UNIVERSITÉ CHEIKH ANTA DIOP

FACULTÉ DES SCIENCES ET TECHNIQUES
DÉPARTEMENT DE MATHÉMATIQUES ET INFORMATIQUE

MÉMOIRE DE MASTER II

Présenté par

Jean Michel Amath SARR

Introduction à l'apprentissage automatique et aux réseaux de neurones artificiels

---

Soutenue le xx xxxxx xxxx

Devant le jury

Président	-	**M. Abdoulaye SENE**, Professeur titulaire, (UCAD-FST)
Membres		**M. Papa NGOM**, Professeur titulaire, (UCAD-FST)
	-	**M. Souleye KANE**, Maître de conférence titulaire, (UCAD-FST)
	-	**M. Cheikh SECK**, Maître assistant titulaire, (UCAD-FST)
Directeurs	-	**M. Mountaga LAM**, Maître de conférence titulaire, (UCAD-FST)
		M. Thimothée BROCHIER, Docteur, (IRD)

Table des matières

Résumé **iv**

Remerciements **vii**

Introduction Générale **1**

1 Fondations théoriques **4**
1.1 L'apprentissage statistique 4
La minimisation du risque empirique 6
Minimisation du risque empirique avec biais induit 6
1.2 Un modèle d'apprentissage formel 9
Généralisation 9
1.3 Un dilemme ! Tout se paye 11
Décomposition de l'erreur 11
Le théorème : "tout se paye" 12

2 Apprentissage automatique **19**
2.1 Contexte et définition 19
2.2 L'apprentissage supervisé 20
2.2.1 La régression logistique 20
2.2.2 La régression linéaire 22
2.3 Méthode d'optimisation : l'algorithme du gradient 24
2.4 Généralisation 25
2.4.1 Le procédé de validation 25
2.4.2 Inégalité de Hoeffding et conséquences 26
2.4.3 Que faire en cas d'échec de l'apprentissage ? 28
2.4.4 Régulariser 29
2.5 Schéma de synthèse 32

3 Réseaux de neurones artificiels **33**
3.1 Introduction et représentation 33
3.2 Théorème d'approximation universelle 34
3.2.1 Résultats préliminaires 34
3.2.2 Résultat principal 35

3.3 Neurone artificiel 36
3.3.1 Apprentissage à partir d'un exemple 36
Propagation 36
Rétropropagation 37
3.3.2 Apprentissage à partir de plusieurs exemples 39
3.4 Réseau de neurones entièrement connectés 39
3.4.1 Notations et exemple 40
3.4.2 Une discipline empirique 41

Conclusion générale **43**

Bibliographie **44**

Table des figures

1 L'apprentissage supervisé 1

1.1 Une classe $\mathcal{H}$ est enseignable lorsqu'il existe un algorithme d'apprentissage A permettant d'exhiber probablement une prédicteur approximativement correct. 17
1.2 Une classe $\mathcal{H}$ n'est pas enseignable lorsqu'il n'existe pas d'algorithme d'apprentissage A permettant d'exhiber probablement une prédicteur approximativement correct. 18

2.1 Fonction sigmoïde 21
2.2 Comparaison des erreurs 25
2.3 Apprentissage standard vs régularisé 30
2.4 Apprentissage automatique 32

3.1 Changement de coordonnées 34
3.2 Propagation 36
3.3 Rétropropagation 37
3.4 Réseau de neurones à deux couches 41

Remerciements

Je tiens à saisir cette occasion pour remercier mon directeur de mémoire, les membres du jury et mes professeurs de master 1 (M1) et master 2 (M2) qui m'ont enseigné aussi bien leurs connaissances scientifiques dans les cours magistraux et travaux dirigés, mais aussi leur savoir-être de mathématicien, à travers leurs remarques et attitudes face aux problèmes. Un atout qui s'est révélé précieux dans la réalisation de ce manuscrit.

En premier lieu merci à mon directeur de mémoire Mountaga Lam de sa confiance pour avoir accepté de superviser ce travail, qui a été aussi mon professeur de simulation numérique en M2, et au contact duquel j'ai beaucoup appris.

Mes vifs remerciements vont aux membres du jury :
M.Abdoulaye SENE qui a été mon professeur d'analyse fonctionnelle, et d'équations aux dérivées partielles en M1, puis mon professeur d'équations d'évolution et de contrôlabilité exacte en M2, dont la rigueur m'a été très formatrice.
M.Papa NGOM, mon professeur de probabilité et de statistique en M1, qui a toujours été d'une grande disponibilité et dont les conseils m'ont été très pertinents.
M.Souleye KANE, mon professeur de calcul scientifique en M1 qui m'a initié aux méthodes modernes d'analyse numérique.
M.Cheikh SECK pour l'intérêt qu'il porte à cette recherche en ayant accepté d'examiner ce travail.

Je remercie aussi tous mes professeurs de M1 et M2, M.Benjamin MAMPASSI, Mme FAYE SYLLA, Mme Salimata GUEYE DIAGNE, M.Gabriel Birame NDIAYE et M.Hamidou DATHE.

Un grand merci à mes parents, qui m'ont soutenu depuis mon plus jeune âge.

Enfin, je remercie Thimothée BROCHIER, Docteur à l'Institut de Recherche et Développement pour m'avoir donné l'opportunité d'appliquer les réseaux de neurones sur la reconnaissance du fond de l'océan.

Introduction Générale

Considérons X un ensemble de données, une boîte noire effectuant un traitement, et Y le résultat de ce traitement. L'objectif est de déterminer une fonction qui puisse prédire la transformation d'une entrée par ce traitement.

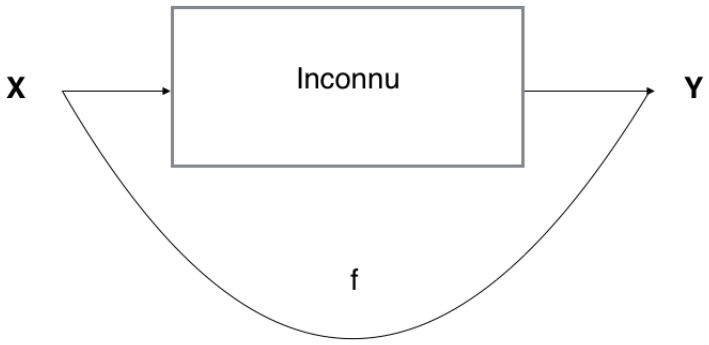

FIGURE 1 – L'apprentissage supervisé

Un *algorithme d'apprentissage automatique* est un algorithme capable d'apprendre cette fonction f à partir des données. *L'apprentissage automatique* est un ensemble de méthodes de calcul utilisant les données pour effectuer des prédictions. Comme le succès de telles méthodes dépend des données utilisées l'apprentissage automatique est fortement interdisciplinaire requérant des concepts de statistiques, de probabilité, d'optimisation, d'algèbre linéaire, mais aussi des idées d'informatique, d'algorithmique. La discipline est empirique, on veut mettre en place des systèmes capable d'apprendre. La boite noire évoquée plus haut est souvent un traitement effectué par des personnes physiques, mais très difficile à programmer directement, ça peut être par exemple reconnaître quelqu'un. Conduire une voiture, trier notre boite courriel, traduire un texte, etc.

Aujourd'hui, la plupart des entreprises de hautes technologies l'utilisent sans même qu'on s'en rende compte : quand nous voulons "taguer" un ami sur une photo et que Facebook le reconnaît instantanément, lorsque que nous utilisons Google translate pour traduire du texte. Notre boîte e-mail filtre automatiquement les messages publicitaires "spams". Lorsque nous entrons des mots-clés dans un moteur de recherche, ce dernier utilise notre requête pour améliorer ses performances. Mais il y a également des marchés

en construction qui reposent sur une intégration de ces technologies avec d'autres secteurs, c'est le cas de la voiture autonome, de la médecine prédictive, etc.

