classification est que les données suivent typiquement une distribution non stationnaire.

1.3.2 L'apprentissage non-supervisé

Dans l'apprentissage *non supervisé* il n'y a pas de notion de sortie désirée, on dispose seulement d'un nombre fini de données d'apprentissage, constituées "d'entrées", sans qu'aucun label n'y soit rattaché.

Estimation de densité

Dans un problème d'estimation de densité, on cherche à modéliser convenablement la distribution des données. L'estimateur obtenu $\hat{f}(x)$ doit pouvoir donner un bon estimé de la densité de probabilité à un point de test x issu de la même distribution (inconnue) que les données d'apprentissage.

Partitionnement

Le problème du partitionnement[2] est le pendant non-supervisé de la classification. Un algorithme de partitionnement tente de partitionner l'espace d'entrée en un certain nombre de "classes" en se basant sur un ensemble d'apprentissage fini, ne contenant aucune information de classe explicite. Les critères utilisés pour décider si deux points devraient appartenir à la même classe ou à des classes différents sont spécifiques à chaque algorithme, mais sont très souvent liés à une mesure de distance entre points.

Réduction de dimensionalité

Le but d'un algorithme de réduction de dimensionalité est de parvenir à "résumer" l'information présente dans les coordonnées d'un point en haute dimension ($x \in \mathbb{R}^n$, n grand) par un nombre plus réduit de caractéristiques ($y = f(x), y \in \mathbb{R}^m, m < n$). Le but espéré est de préserver l'information "importante", de la mettre en évidence en la dissociant du bruit, et possiblement de révéler une structure sous-jacente qui ne serait pas immédiatement apparente dans les données

[2] Anglais : *clustering*

d'origine en haute dimension. L'exemple le plus classique d'algorithme de réduction de dimensionalité est l'Analyse en Composantes Principales (ACP) [52].

1.3.3 L'apprentissage par renforcement

Nous ne faisons ici que mentionner très succinctement le cadre général de l'aprentissage par renforcement, ce domaine étant hors du champ de notre sujet. Nous invitons le lecteur désireux d'en savoir plus à se référer à [90]. La particularité et la difficulté du cadre de l'apprentissage par renforcement est que les décisions prises par l'algorithme influent sur l'environnement et les observations futures. L'exemple typique est celui d'un robot autonome qui évolue et effectue des actions dans un environnement totalement inconnu initialement. Il doit constamment apprendre de ses erreurs et succès passés, et décider de la meilleure politique à appliquer pour choisir sa prochaine action.

1.3.4 Inter-relations entre les techniques

Bien entendu, les frontières entre les tâches que nous venons de présenter sont souples. Ainsi on applique couramment, et avec succès, des algorithmes conçus pour faire de la régression à des problèmes de classification, ou bien on estime des densités dans le but de faire de la classification (voir la section 3.1).

Notez qu'une bonne estimation de densité permet en théorie de prendre la décision optimale concernant un problème de classification ou de régression. Mais d'un autre côté l'estimation de densité est souvent un problème plus difficile, en pratique avec un nombre fini de données d'entraînement.

Précisons que, dans la suite de l'exposé, nous limiterons notre attention aux problèmes de classification, régression, et estimation de densité dans $\mathbb{R}^n$ (la représentation de donnée la plus souvent utilisée). Nous porterons un intérêt tout particulier aux cas où n est grand, de l'ordre de 100 à 1000 : la *haute dimension*.

Chapitre 2

La généralisation : le grand défi de l'apprentissage

2.1 Mémoriser n'est pas généraliser

Le terme *apprentissage* dans la langue courante est ambigu. Il désigne aussi bien l'apprentissage "par coeur" d'une poésie, que l'apprentissage d'une tâche complexe telle que la lecture.

Clarifions la distinction :

– Le premier type d'apprentissage correspond à une simple mémorisation. Or les ordinateurs contemporains, avec leurs mémoires de masse colossales, n'ont aucune difficulté à mémoriser une encyclopédie entière[1], sons et images inclus.

[1]bien que la façon dont ils accèdent à cette mémoire ne soit pas qualitativement très différente de le façon dont un être humain accède à l'encyclopédie de sa bibliothèque - sa mémoire étendue -, juste un peu plus rapide…

– Le second type d'apprentissage se distingue fondamentalement du premier en cela qu'il fait largement appel à notre faculté de généraliser. Ainsi pour apprendre à lire, on doit être capable d'identifier un mot écrit d'une manière que l'on n'a encore jamais vue auparavant.

Bien qu'il soit trivial de mémoriser une grande quantité d'exemples, les généraliser à la résolution de nouveaux cas, même s'ils ne diffèrent que légèrement, est loin d'être un problème évident. Ce que l'on entend par *apprentissage* dans les algorithmes d'apprentissage, et qui en fait tout l'intérêt et toute la difficulté est bel et bien la capacité de généraliser, et non pas simplement celle d'un apprentissage par coeur.

2.2 Notations et formalisation du problème

En général nous utilisons des P majuscules pour identifier des probabilités, et des p minuscules pour les densités de probabilité, f étant réservé pour les fonctions de décision. Mais dans ce qui suit, par un léger abus de notation, nous utiliserons un P majuscule indifféremment pour désigner une probabilité *ou* une densité, en fonction du contexte et de la nature, discrète, catégorique, ou continue des variables. Ainsi avec des variables aléatoires X continue et Y discrète, $P(X = x|Y = y)$ sera une densité conditionnelle, alors que $P(Y = y|X = x)$ sera une probabilité conditionnelle.

Formalisons à présent le problème de l'apprentissage supervisé :[2]

[2]nous rappelons que nous ne considérons pas ici le cas des données temporelles, dont la nature non stationnaire occasionne des complications supplémentaires

On dispose d'un ensemble $\mathcal{D}$ de l données d'apprentissage, sous forme de paires *(entrée, sortie)* : $\mathcal{D} = \{(x_1, y_1), (x_2, y_2), ..., (x_l, y_l)\}$ que l'on suppose tirées de manière $i.i.d.$ d'une distribution inconnue $P(X, Y)$.

Typiquement, entrée et sortie sont représentées sous forme d'un vecteur réel : $X \in \mathbb{R}^n$, $Y \in \mathbb{R}^{n'}$

Le problème de l'apprentissage supervisé est alors, à partir de l'ensemble d'apprentissage (et possiblement de connaissances à priori que l'on possède du domaine), de trouver, pour de nouveaux (x, y) tirés de la même distribution $P(X, Y)$, un moyen de calculer le y associé au x en commettant le moins d'erreurs possible.

L'approche la plus couramment utilisée consiste à tout d'abord utiliser $\mathcal{D}$ pour trouver la "meilleure" fonction $f : x \mapsto y$, $f \in \mathcal{F}$ (où $\mathcal{F}$ est choisi à l'avance). Puis à utiliser la fonction ainsi modélisée pour associer un y à tout nouveau x de test.

On dispose en général d'une fonction de coût $C(y_{\textit{calculé}}, y_{\textit{réel}})$. Par exemple, pour la classification, si y représente le numéro de la classe, on pourrait compter le nombre d'erreurs de classification à l'aide de la fonction de coût définie comme :

$C(y_1, y_2) = 0$ si $y_1 = y_2$

$C(y_1, y_2) = 1$ si $y_1 \neq y_2$

Ou bien pour un problème de régression, on pourra s'intéresser à l'erreur quadratique : $C(y_1, y_2) = \|y_1 - y_2\|^2 = \sum_{j=1}^{n'} (y_{1j} - y_{2j})^2$

Idéalement, on voudrait trouver $f \in \mathcal{F}$ qui minimise l'erreur de généralisation, ou risque espéré : $R(f) = E[C(f(X), Y)] = \int C(f(x), y) P(x, y) dx dy$

Mais comme on ne connaît pas la vraie distribution $P(X, Y)$, on doit se contenter de trouver le f qui minimise un *estimé* de cette erreur de généralisation. Typiquement cet estimé $\tilde{R}(f)$ est construit à partir de deux termes :

– le risque (ou erreur) empirique calculé sur les données d'apprentissage : $R(f, \mathcal{D}) = \frac{1}{l} \sum_{i=1}^{l} C(f(x_i), y_i)$