Le présent manuscrit est séparé en trois chapitres. Dans le chapitre un : fondations théoriques, nous définissons mathématiquement la notion d'apprentissage et nous montrons qu'il n'existe pas de procédure pouvant apprendre à résoudre n'importe quel problème. La conclusion de ce chapitre est qu'un algorithme d'apprentissage est nécessairement spécialisé sur une tâche précise. Le chapitre deux : apprentissage automatique présente l'approche pratique lorsque l'on veut apprendre à partir des données ainsi que sa justification théorique. Le troisième et dernier chapitre présente un algorithme ayant eu des résultats spectaculaires dans de nombreux domaines, *les réseaux de neurones*. En effet, les réseaux de neurones sont des approximateurs universels dans le sens où ils sont capables d'approcher arbitrairement n'importe qu'elle fonction continue sur un compact. Nous présentons le théorème derrière ce résultat ainsi que les équations pour mettre en place un tel algorithme. À la fin de chaque chapitre, on peut trouver un paragraphe note bibliographique récapitulant les sources principales ayant été utilisées. Nous avons inclus parmi celles-ci une conférence donnée au MIT Technology review par Andrew Ng un professeur d'apprentissage automatique à Stanford. Il s'agit d'une vidéo d'une trentaine de minutes qui donne un bon aperçu de la discipline [4].

Notations

Symboles	Signification
$\mathbb{R}$	Ensemble de réels
$\mathbb{R}_+$	Ensemble de réels positifs
$\mathbb{N}$	Ensemble de entiers naturels
$[a, b]$	Intervalle de réels
$[\![a, b]\!]$	Intervalle d'entiers naturels
$\mathcal{X}$	Domaine
$\mathcal{Y}$	Étiquettes
$\mathcal{F}$	Ensemble des fonctions définies sur $\mathcal{X}$ à valeur dans $\mathcal{Y}$
$\mathcal{H}$	Classe des prédicteurs, typiquement un sous ensemble de $\mathcal{F}$
$\mathcal{D}$	Distribution (loi de probabilité) sur un ensemble ($\mathcal{X}$ ou $\mathcal{X} \times \mathcal{Y}$)
$\mathcal{D}(A)$	Probabilité de l'évènement A par la loi $\mathcal{D}$
$x \sim \mathcal{D}$	Tirage de x selon la loi $\mathcal{D}$
$S = (x^{(1)}, \cdots, x^{(m)})$	Suite de m éléments
$S \sim \mathcal{D}^m$	Tirage i.i.d d'un échantillon S de m éléments selon la loi $\mathcal{D}$
$\mathbb{1}_{\{.\}}$	Fonction indicatrice
$\mathbb{P}$	Probabilité
$l : Z \times \mathcal{H} \to \mathbb{R}_+$	Fonction coût
$\mathbb{P}_{x\sim\mathcal{D}}(X(x))$	Probabilité de l'évènement $X(x)$ avec x tiré suivant $\mathcal{D}$
$E_{x\sim\mathcal{D}}(X(x))$	Espérance de la variable aléatoire X avec x tiré suivant $\mathcal{D}$
σ	Fonction sigmoïde
∇f	Gradient de la fonction f
$\frac{\partial f}{\partial x}$	Dérivée partielle de f par rapport à x
$\|\|.\|\|$	Norme usuelle de $\mathbb{R}^d$
$< ., . >$	Produit scalaire usuel dans $\mathbb{R}^d$
λ, α	Hyper-paramètres
$M(\mathbb{R}^d)$	Ensemble des mesures de Borel finies et signées de $\mathbb{R}^d$
μ	Mesure signée
$C(\mathbb{R}^d)$	Ensemble des fonction continues de $\mathbb{R}^d$
$\overline{G}$	Adhérence de l'ensemble G
$\forall, \exists$	Quantificateurs quelque soit et il existe
$W^{[l]}, b^{[l]}$	Paramètres de la couche l d'un réseau de neurones
$A * B$	Produit de Hadamard de deux matrices

Chapitre Un

Fondations théoriques

Introduction

Avant de parler d'apprentissage automatique, nous devons bien comprendre la notion d'apprentissage de façon formelle. Pour illustrer les nuances qu'il peut y avoir dans l'apprentissage voici deux exemples.
Rat apprenant à détecter une nourriture empoisonnée : lorsque l'on dispose une nouvelle nourriture devant un rat, il commencera par la renifler et en manger un petit morceau ; si l'effet est néfaste pour sa santé il évitera cet aliment par la suite qui sera associé à un état maladif .
La superstition du pigeon : un psychologue plaça des pigeons affamés dans une cage reliée à un mécanisme distribuant de la nourriture à intervalles réguliers. Lors de la première distribution de nourriture, les pigeons étaient engagés dans différentes activités. Par la suite chaque pigeon associait son comportement spécifique à la nourriture, cela entraînant une spécialisation de chaque pigeon à n'effectuer et que cette tâche. Qu'est-ce qui distingue les mécanismes impliquant un véritable apprentissage, de ceux entraînant une superstition ? Cette question est centrale dans l'élaboration d'apprenant automatisés.

1.1 L'apprentissage statistique

Nous introduirons le vocabulaire à l'aide d'un modèle formel et d'un exemple. Imaginons un enfant qui n'a jamais goûté de mangue, soudain pendant l'hivernage il voit ses frères et soeurs, parents et amis en manger il veut lui aussi en goûter à ce stade il n'a aucune connaissance pour savoir à l'avance si une mangue est bonne ou mauvaise. Comment ne manger que les meilleures mangues ? Une approche naïve, est de goûter toutes celles qui lui passent sous la main. Quelques années plus tard son approche a évolué, elle est plus structurée ; il regarde la couleur du fruit la tâte de ces mains (est-ce que c'est mou ou dur ?). On peut formaliser le cadre comme suit.

— *Description abstraite de la réalité* :
  — Un ensemble ou domaine $\mathcal{X}$. Dans notre exemple on peut poser $\mathcal{X} \subset \mathbb{R}^2$ il s'agit de l'ensemble des mangues. chacune représentant un vecteur ayant deux caractéristiques (couleur,texture).
  — Une classe d'étiquettes $\mathcal{Y}$. Une classe binaire, dans notre exemple $\mathcal{Y} = \{0, 1\}$représentant

le goût (bon ou mauvais)

— *Un modèle de génération des données* :
Une loi de probabilité $\mathcal{D}$ inconnue définie sur $\mathcal{X}$, cette loi décrit comment sont distribuées les mangues dans notre exemple. Une fonction cible $f : \mathcal{X} \to \mathcal{Y}$ représentant la connaissance du goût d'une mangue en fonction de sa couleur et sa texture. C'est cette fonction que l'enfant cherche à apprendre.

— *Données étiquetées* :
Un jeu d'apprentissage $S = \{(x^{(1)}, f(x^{(1)}), \cdots (x^{(m)}, f(x^{(m)})\}$, c'est une suite fini sur $\mathcal{X} \times \mathcal{Y}$.

— *Résultat de l'apprentissage* :
On appelle classe d'hypothèse ou espace d'hypothèse un ensemble de fonctions $\mathcal{H} \subseteq \mathcal{F} \coloneqq \{f, f : \mathcal{X} \to \mathcal{Y}\}$, $h \in \mathcal{H}$ est appelée règle de prédiction, hypothèse ou tout simplement prédicteur.

— *Mesure de la réussite* :
Un moyen d'évaluer l'erreur commise par un prédicteur. on la note :

$$L_{\mathcal{D},f}(h) \coloneqq \mathbb{P}_{x\sim\mathcal{D}}(h(x) \neq f(x)) = \mathcal{D}(x : h(x) \neq f(x))$$

Cette erreur est tout simplement la probabilité qu'une règle de prédiction se trompe.