– et une pénalité $H(f)$ qui induit une préférence sur les solutions f

$\mathcal{F}$ est souvent un ensemble de fonctions paramétré par un vecteur de paramètres θ et rechercher $f_\theta \in \mathcal{F}$ revient dans ce cas à rechercher un $\theta \in \mathbb{R}^p$ qui minimise $\tilde{R}(f_\theta)$

Cette approche qui consiste d'abord à trouver une fonction à partir des données d'entraînement, pour ensuite appliquer cette fonction sur les nouvelles données de test est l'approche *inductive*. Une approche légèrement différente, qui consiste à trouver une fonction dépendant également du ou des *points de test* considérés est dite *transductive.*

La fonction ainsi obtenue constitue un *modèle,* dans le sens qu'elle permet de modéliser la relation entre entrée et sortie. L'optimisation des paramètres se nomme la phase *d'entraînement* ou *d'apprentissage* du modèle, et permet d'obtenir un *modèle entraîné.*

2.3 Mesure de la performance de généralisation

Lorsque l'on construit un modèle afin qu'il minimise le risque empirique calculé sur les données d'apprentissage, l'erreur d'apprentissage ainsi obtenue ne peut être considérée comme une bonne mesure de l'erreur de généralisation : elle est

évidemment biaisée. Pour obtenir un estimé non biaisé de l'erreur de généralisation, il est crucial de mesurer l'erreur sur des exemples qui n'ont pas servi à entraîner le modèle. Pour cela, on divise l'ensemble des données disponibles en deux parties :

– un sous ensemble d'entraînement, dont les données serviront à l'apprentissage (ou entraînement) du modèle ;
– un sous ensemble de test, dont les données seront utilisées uniquement pour évaluer la performance du modèle entraîné. Ce sont les données "hors-échantillon d'entraînement". On obtient ainsi l'*erreur de test* qui est un estimé bruité, mais non biaisé de l'erreur de généralisation.

Par ailleurs l'ensemble d'entraînement est souvent lui-même partagé entre un sous-ensemble qui sert à apprendre les paramètres d'un modèle à proprement dit, et un autre sous-ensemble dit "de validation", qui sert à la sélection de modèle. La *sélection de modèle* est ici comprise au sens large : il peut s'agir de choisir le meilleur entre plusieurs familles de modèles très différents, ou entre des variantes très semblables d'un même modèle, dues uniquement à des variations des valeurs d'un hyper-paramètre contrôlant la définition du modèle.

Dans les cas où l'on dispose de trop peu de données, on peut utiliser une technique de *validation croisée*[89], pour générer plusieurs paires de sous-ensembles entraînement/test. Cette technique est coûteuse en temps de calcul, mais permet d'obtenir un bon estimé de l'erreur de test, tout en conservant suffisamment d'exemples pour l'entraînement.

Lorsque, dans le présent document, nous parlons de *performance* d'un modèle, nous entendons la performance "en test", telle que mesurée sur un ensemble de test ou par validation croisée.