Remarque 1.1. En toute rigueur on définit usuellement un espace de probabilité par un triplet : domaine, tribu, probabilité. Ici on suppose à chaque fois l'existence d'une tribu adéquate. Ce procédé est valide dans le sens ou l'existence d'une telle tribu ne pose pas de problème (ensemble des parties pour un domaine discret, et tribu borélienne pour un domaine continu).

Le vocabulaire introduit nous permet de définir formellement un algorithme d'apprentissage.

Définition 1.1. Un algorithme d'apprentissage est une fonction :

$$A : \mathcal{P}(\mathcal{X} \times \mathcal{Y}) \to \mathcal{H}$$

avec $\mathcal{P}(\mathcal{X} \times \mathcal{Y})$ représentant l'ensemble des suites de $\mathcal{X} \times \mathcal{Y}$.

Il convient d'insister sur le fait que la loi de probabilité $\mathcal{D}$ et la fonction cibe f sont inconnues, par conséquent l'erreur réelle n'est pas calculable et nous ne pouvons pas encore établir comment apprendre. Dans le paragraphe suivant nous introduisons un procédé permettant l'apprentissage.

La minimisation du risque empirique

Nous n'avons accès qu'au jeu de données S comme représentation de la loi sous jacente $\mathcal{D}$. Par conséquent nous pouvons introduire une mesure de l'erreur empirique et un moyen de sélectionner une hypothèse basé sur cette erreur. L'erreur empirique ou risque empirique est donné pour un jeu d'apprentissage échantillonné suivant la loi $\mathcal{D}$.

$$L_S(h) \coloneqq \frac{1}{m} \cdot \sum_{i=1}^{m} |h(x^{(i)}) - f(x^{(i)})|$$

Un algorithme A suit la règle de minimisation du risque empirique (MRE) si :
$A(S) = \min_{h \in \mathcal{H}} L_S(h)$. Ce mode d'apprentissage empirique bien que fonctionnel est largement insuffisant en l'état actuel, et peut échouer lamentablement dans certains cas. En effet si l'échantillon d'apprentissage S n'est pas représentatif de la distribution $\mathcal{D}$ un algorithme peut proposer une hypothèse qui échoue à généraliser. En d'autres termes une hypothèse ayant une bonne performance uniquement sur le jeu d'apprentissage. Illustrons cela par un exemple.

Exemple 1.1. Soit $\mathcal{X} = [\![-2, 2]\!], \mathcal{Y} = \{0, 1\}$. On suppose que $\mathcal{D}$ est une loi uniforme et soit la fonction cible f défini ainsi : $f(x) = \begin{cases} 1 \ si - 1 \leq x \leq 1 \\ 0 \ sinon \end{cases}$
Remarquons que la fonction $h_S(x) = \begin{cases} y \ si \ (x, y) \in S \\ 0 \ sinon \end{cases}$ a une erreur empirique nulle pour tout S. Cependant si nous avons les jeux d'apprentissage suivants $S_1 = \{(-2, 0), (1, 1)\}$ et $S_2 = \{(-2, 0), (-2, 0)\}$, $S_3 = \{(-1, 1), (0, 1), (1, 1)$ alors la règle de MRE pourrait proposer les fonctions suivantes :

$$h_{S_1} : \begin{cases} -2 \ \mapsto 0 \\ -1 \ \mapsto 0 \\ 0 \ \mapsto 0 \\ 1 \ \mapsto 1 \\ 2 \ \mapsto 0 \end{cases}, \ h_{S_2} : \begin{cases} -2 \ \mapsto 0 \\ -1 \ \mapsto 0 \\ 0 \ \mapsto 0 \\ 1 \ \mapsto 0 \\ 2 \ \mapsto 0 \end{cases}, \ h_{S_3} : \begin{cases} -2 \ \mapsto 0 \\ -1 \ \mapsto 1 \\ 0 \ \mapsto 1 \\ 1 \ \mapsto 1 \\ 2 \ \mapsto 0 \end{cases}$$

En effet ces trois fonctions minimisent l'erreur empirique. Cependant elles se comportent différemment lorsque l'on généralise : $L_{\mathcal{D},f}(h_{S_1}) = \frac{2}{5}$ et $L_{\mathcal{D},f}(h_{S_2}) = \frac{3}{5}$, $L_{\mathcal{D},f}(h_{S_1}) = 0$. On voit que suivant le jeu d'apprentissage on appréhende plus ou moins bien la réalité, nous verrons un moyen de controller cette *représentativité* de l'échantillon dans la section suivante. Dans les deux premiers cas on parle de *sur-ajustement* de l'algorithme d'apprentissage . Intuitivement cela arrive lorsqu'un apprenant déduit une règle de prédiction à l'aide d'un jeu de donnée non représentatif de la réalité sous-jacente.

Minimisation du risque empirique avec biais induit

Nous venons de voir un cas mettant à mal la règle MRE. La question qui suit, serait : existe-t-il des conditions pour que la règle MRE soit une bonne règle d'apprentissage ? Mais en passant, qu'est-ce qu'une bonne règle d'apprentissage ? Comment le formaliser ?

Nous répondrons à ces questions dans cette section.

Une bonne règle d'apprentissage exhibe un prédicteur h ayant une erreur réelle arbitrairement faible. Pour réaliser cet objectif nous allons fixer une condition et deux hypothèses. Mais tout d'abord voici une petite définition.

Définition 1.2. On dit qu'une distribution $\mathcal{D}$ sur $\mathcal{X}$ est séparable par rapport à une classe $\mathcal{H}$ et une fonction cible f, si $\exists h^* \in \mathcal{H}, L_{\mathcal{D},f}(h^*) = 0$

— On suppose que $\mathcal{H}$ est de cardinal fini. On dit qu'on biaise notre approche ou que l'on induit un biais. Concrètement, cela revient à restreindre notre recherche.

1. Nous faisons l'hypothèse que la distribution $\mathcal{D}$ est séparable par rapport à $\mathcal{H}$ Cela revient à supposer qu'il est possible d'apprendre la fonction cible. En d'autres termes dans notre exemple, on convient qu'il est possible à l'enfant (certainement après plusieurs années) de parfaitement anticiper le goût d'une mangue rien qu'avec sa texture et sa couleur. Cette hypothèse est forte et n'est pas souvent réalisée en pratique, nous la relaxerons dans la prochaine sous- section.
2. On suppose que les données sont générées indépendamment et sont identiquement distribuées suivant la loi $\mathcal{D}$ (i.i.d). On notera $S \sim \mathcal{D}^m$ pour dire que m example sont échantillonnés (i.i.d) suivant $\mathcal{D}$ puis étiquetés par la fonction f. Intuitivement cela revient à nous donner un moyen d'évaluer correctement la justesse d'un prédicteur sur n'importe quel échantillon.

On va noter $MRE_{\mathcal{H}}$ la régle de minimisation du risque empirique sur $\mathcal{H}$ et $h_S \in \arg\min_{h\in\mathcal{D}} L_S(h)$ le prédicteur exhibé par $MRE_{\mathcal{H}}$ sur S .

Rappel 1.1. Notre objectif est de proposer une règle d'apprentissage capable d'exhiber un prédicteur ayant non seulement une erreur empirique faible, mais surtout une erreur réelle arbitrairement faible.

Comment formaliser ça ? $MRE_{\mathcal{H}}$ est une règle d'apprentissage. Soit $\varepsilon \in [0, 1]$ un réel pour la précision attendue. Nous voulons obtenir dans l'idéal pour tout échantillon S donné un prédicteur h_S tel que $L_{\mathcal{D},f}(h_S) < \varepsilon$. Cependant S est généré aléatoirement, et nous pouvons obtenir un échantillon non représentatif de la distribution $\mathcal{D}$, comme ça avait été le cas dans l'exemple précédent. On va donc appeler δ la probabilité de non représentativité d'un échantillon. Dans la proposition qui suit nous montrons que l'on peut controller cette probabilité de non représentativité par une fonction de la taille m de l'échantillon S. Intuitivement cela indique que plus un échantillon (i.i.d) est large plus il est représentatif de la loi sous jacente.

Proposition 1.1. *Soit $\mathcal{X}$ un domaine, $\mathcal{Y}$ une classe binaire, $\mathcal{H}$ une classe d'hypothèses de cardinal fini. Alors pour toute distribution de probabilité $\mathcal{D}$ séparable par rapport à $\mathcal{H}$ sur $\mathcal{X}$, pour toute fonction cible $f : \mathcal{X} \to \mathcal{Y}$, pour toute précision $\varepsilon \in [0, 1]$, et pour tout échantillon $S \sim \mathcal{D}^m$:*

$$\mathcal{D}^m(\{S : L_S(h_S) > \varepsilon\}) \leq |\mathcal{H}| \cdot e^{-m\varepsilon}$$

Démonstration. Pososns $\mathcal{H}_B = \{h \in \mathcal{H}, L_{\mathcal{D},f}(h) > \varepsilon\}$, $M = \{S, \exists h \in \mathcal{H}_B, L_S(h) = 0\}$.
$\mathcal{H}_B$ est l'ensemble des mauvais prédicteurs de $\mathcal{H}$, et M représente l'ensemble des échantillons admettant au moins un mauvais prédicteur ayant une erreur empirique nulle (un mauvais prédicteur passant sous les radars pour analogie).
Premiére remarque $\{S, L_S(h_S) > \varepsilon\} \subset M$.
En effet, $(H^*) \Leftrightarrow \exists h^*, \forall S\ L_S(h^*) = 0$. En quelque sorte pour un échantillon corrompu S, la règle MRE_H va proposer un prédicteur h_S dont l'erreur réelle est supérieure à la précision attendue, mais par hypothèse $(H^*), L_S(h_S) = 0$.

On peut réecrire M de la façon suivante.

$$M = \bigcup_{h \in \mathcal{H}_B} \{S, L_S(h) = 0\}$$

Ainsi

$$\mathcal{D}^m(\{S : L_S(h_S) > \varepsilon\}) \leq \mathcal{D}^m(M) = \mathcal{D}^m(\bigcup_{h \in \mathcal{H}_B} (\{S, L_S(h) = 0\})$$

$$\Rightarrow \mathcal{D}^m(\{S : L_S(h_S) > \varepsilon\}) \leq \sum_{h \in \mathcal{H}_B} \mathcal{D}^m(\{S : L_S(h) = 0\})$$

En effet on sait qu'en probabilité si A et B sont des évènements $\mathcal{D}(A \cup B) \leq \mathcal{D}(A) + \mathcal{D}(B)$. Observons que du fait que les données sont (i.i.d) on a :

$$\mathcal{D}^m(\{S : L_S(h) = 0\}) = \prod_{i=1}^{m} \mathcal{D}(\{x^{(i)} : h(x^{(i)}) = f(x^{(i)})\}) = \prod_{i=1}^{m} (1 - L_{\mathcal{D},f}(h))$$

$$\Rightarrow \mathcal{D}^m(\{S : L_S(h_S) > \varepsilon\}) \leq \sum_{h \in \mathcal{H}_B} \prod_{i=1}^{m} (1 - L_{\mathcal{D},f}(h)) \leq |\mathcal{H}_B| \cdot (1 - \varepsilon)^m$$

or nous savons que $\forall x \in [0, 1], (1 - x) \leq e^{-x}$ on en déduit finalement que

$$\mathcal{D}^m(\{S : L_S(h_S) > \varepsilon\}) \leq \mathcal{H}_B| \cdot (1 - \varepsilon)^m \leq |\mathcal{H}| \cdot e^{-m\varepsilon} \tag{1.1}$$

□

Corollaire 1.1. *Soit $\mathcal{X}$ un domaine, $\mathcal{Y}$ une classe binaire, $\mathcal{H}$ une classe d'hypothèses de cardinal fini. Alors $\forall \varepsilon > 0, \forall \delta \in [0, 1]$ si l'entier m vérifie :*

$$m \geq \frac{1}{\varepsilon} \cdot log(\frac{|\mathcal{H}|}{\delta})$$

On a pour toute distribution $\mathcal{D}$ séparable sur $\mathcal{X}$, pour toute fonction cible f, pour tout échantillon $S \sim \mathcal{D}^m$:

$$L_{\mathcal{D},f}(h_S) \leq \varepsilon$$

avec une probabilité supérieure ou égale à $1 - \delta$, et h_S un prédicteur proposé par la règle de MRE.

Démonstration. Étant donné ε une précision attendue, et $\delta \in [0,1]$ une probabilité de non représentativité fixée $\exists m \in \mathbb{N}$ tel que $|\mathcal{H}| \cdot e^{-m\varepsilon} \leq \delta$ d'après (1.1).
Alors :

$$\frac{|\mathcal{H}|}{\delta} \leq e^{m\varepsilon} \Rightarrow \frac{1}{\varepsilon} \cdot log(\frac{|\mathcal{H}|}{\delta}) \leq m$$

□

Dans la section à venir nous allons introduire notre premier modèle d'apprentissage formel en nous basant sur le vocabulaire introduit jusqu'à présent.

1.2 Un modèle d'apprentissage formel

Définition 1.3. Une classe $\mathcal{H}$ est correctement probablement $(1-\delta)$ approximativement (ε) enseignable (CPA enseignable) si :

- $\exists m_{\mathcal{H}} : [0,1]^2 \to \mathbb{N}$.
- Il existe un algorithme tel que $\forall \delta, \varepsilon \in [0,1]$, pour fonction cible f, pour toute distribution $\mathcal{D}$ séparable sur $\mathcal{X}$ par rapport à $\mathcal{H}$ et f :
  - Si $m \geq m_{\mathcal{H}}(\delta, \varepsilon)$.
  - Alors $\forall S \sim \mathcal{D}^m$ l'algorithme proposera une hypothèse h tel qu'avec une probabilité supérieure ou égale à $(1-\delta)$ $L_{\mathcal{D},f}(h) \leq \varepsilon$.

Le modèle tient en compte deux paramètres d'incertitude : $(1-\delta)$ pour l'échantillon généré aléatoirement, et ε pour la précision attendue. La fonction $m_{\mathcal{H}}$ est appelée complexité de l'échantillonage. Comme on l'a vu précédemment, les classes d'hypothèses $\mathcal{H}$ de cardinal fini sont CAP enseignable, et il peut y avoir une infinité de fonction $m_{\mathcal{H}}$ pour cela. Donc en pratique on choisira la plus petite $i.e$

$$m_{\mathcal{H}}(\delta, \varepsilon) = \lceil \frac{1}{\varepsilon} \cdot log(\frac{|\mathcal{H}|}{\delta}) \rceil$$

Généralisation

En pratique l'hypothèse de séparabilité n'est presque jamais vérifiée, puis le modèle que nous avons présenté et concentré sur les tâches de classification. Comment généraliser à des problèmes de régression ($i.e$ ou $\mathcal{Y} \subset \mathbb{R}$) ?

Définition 1.4. Soit Z un domaine et $\mathcal{H}$ un ensemble de fonction, alors une fonction coût généralisée est une application de la forme : $l : Z \times \mathcal{H} \to \mathbb{R}^+$
avec $\forall h \in \mathcal{H}$ $l(\cdot, h)$ mesurable.

Commençons par relaxer l'hypothèse de séparabilité. Jusqu'à présent on supposait qu'il était parfaitement possible d'apprendre une connaissance ou fonction cible à partir des données issues de $\mathcal{X}$. Or dans la réalité nous ne sommes pas sûr qu'il y ait assez d'informations dans $\mathcal{X}$, dans notre exemple on supposait qu'il est possible de déterminer parfaitement le goût d'une mangue en fonction de sa texture et de sa couleur, mais peut-être que d'autres caractéristiques clés sont manquantes comme le parfum, la date de récolte etc. Comme ce manuscrit est une introduction à l'apprentissage automatique et aux réseaux de neurones

profonds qui représentent une catégorie d'algorithme d'apprentissage supervisé, le modèle que nous décrirons par la suite sera celui que nous utiliserons principalement.