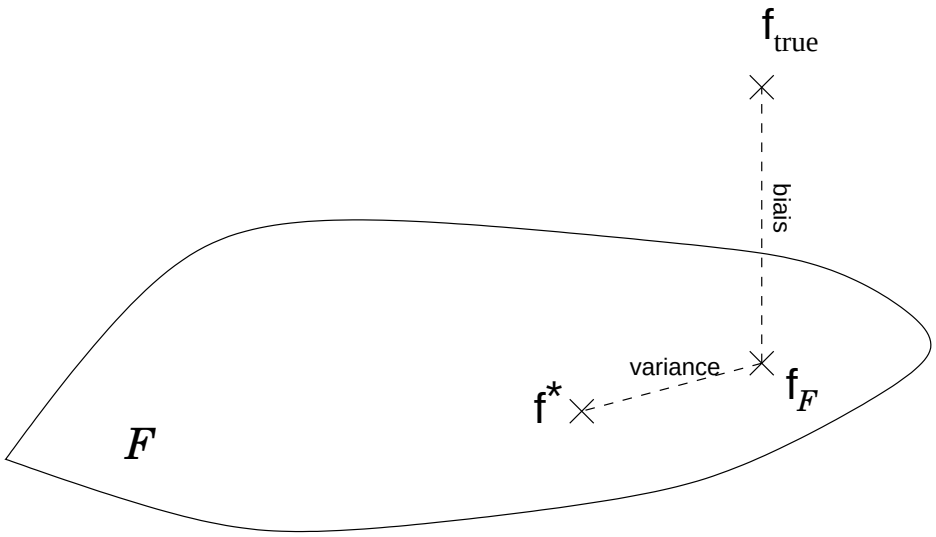

FIG. 2.1 – Biais et Variance

2.4 Quelques notions de théorie d'apprentissage

Le lecteur intéressé par un développement formel plus complet de la théorie de l'apprentissage statistique est invité à se référer à [99]. Nous nous contentons de présenter ici quelques notions.

D'une manière générale, l'approche inductive permet de combiner deux types d'informations pour résoudre le problème particulier (de classification ou de régression) qui nous occupe :

– Des connaissances ou intuitions à priori sur la forme que la solution devrait avoir. Elles se traduisent dans le choix de l'ensemble de fonctions $\mathcal{F}$ dans lequel on va chercher la solution, dans le choix de la fonction de coût C, et dans le choix de la fonction de pénalité $H(f)$ qui permet de spécifier une préférence parmi les fonctions de $\mathcal{F}$.

– Un nombre fini l d'exemples de paires (x, y), possiblement bruités, de valeurs de la "vraie" fonction en un certain nombre de points : notre ensemble d'apprentissage.

Un algorithme d'apprentissage nous permet en principe de trouver dans l'ensemble $\mathcal{F}$ la fonction f^* qui satisfait le mieux ces contraintes. Mais on se heurte typiquement à deux problèmes inconciliables :

– La "vraie fonction" idéale f_{true} ne se trouve peut-être pas dans l'ensemble $\mathcal{F}$ que nous avons choisi. En conséquence de quoi on aurait tendance à choisir un ensemble de fonction *plus vaste* pour limiter ce problème.
– Le nombre limité d'exemples d'entraînement dont nous disposons n'est pas suffisant pour localiser de façon précise la fonction $f^*_{\mathcal{F}}$ qui est la plus proche de f_{true} parmi notre ensemble de fonctions. Ce problème peut logiquement être réduit en choisissant un ensemble de fonctions $\mathcal{F}$ *plus petit*.

Le terme d'erreur dû au premier problème se nomme le *biais*, et celui dû au second se nomme la *variance* (car il est dû à la variabilité de l'échantillon fini qu'est l'ensemble d'apprentissage que l'on nous donne). Et le dilemme que cela occasionne est appelé *dilemme biais-variance*. Voir la figure 2.1.

On voit que la "taille" ou "complexité" de l'ensemble de fonctions $\mathcal{F}$ joue un rôle fondamental. Ce que l'on nomme la *capacité* $h(\mathcal{F})$ est une mesure de cette complexité.

2.5 Pratiques courantes de contrôle de capacité

Un nombre de pratiques couramment employées permet d'exercer un certain contrôle sur la *capacité*. Le principe fondamental est celui de la *régularisation*[94], qui permet d'introduire une préférence sur les fonctions de $\mathcal{F}$. La fonction de pénalité $H(f)$ que nous avons déjà mentionnée, est ajoutée au coût optimisé, et ce terme est appelée *terme de régularisation* : on favorise ainsi les fonctions plus simples (faible capacité) et pénalise davantage les plus complexes (capacité élevée).

Voici quelques méthodes couramment utilisées pour traduire ce principe en pratique. Le lecteur intéressé à la mise en pratique concrète de ces techniques particulièrement pour l'entraînement des réseaux de neurones est invité à se référer à [68].

– "*weight-decay*" : introduit une pénalité $H(f_\theta) = \sum \theta_i^2$ quadratique sur les valeurs des *paramètres* θ_i. On introduit ainsi une préférence pour des valeurs de θ proches de 0.
– design d'architecture : permet de choisir $\mathcal{F}$
– design de fonction de coût : permet de choisir $H(f)$ et ainsi d'établir une préférence sur les $f \in \mathcal{F}$
– arrêt prématuré : technique permettant de décider d'arrêter un algorithme itératif avant qu'il n'atteigne la fonction f optimale pour le coût optimisé, et qui limite ainsi la taille ou capacité "effective" de l'espace de fonctions exploré.