- *Description abstraite de la réalité* : Un domaine $\mathcal{X} \times \mathcal{Y}$
- *Un modèle de génération des données* : Une distribution $\mathcal{D}$ sur $\mathcal{X} \times \mathcal{Y}$. L'avantage de cette approche est de pouvoir travailler avec les lois de probabilité marginales et conditionnelles. Dans notre example on aurait $\mathcal{D}(y = 1|x)$ pour représenter la **probabilité** qu'une mangue soit bonne étant donné sa couleur et sa texture.
- *Données étiquetées* : un jeu d'apprentissage
  $S = \{(x^{(1)}, y^{(1)}), \cdots (x^{(m)}, y^{(m)})\}$ une suite finie.
- *Résultat de l'apprentissage* : un prédicteur $h \in \mathcal{H}$ avec $\mathcal{H} = \{f, f : \mathcal{X} \to \mathcal{Y}\}$.
- *Mesure de la réussite* : étant donné un prédicteur h, et une fonction coût
  $l : \mathcal{X} \times \mathcal{Y} \times \mathcal{H} \to \mathbb{R}^+$:
  $$L_{\mathcal{D}}(h) \coloneqq E_{(x,y)\sim\mathcal{D}}[l((x, y), h)]$$
- *Risque empirique généralisé* : étant donné $S \sim \mathcal{D}^m$, et h un prédicteur l'erreur empirique est définie par :

$$L_S(h) \coloneqq \frac{1}{m} \cdot \sum_{i=1}^{m} l((x^{(i)}, y^{(i)}), h)$$

Remarque 1.2. La fonction coût l étant mesurable par rapport au domaine Z si $\mathcal{D}$ est une loi de probabilité sur Z, on peut générer des données aléatoirement, ainsi pour tout $h \in \mathcal{H}$ fixé, $l(\cdot, h)$ est une variable aléatoire, on peut calculer son espérance.

Une des différences fondamentales avec l'approche précédente est que désormais nous ne sommes pas sûr qu'il soit possible d'obtenir une erreur arbitrairement faible avec une règle de MRE classique. Notre but est désormais de nous rapprocher arbitrairement de de la plus petite erreur possible.

Définition 1.5. Une classe $\mathcal{H}$ est dite agnostiquement correctement probablement approximativement enseignable (ACPA enseignable) par rapport à un domaine $\mathcal{X} \times \mathcal{Y}$ et une fonction coût $l : \mathcal{X} \times \mathcal{Y} \times \mathcal{H} \to \mathbb{R}^+$ si :

- $\exists\ m_{\mathcal{H}} : [0, 1]^2 \to \mathbb{N}$
- Il existe un algorithme d'apprentissage tel que $\forall \delta, \varepsilon \in [0, 1]$, pour toute loi de probabilité $\mathcal{D}$ sur $\mathcal{X} \times \mathcal{Y}$:
  - Si $m \geq m_{\mathcal{H}}(\delta, \varepsilon)$
  - Alors $\forall S \sim \mathcal{D}^m$ l'hypothèse h_S proposé par l'algorithme vérifie :

$$L_{\mathcal{D}}(h_S) \leq \min_{h\in\mathcal{H}} L_{\mathcal{D}}(h) + \varepsilon$$

avec une probabilité supérieure ou égale à $(1 - \delta)$.

Définition 1.6. Étant donné une distribution $\mathcal{D}$ sur $\mathcal{X} \times \mathcal{Y}$ et une fonction coût l, on appelle prédicteur de **Bayes** $f_{\mathcal{D}}$ la meilleure hypothèse dans le sens suivant :

$$f_{\mathcal{D}} = inf_{g\in\mathcal{F}}\ L_{\mathcal{D}}(g)$$

Exemple 1.2. Dans le cas d'une tâche de classification binaire, si $\mathcal{D}$ est une distribution sur $\mathcal{X} \times \mathcal{Y}$

$$f_{\mathcal{D}}(x) : \begin{cases} 1 \ si \ \mathcal{D}(y = 1|x) \geq \frac{1}{2} \\ 0 \ sinon \end{cases}$$

En pratique on ne connaît pas l'estimateur de Bayes, car la distribution $\mathcal{D}$ est inconnue.

1.3 Un dilemme ! Tout se paye

Dans la section précédente nous avons vu que lorsqu'une classe d'hypothèses et trop générale une règle d'apprentissage peut sur-ajuster. Nous sommes parvenus à obtenir un bon résultat en limitant la classe d'hypothèses à une classe de cardinal fini. Le choix d'une classe d'hypothèse peut s'interpréter par une intuition préalable de la distribution recherchée. Formellement cela peut s'interpréter dans les cadres que nous venons de développer.

— Dans le modèle CAP on choisit une classe $\mathcal{H}$ en supposant que la distribution est séparable, i.e on suppose que la connaissance parfaite est dans $\mathcal{H}$.

— Dans le modèle ACAP on choisit une classe $\mathcal{H}$ en supposant qu'elle contient la meilleure hypothèse.

Dans notre exemple si l'enfant demande conseil à ses parents pour sélectionner une bonne mangue, ceux-ci lui donneront certainement des astuces pour en choisir une bonne parmi d'autres. En leur faisant confiance il réduit le nombre d'hypothèses à tester. Cependant dans l'absolu si nous restreignons trop cette classe nous pouvons "rater" la meilleure hypothèse c'est ce qu'on appelle le sous-ajustement. Pour illustrer cela imaginons que l'enfant se restreigne uniquement aux mangues vertes (il manque alors les mangues rouge-orange qui sont excellentes). Nous faisons donc face à un dilemme : comment choisir la classe $\mathcal{H}$ de sorte à limiter à la fois le sous-ajustement et le sur-ajustement ?

On peut également se demander pourquoi faire face à un tel dilemme. Peut-être qu'il existe un algorithme d'apprentissage universel capable d'apprendre n'importe qu'elle tâche sans aucune connaissance préalable. Nous démontrerons qu'il n'existe pas de tel algorithme avec le théorème "tout se paye".

Dans un premier temps cependant formalisons ce compromis entre sous et sur ajustement.

Décomposition de l'erreur

On peut décomposer l'erreur commise en deux composantes l'erreur d'approximation et l'erreur d'estimation que nous notons respectivement ϵ_{app} et ϵ_{est}. Soit une classe d'hypothèse $\mathcal{H}$ Étant donné une distribution $\mathcal{D}$, un échantillon $S \sim \mathcal{D}^m$ on peut imaginer que la règle de minimisation du risque empirique $MRE_{\mathcal{H}}$ exhibe un prédicteur h_S. Alors :

$$L_{\mathcal{D}}(h_S) = \epsilon_{app} + \epsilon_{est}$$

avec

$$\epsilon_{app} = min_{h \in \mathcal{H}} L_{\mathcal{D}}(h), \ \epsilon_{est} = L_{\mathcal{D}}(h_S) - \epsilon_{app}$$

Le choix de la classe $\mathcal{H}$ a une incidence directe sur les deux erreurs, alors que le choix de l'échantillon S n'affecte que l'erreur d'estimation. On peut se représenter l'effet du choix de la classe $\mathcal{H}$ sur ces erreurs comme un levier sur des vases communiquant. En effet :

— Choisir $\mathcal{H}$ large va diminuer l'erreur d'approximation et augmenter l'erreur d'estimation, c'est dans ce cas qu'il y a sur-ajustement

— Réduire la classe $\mathcal{H}$ aura tendance à augmenter l'erreur d'approximation et diminuer l'erreur d'estimation. Cela à du sens car lorsque la classe est moins riche, un prédicteur proposé par la règle de MRE ne ce sera pas spécialisée outre mesure sur S. C'est dans ce cas que l'on risque le sous-ajustement.