Pour résumer, un algorithme d'apprentissage performant pour une tâche donnée s'obtient avant tout en combinant un nombre *suffisamment élevé* de données d'en-

traînement et de bonnes connaissances à priori sur le problème, à condition qu'on puisse en disposer.

Chapitre 3

Une tentative de classifi cation des algorithmes d’apprentissage

La quantité et la diversité des algorithmes d’apprentissage rend toute entreprise de taxonomie hasardeuse. Celle, peut-être peu conventionnelle, que nous présentons ici, tente une classification des algorithmes d’après ce que nous considérons comme leur *philosophie*. Dans cette présentation nous nous limiterons la plupart du temps, par souci de simplicité, à une perspective de classification, mais certains des concepts présentés peuvent s’adapter aisément à des problèmes de régression ou d’estimation de densité.

3.1 Les modèles génératifs

Ces algorithmes partent de l'hypothèse que les données que l'on observe ont été générées par un processus aléatoire que l'on va modéliser. On part alors d'un modèle paramétrisé de ce que l'on pense pouvoir être ce processus, et on tente d'estimer les paramètres qui ont le plus vraisemblablement donné naissance aux données observées (principe du maximum de vraisemblance). Ceci correspond au cadre des statistiques paramétriques tel que développé par Fisher [30, 31, 32, 33, 2], même si en pratique on ne suppose pas forcément que le modèle envisagé est réellement le "vrai" modèle ayant généré les données, mais simplement qu'il permettra de bien généraliser une fois appliqué à de nouveaux points de test.

Un algorithme basé sur un modèle génératif aboutit généralement à un estimateur de densité $p(x)$. Mais on peut aussi utiliser ces techniques en appliquant la règle de Bayes pour construire un classifieur. Pour les problèmes de classification, on construit un modèle de densité p_c différent pour chaque classe c : $P(X = x|Y = c) = p_c(x)$, dont les paramètres sont estimés de manière à maximiser la vraisemblance des x_i qui correspondent à cette classe dans l'ensemble d'apprentissage. Puis, au moment de prendre une décision quant à la classe c d'un nouveau x qui nous est présenté, on utilise la règle de Bayes pour obtenir la probabilité à posteriori de la classe :

$$P(Y = c|X = x) = \frac{P(X=x|Y=c)P(Y=c)}{P(X=x)}$$

où $P(Y = c)$ est la probabilité à priori de la classe c (et $P(X = x)$ est un simple facteur de normalisation).

Une alternative à l'adoption de la solution de maximum de vraisemblance, est la pure approche Bayesienne qui considère l'intégrale sur toutes les valeurs possible des paramètres du modèle, en tenant compte d'une probabilité à priori sur ces valeurs (un à priori sur le modèle). Bien que très attrayante d'un point de vue théorique, l'approche Bayesienne présente souvent des difficultés contraignantes de mise en oeuvre en pratique, impliquant le recours à des approximations, et limitant leur applicabilité.

Parmi les approches inspirées de modèles génératifs qui ont emporté un grand succès, on peut citer les Chaînes de Markov Cachées (utilisées notamment dans les systèmes de reconnaissance de la parole), et les modèles de Mixtures de Gaussiennes.

3.2 Modélisation directe de la surface de décision

Le concept de surface de décision est commun à tous les algorithmes de classification dans $\mathbb{R}^n$. En effet, la capacité de décider d'une classe pour chaque point x de l'espace d'entrée induit automatiquement une partition de cet espace. Si on se limite, pour la simplicité de l'exposé, au cas de deux classes ($y \in \{-1, +1\}$), pour $x \in \mathbb{R}^n$, il en résulte une frontière (pas nécessairement continue) qui délimite les zones des deux classes, et que l'on nomme surface de décision. Il s'agit d'une "surface" de dimension $n - 1$ dans l'espace d'entrée de dimension n.