Pour résumer :

$$\text{Élargir } \mathcal{H} \Rightarrow \searrow \epsilon_{app} \; et \; \nearrow \epsilon_{est}$$
$$\text{Réduire } \mathcal{H} \Rightarrow \nearrow \epsilon_{app} \; et \; \searrow \epsilon_{est}$$

En pratique on cherchera donc une classe de sorte à garder ces erreurs faibles, nous introduirons différentes techniques pour cela dans le chapitre sur l'apprentissage automatique.

Le théorème : "tout se paye"

On peut penser a priori que le compromis évoqué plus haut est plutôt contraignant. En effet on a vu que pour apprendre quelque chose il faut restreindre l'ensemble des stratégies d'apprentissage (ou hypothèses). Cela fait sens pour une personne ; si un étudiant se mettait à jouer aux jeux-vidéos pour apprendre ses leçons il se rendrait vite compte que ça ne marche pas. Cependant, on peut aussi se demander s'il existe une procédure d'apprentissage universelle capable d'apprendre n'importe quoi sans aucune connaissance préalable sur la tâche à apprendre. Nous allons énoncer et prouver le théorème "tout se paye", qui statue que ce n'est pas possible. "Tout se paye", "rien n'est gratuit" (No free lunch en anglais) est une analogie pour dire que sans connaissance préalable sur un problème on ne peut pas apprendre correctement. Ce théorème justifie l'approche de l'apprentissage automatique, en outre on peut automatiser une tâche lorsqu'on a au moins une connaissance préalable, une expérience (hypothèse de la loi sous-jacente) pour cela. Ainsi un algorithme est très bon pour effectuer une tâche donnée quand il est basé sur des hypothèses simplificatrices pertinentes à propos de la réalité. La réalité étant complexe, on peut montrer qu'un algorithme "universel" échoue à la simple tâche de classification s'il n'a pas "vu" l'ensemble des données. En d'autres termes on va montrer que si un algorithme est exposé à un jeu de données partiel pour la classification binaire, il va exhiber une hypothèse qui échouera à généraliser car une multitude de distributions inconnus peuvent représenter les données auxquelles il n'a pas encore été exposé. Commençons par définir la fonction coût pour la classification binaire.

Définition 1.7. On considère $\mathcal{X}$ un domaine et $\mathcal{Y} = \{0, 1\}$, $\mathcal{H}$ une classe d'hypothèse, on appelle fonction coût associée à la classification binaire et on note l_{0-1}, la fonction définie par :

$$l_{0-1} : \mathcal{X} \times \mathcal{Y} \times \mathcal{H} \to \mathbb{R}_+$$
$$(x, y, h) \mapsto \begin{cases} 1 \; si \; h(x) \neq y \\ 0 \; sinon \end{cases}$$

Théorème 1.1. *Soit $\mathcal{X}$ un domaine et $\mathcal{Y} = \{0, 1\}$, $m \leq \frac{|\mathcal{X}|}{2}$, et l_{0-1} la fonction coût pour la classification binaire.*

1. *Il existe une distribution $\mathcal{D}$ sur $\mathcal{X} \times \mathcal{Y}$, une fonction d'annotation $f : \mathcal{X} \to \mathcal{Y}$, telle que $L_{\mathcal{D}}(f) = 0$.*
2. *Alors pour tout algorithme d'apprentissage A on a $\mathbb{P}_{S\sim\mathcal{D}^m}[L_{\mathcal{D}}(A(S)) \geq \frac{1}{8}] \geq \frac{1}{7}$.*

Démonstration. L'idée de la preuve est de démontrer cela sur un sous-ensemble $C \subset \mathcal{X}$ de $2m$ éléments.
Pour cela nous exploiterons les propriétés du problème tel que posé. On peut survoler la lecture de ces propriétés dans un premier temps et y revenir au fur et à mesure de la démonstration.

1. Combien y a-t-il de fonctions d'annotations différents définies sur C à valeur dans $\mathcal{Y}$? On demande le cardinal de l'ensemble $\{f, f : C \to \mathcal{Y}\}$: la réponse est 2^{2m}. Il s'agit du nombre de tirage avec remise sans ordre. Dans la suite on pose $T = 2^{2m}$ Ainsi on peut noter $(f_1, \cdots, f_T)$ l'ensemble des fonctions d'annotations.
2. Combien y a-t-il de suites de m éléments dans C ? C'est le cardinal de l'ensemble $\{(x_1, \cdots, x_m), x_i \in C\}$ il y a $2m$ choix pour chaque x_i, on en déduit qu'il y a $(2m)^m$ choix au total. On pose $K = (2m)^m$. Ainsi on peut noter $(S_1, \cdots, S_K)$ les différentes suites de m éléments de C.
3. On définit une loi jointe $\mathcal{D}_i$ sur $\mathcal{X} \times \mathcal{Y}$ par :

$$\mathcal{D}_i(x, y) = \begin{cases} \frac{1}{C} \; si \; f_i(x) = y \\ 0 \; sinon \end{cases}$$

   Alors, on peut annoter toute suite S_j de C par une fonction d'annotation f_i, ce faisant on obtient un échantillon de $\mathcal{X} \times \mathcal{Y}$ qu'on notera S_j^i.
4. $\forall x \in C$ on peut partitionner les $(f_1, \cdots, f_T)$ en $\frac{T}{2}$ paires $(f_i, f_{i'})$ telles que $f_i(c) \neq f_{i'}(c) \Leftrightarrow c = x$.
   — En effet, soit $x \in C$ et notons $B = C \setminus \{x\}$. Combien y a-t-il de fonctions $f : C \to \mathcal{Y}$? Il y en a $2^{2m-1} = \frac{T}{2}$, on les notera $(g_1, \cdots, g_{\frac{T}{2}})$.
   L'ensemble des fonctions $f : C \to \mathcal{Y}$ est de cardinal T, on peut supposer que $(f_1, \cdots, f_T) = (g_1, \cdots, g_{\frac{T}{2}};\; g_{1'}, \cdots, g_{\frac{T}{2}'})$ avec $g_i(c) = g_{i'}(c)\ \forall c \in B$ mais $g_i(x) \neq g_{i'}(x)$.

Remarquons que $\forall i \in [\![1, T]\!]$ on a $L_{\mathcal{D}_i}(f_i) = 0$. Car :

$$L_{\mathcal{D}_i}(f_i) = E_{(x,y)\sim\mathcal{D}_i}[l_{0-1}(x, y, f_i)] = \sum_{(x,y)\in C\times\mathcal{Y}} l_{0-1}(x, y, f_i)\mathcal{D}_i(x, y)$$

En effet $\forall (x, y) \in C \times \mathcal{Y}$ on a $\begin{cases} y = f_i(x) \;\Rightarrow\; l_{0-1}(x, y, f_i) = 0 \\ y \neq f_i(x) \;\Rightarrow\; \mathcal{D}_i(x, y) = 0 \end{cases}$

Nous avons montré le premier point de notre théorème. On va maintenant montrer que tout algorithme A recevant un jeu d'apprentissage $S \subset C \times \mathcal{Y}$ de m éléments va exhiber une hypothèse $A(S) : C \to \mathcal{Y}$ telle que :

$$\max_{i \in [\![1,T]\!]} E_{S \sim \mathcal{D}^m}[L_{D_i}(A(S))] \geq \frac{1}{4} \tag{1.2}$$

Ce résultat entraîne le second point du théorème (On le verra dans le **Lemme 1.1**)
L'argument majeur pour montrer cette inégalité est que pour toute suite fini de réels positifs $(x_1, \cdots, x_n)$ on a :

$$\max(x_1, \cdots, x_n) \geq \frac{1}{n} \sum_{i=1}^{n} x_i \geq \min(x_1, \cdots, x_n) \tag{1.3}$$