Tous les algorithmes de classification dans $\mathbb{R}^n$ induisent de telles surfaces de décision, mais seuls quelques uns en font leur point de départ : ils partent d'une hypothèse quant à la forme de cette surface de décision : linéaire, polynomiale,

etc... puis cherchent, pour cette classe de fonctions, les paramètres qui vont minimiser le critère de coût désiré (idéalement l'espérance des erreurs de classification). Ainsi, l'ancêtre des réseaux de neurones, l'algorithme du Perceptron [75] recherche une surface de décision linéaire, représentée par une *fonction de décision* $f(x) = (< w, x > +b), w \in \mathbb{R}^n, b \in \mathbb{R}$.

La surface de décision correspond à $f(x) = 0$, et la classe d'un point x est donnée par le signe de $f(x)$.

Il en va de même pour le plus récent et très populaire algorithme des SVM [11, 99], sur lequel nous reviendrons en détails au chapitre 5. Bien que le critère optimisé, qui repose sur des fondations théoriques plus solides, soit quelque peu différent de celui du Perceptron, et qu'une "astuce"[1] permette d'étendre aisément l'algorithme à la modélisation de surfaces de décision non linéaires (Polynomiales...), le point de départ des SVM n'en reste pas moins une façon de trouver une simple surface de décision linéaire convenable.

3.3 Extraction progressive de caractéristiques

Les modèles génératifs, lorsqu'ils sont utilisés comme indiqué précédemment pour la classification, suggèrent un processus qui part des classes et génère les entrées observées (la variable aléatoire X). Mais on peut également imaginer un processus inverse, qui part des entrées, et par transformations et calculs successifs, produit en sortie la décision quant à la classe correspondante, et ceci sans suppo-

[1] *l'astuce du noyau* (Anglais : *kernel trick*), qui peut d'ailleurs également s'appliquer au Perceptron [37].

ser que ces transformations aient quoi que ce soit à voir avec un processus qui aurait généré les données. Dans cette catégorie on trouve les réseaux de neurones multi-couches, qui se sont largement inspirés de nos connaissances sur le système nerveux (en particulier les architectures en couches des premiers étages du processus visuel). Dans cette optique, la couche d'entrée représente les données sensorielles brutes, et chaque couche subséquente calcule des caractéristiques de plus haut niveau, et ce progressivement jusqu'à la dernière couche, qui calcule un score pour chaque classe. Le plus bel exemple de succès de ce type d'approche est sans doute l'architecture LeNet pour la reconnaissance de caractères [55, 12, 56].

3.4 Modèles basés sur des distances à des prototypes

Un grand nombre d'algorithmes d'apprentissage se basent, pour prendre leur décision (concernant la classe d'un point de test par exemple), sur la distance calculée entre le point de test, et un certain nombre de *prototypes* (des points appartenant au même espace d'entrée que le point de test). Il faut comprendre ici la notion de *distance* au sens large, comme une mesure de similarité-dissemblance entre deux points. Les *noyaux*, sur lesquels nous reviendrons au chapitre 5, entrent généralement dans cette catégorie de "mesures de similarité".

La grande variété de ces algorithmes est due à

– La façon de choisir les prototypes : se limite-t-on à des points appartenant à l'ensemble d'apprentissage (tous ? un sous ensemble ? lesquels ?) ou bien cherche-t-on à "inventer" des prototypes qui résumeraient l'ensemble d'apprentissage (comment ?).

– Le choix de la distance permettant de mesurer la similarité entre les points. Le choix de loin le plus courant pour les problèmes dans $\mathbb{R}^n$ est de se baser sur la distance Euclidienne.
– Comment l'information de distance aux prototypes est utilisée pour prendre la décision.

Parmi les algorithmes qui conservent comme prototypes la totalité des points de l'ensemble d'apprentissage, on peut inscrire les méthodes statistiques non paramétriques classiques : K plus proches voisins, régression à noyau, estimateur de densité à fenêtres de Parzen. Nous présenterons plus en détails tous ces algorithmes dans la section 5.2. On regroupe parfois ce type de méthodes sous les termes anglais *memory based*, *template matching*, ou encore *lazy learning*. On peut voir que l'optique est très différente de celle des modèles génératifs et de l'approche statistique paramétrique exposés à la section 3.1.