Pour $i \in [\![1, T]\!]$, alors il y a K échantillons $(S_1^i, \cdots, S_K^i)$ possibles de $C \times \mathcal{Y}$ ayant chacun la même probabilité d'être tiré. On obtient donc :

$$E_{S \sim \mathcal{D}_i^m}[L_{\mathcal{D}^i}(A(S))] = \sum_{S \sim \mathcal{D}_i^m} L_{\mathcal{D}_i}(A(S))\mathbb{P}(S) = \frac{1}{K} \sum_{j=1}^{K} L_{\mathcal{D}_i}(A(S_j^i)) \tag{1.4}$$

Ainsi, on obtient en utilisant l'argument (1.3) à l'équation (1.4)

$$\begin{aligned} \max_{i \in [\![1,T]\!]} \frac{1}{K} \sum_{j=1}^{K} L_{\mathcal{D}_i}(A(S_j^i)) &\geq \frac{1}{T} \sum_{i=1}^{T} \frac{1}{K} \sum_{j=1}^{K} L_{\mathcal{D}_i}(A(S_j^i)) \\ &= \frac{1}{K} \sum_{j=1}^{K} \frac{1}{T} \sum_{i=1}^{T} L_{\mathcal{D}_i}(A(S_j^i)) \\ &\geq \min_{j \in [\![1,K]\!]} \frac{1}{T} \sum_{i=1}^{T} L_{\mathcal{D}_i}(A(S_j^i)) \end{aligned} \tag{1.5}$$

Par ailleurs, pour tout prédicteur $h : C \to \mathcal{Y}$, si j est fixé $\exists \{v_1, \cdots, v_p\} \in C$ tel que $C = S_j^i \bigsqcup \{v_1, \cdots, v_p\}$ avec $p \geq m$. On a alors en utilisant l'expression de l_{0-1} :

$$\begin{aligned} L_{\mathcal{D}_i}(h) &= \frac{1}{2m} \sum_{x \in C} \mathbb{1}_{\{h \neq f_i\}}(x) \\ &\geq \frac{1}{2p} \sum_{x \in C} \mathbb{1}_{\{h \neq f_i\}}(x) \\ &\geq \frac{1}{2p} \sum_{r=1}^{p} \mathbb{1}_{\{h \neq f_i\}}(v_r) \end{aligned} \tag{1.6}$$

Comme $A(S)$ est un prédicteur, en mêlant (1.5) et (1.6) on obtient pour j fixé :

$$\begin{aligned}\frac{1}{T}\sum_{i=1}^{T} L_{\mathcal{D}_i}(A(S_j^i)) &\geq \frac{1}{T}\sum_{i=1}^{T}\frac{1}{2p}\sum_{r=1}^{p}\mathbb{1}_{\{(A(S_j^i)\neq f_i\}}(v_r)\\ &= \frac{1}{2p}\sum_{r=1}^{p}\frac{1}{T}\sum_{i=1}^{T}\mathbb{1}_{\{(A(S_j^i)\neq f_i\}}(v_r)\\ &\geq \frac{1}{2}\min_{r\in[\![1,p]\!]}\frac{1}{T}\sum_{i=1}^{T}\mathbb{1}_{\{(A(S_j^i)\neq f_i\}}(v_r) \qquad (1.7)\end{aligned}$$

Utilisons la propriété (4) du problème que nous avons énoncé plus haut. $\forall r \in [\![1,p]\!]$ on peut partitionner l'ensemble $\{f_1, \cdots, f_T\}$ en $\frac{T}{2}$ paires $(f_i, f_{i'})$ telles que
$\forall c \in C\ f_i(c) \neq f_{i'}(c) \Leftrightarrow c = v_r$ ainsi, on obtient :

$$\sum_{i=1}^{T}\mathbb{1}_{\{(A(S_j^i)\neq f_i\}}(v_r) = \sum_{i=1}^{\frac{T}{2}}\mathbb{1}_{\{(A(S_j^i)\neq f_i\}}(v_r) + \mathbb{1}_{\{(A(S_j^i)\neq f_{i'}\}}(v_r) = \frac{T}{2} \qquad (1.8)$$

En effet la propriété (4) implique que :

$$\mathbb{1}_{\{(A(S_j^i)\neq f_i\}}(v_r) + \mathbb{1}_{\{(A(S_j^i)\neq f_{i'}\}}(v_r) = 1$$

On en déduit que l'équation (1.8) combiné aux inéquations (1.7) et (1.5) donne l'équation (1.2). En appliquant la troisième inégalité du **Lemme 1.1** (la relation (1.11)) il nous vient $\mathbb{P}_{S\sim\mathcal{D}^m}[L_{\mathcal{D}}(A(S)) \geq \frac{1}{8}] \geq \frac{1}{7}$. Ce qui conclut la preuve.

□

Lemme 1.1. *Soit une variable aléatoire réelle X à valeurs dans $[0,1]$ de densité f_X, alors $\forall\, a \in [0,1]$ on a :*

$$\mathbb{P}(X \geq a) \leq \frac{E[X]}{a} \qquad (1.9)$$

$$\mathbb{P}(X > 1-a) \geq \frac{E[X]-(1-a)}{a} \qquad (1.10)$$

$$\mathbb{P}(X > a) \geq \frac{E[X]-a}{1-a} \qquad (1.11)$$

Démonstration. La première expression est une inégalité de Markov.

$$E[X] = \int_{\mathbb{R}} x\cdot f_X(x)dx \geq \int_{x\geq a} x\cdot f_X(x)dx \geq a\int_{x\geq a} f_X(x)dx$$
$$\Rightarrow \frac{E[X]}{a} \geq \mathbb{P}(X \geq a)$$

Posons $Y = 1 - X$, alors Y est à valeurs dans $[0, 1]$ et $E[Y] = 1 - E[X]$. En appliquant l'inégalité de Markov on obtient :

$$\mathbb{P}(Y \geq a) \leq \frac{E[Y]}{a} \Rightarrow \mathbb{P}(1 - X \geq a) \leq \frac{1 - E[X]}{a}$$

Or

$$\{1 - X \geq a\} = \{1 - a \geq X\} = \{X > 1 - a\}^c$$

Donc

$$\begin{aligned} \mathbb{P}(1 - X \geq a) = 1 - \mathbb{P}(X > 1 - a) &\leq \frac{1 - E[X]}{a} \\ \Rightarrow \frac{E[X] - (1 - a)}{a} &\leq \mathbb{P}(X > 1 - a) \end{aligned}$$

Ce qui prouve la deuxième inégalité. La troisième s'obtient en posant $b = 1 - a$, on a $b \in [0, 1]$ alors

$$\frac{E[X] - b}{1 - b} \leq \mathbb{P}(X > b)$$

□

Pour conclure ce chapitre, donnons un corollaire de ce théorème et récapitulons les idées avancées à l'aide des figures 1.1 et 1.2.

Corollaire 1.2. *Soit $\mathcal{X}$ un domaine infini et $\mathcal{Y} = \{0, 1\}$ et $\mathcal{H} = \mathcal{F}$ alors $\mathcal{H}$ nest pas PAC enseignable.*

Note bibliographique Les travaux présentés ici sont dus à Valiant, Vapnik et Chervonenkis, qui sont les plus cités dans les manuscrits que j'ai effectivement utilisés. Le texte présent est beaucoup inspiré de [1]. En français il y a aussi une thèse cf [3] qui a été faite sur le sujet en 2001, et un ouvrage très récent [2] sur l'apprentissage statistique. L'approche présentée ici n'a pas la prétention d'être exhaustive, le lecteur interessé par le sujet trouvera une matière largement plus développée dans les trois ouvrages cités précédemment. Notamment le cardinal d'une classe n'est pas le facteur fondamental pour expliquer son enseignabilité, une autre notion - dimention de Vapnik Chervonenkis (dimension VC) permet de caractériser une classe, ce qui conduit au théorème fondamental de l'apprentissage statistique.