Parmi les algorithmes qui "construisent" leurs prototypes, on peut mentionner l'algorithme du centroïde, qui résume chaque classe par le centroïde des points d'entraînement appartenant à cette classe (voir la section 5.3.4). Les réseaux de neurones de type RBF [73] peuvent aussi être vus comme apprenant un petit nombre de prototypes, et produisant une fonction de décision qui est une combinaison linéaire de noyaux Gaussiens (fonction de la distance aux prototypes).

3.5 Valeur relative de cette taxonomie

Il nous faut préciser que les catégories présentées ci-dessus sont quelque peu artificielles. Il ne faut pas les considérer comme une classification rigide. Pour certains

algorithmes, il est difficile de décider d'une unique catégorie (par exemple les mixtures de Gaussiennes : modèle génératif ou basé sur des distances à des prototypes ? Les arbres de décisions : extraction progressive de caractéristiques, ou modélisation d'une surface de décision linéaire par morceaux parallèle aux axes ? Dilemme également pour les RBF, et les SVMs à noyaux). La distinction n'est généralement pas aussi tranchée. Un algorithme peut être étudié sous de nombreux points de vue, autres que celui qui lui a donné naissance, et c'est souvent un exercice fort instructif.

Chapitre 4

Les défi s de la haute dimensionalité

Les données provenant de problèmes d'apprentissage concrets réels apparaissent souvent en haute ou très haute dimension : c.a.d. qu'un grand nombre de variables ont été mesurées pour chaque exemple d'apprentissage. Par exemple le profil d'un client d'une compagnie d'assurance ou d'une banque peut comporter les valeurs de plus d'une centaine de variables informatives.

Or l'apprentissage en haute dimension est un problème difficile, du fait de ce que l'on a nommé le *fléau de la dimensionalité*. Par ailleurs les intuitions géométriques valables en faible dimension peuvent se révéler fausses ou inutiles en haute dimension et certains algorithmes qui fonctionnent très bien en faible dimension peuvent donner de très pauvres performances en haute dimension.

Élaborer des modèles capables de bien généraliser en haute dimension est l'un des plus grands défis de l'apprentissage statistique.

4.1 Le fléau de la dimensionalité

Le fléau de la dimensionalité [8] fait référence à la croissance exponentielle de l'espace explorable avec le nombre de dimensions. Il suffit de penser par exemple au nombre de coins d'un hyper-cube en dimension n :

– En dimension 2 (un carré) : $2^2 = 4$

– En dimension 3 (un cube) : $2^3 = 8$

– En dimension 100 : $2^{100} = 1267650600228229401496703205376$.

On voit qu'en dimension 100 déjà, le nombre est très largement supérieur à la taille des bases de données d'apprentissage auxquelles on va typiquement avoir à faire. Ce qui n'empêche nullement ces bases de données d'être en dimension 100 ou plus ! Ainsi le nombre d'exemples dont on dispose est ridiculement faible par rapport à la taille de l'espace dans lequel ils sont disposés...

Notez que le nombre de coins d'un hyper-cube en dimensions n est le nombre de combinaisons possibles de valeurs que peuvent prendre n variables binaires, c.a.d. qui ne peuvent prendre *chacune* que deux valeurs possibles. Si les variables peuvent prendre davantage de valeurs, l'effet est encore plus dramatique, et pour des variables continues, cela devient difficile à concevoir !

4.2 Intuitions géométriques en haute dimension

En plus de la taille gigantesque des espaces en haute dimension, l'intuition géométrique qui nous guide en dimension 2 et 3 est souvent trompeuse en haute dimension.

Une particularité est que l'information de distance Euclidienne entre deux points en faible dimension est beaucoup plus informative qu'en haute dimension. On peut le comprendre dans le sens où la distance est finalement une statistique scalaire résumant n variables (les différences entre les n coordonnées des deux points). En haute dimension, seule une valeur de distance proche de 0 est réellement informative, car elle indique que *toutes* les n variables sont proches. Les valeurs de distance élevées indiquent simplement que certaines variables diffèrent, sans contenir aucune information quant à leur identité (lesquelles diffèrent beaucoup et lesquelles peu ou pas du tout, et il y a bien davantage de possibilités quant à leur identité en dimension élevée). Nous qualifions ce phénomène de *myopie* de la distance Euclidienne, car elle ne permet pas de clairement voir loin, n'étant informative que très localement. Cette *myopie* s'accroît avec la dimensionalité, car plus la dimension est élevée, plus la proportion de points éloignés par rapport aux points proches devient écrasante (si l'on suppose des données tirées d'une distribution uniforme dans un hypercube par exemple).

Un autre point qui vaut d'être mentionné est l'importance de l'extrapolation par rapport à l'interpolation. En faible dimension, on a tendance à penser en termes d'interpolation (entre un petit nombre de points d'une courbe par exemple). Mais en haute dimension, la probabilité qu'un point de test appartienne à la fermeture convexe des données d'apprentissage devient très faible. On se trouve donc la plupart du temps en situation d'extrapolation.

4.3 La notion de variété de plus faible dimension

En dépit du fléau de la dimensionalité, en pratique on parvient parfois à obtenir des résultats raisonnables même en très haute dimension (par exemple la base de données de chiffres MNIST qui est en dimension 784). La raison en est que les nombreuses variables observées ne sont généralement pas indépendantes, et que les données d'un problème sont naturellement très loin d'être distribuées uniformément dans un hypercube (sinon on n'arriverait effectivement à rien). Elles reflètent une certaine structure sous-jacente, qu'un algorithme d'apprentissage est capable de plus ou moins bien capturer.

Dans cette optique, une notion qui connaît un regain de popularité, surtout depuis la publication de [76, 92] est l'idée que des données en haute dimension pourraient typiquement être concentrées le long d'une *variété non-linéaire de dimension inférieure*. Cette situation est illustrée dans la Figure 4.1. On peut par exemple imaginer qu'un processus sous-jacent effectivement modélisable, dépendrait d'un plus petit nombre m de facteurs que la dimension n de l'espace où les données sont observées. Si l'on suppose en outre que de petites variations continues de ces facteurs devrait se traduire par de petites variations des variables observées, alors ces m facteurs constituent effectivement une paramétrisation d'une variété de dimension m dans l'espace observé de dimension $n > m$.

Cette notion de variété est aussi ce qui sous-tend la technique des *distances tangentes* [84], où le sous-espace tangent en chaque point est *déduit de connaissances à priori* sur des transformations invariantes pour la tâche en question (petites rotations ou translations de tous les pixels de l'image d'un chiffre manuscrit, qui ne changent pas sa classe).

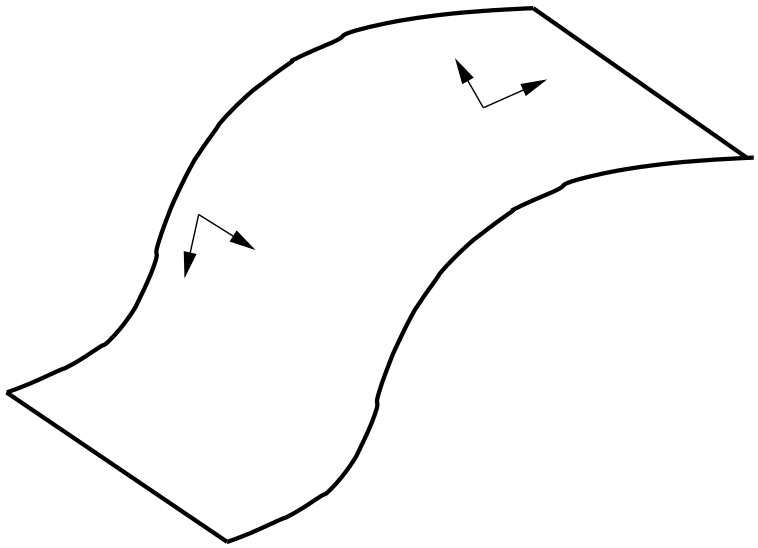

FIG. 4.1 – Illustration du concept de variété. Une variété non-linéaire de dimension 2 dans un espace de dimension 3. Deux points sont représentés, avec pour chacun deux vecteurs définissant le sous-espace tangent en ce point.