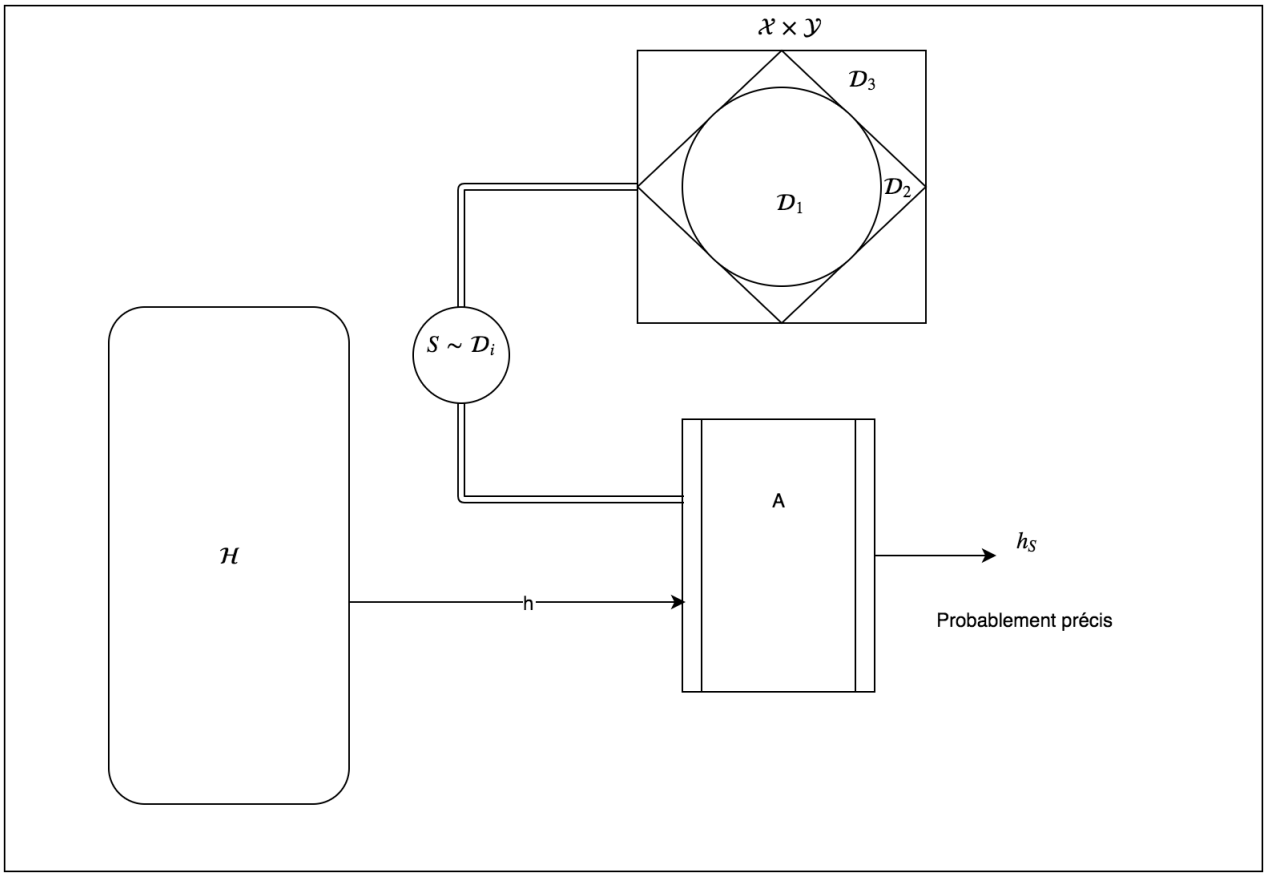

FIGURE 1.1 – Une classe $\mathcal{H}$ est enseignable lorsqu'il existe un algorithme d'apprentissage A permettant d'exhiber probablement une prédicteur approximativement correct.

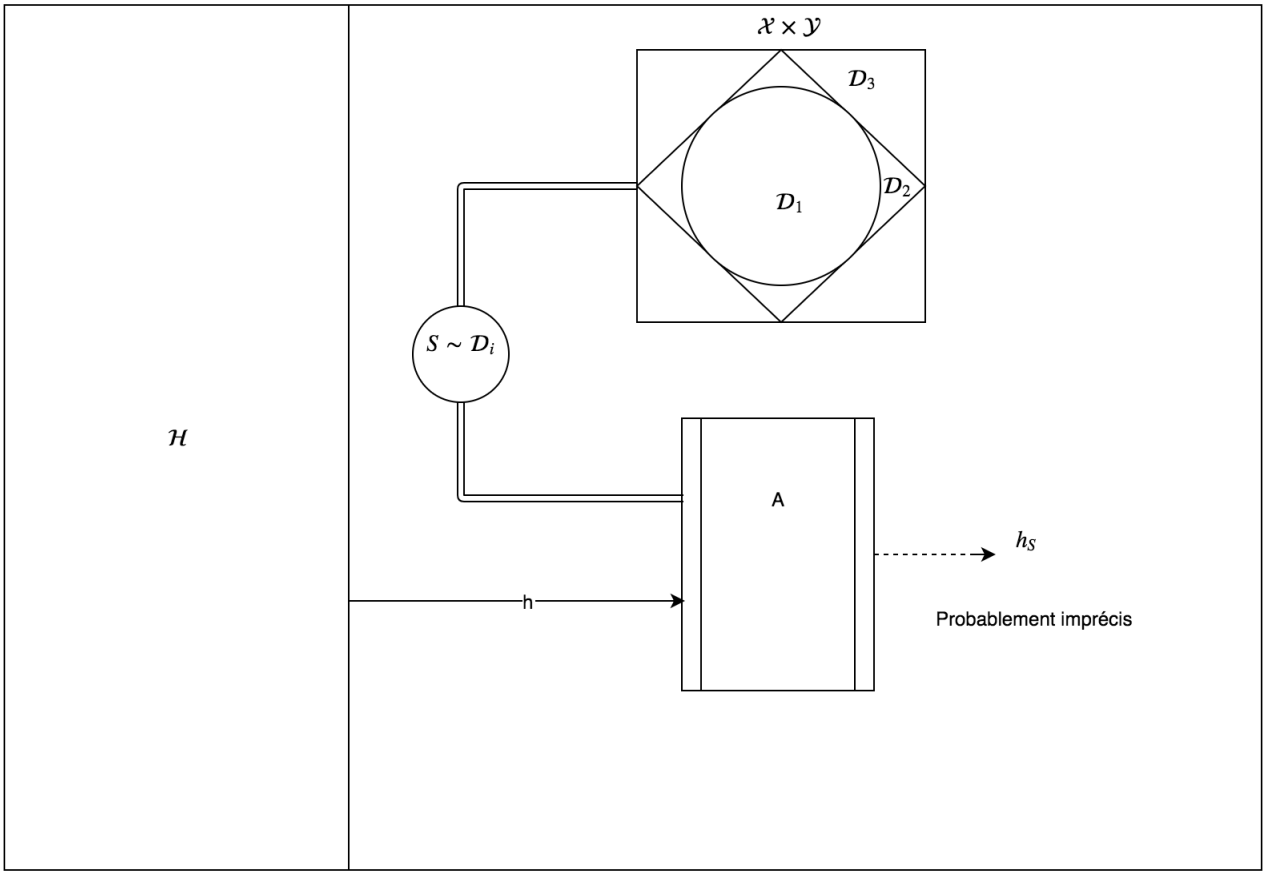

FIGURE 1.2 – Une classe $\mathcal{H}$ n'est pas enseignable lorsqu'il n'existe pas d'algorithme d'apprentissage A permettant d'exhiber probablement une prédicteur approximativement correct.

Chapitre Deux

Apprentissage automatique

Sommaire

1.1 L'apprentissage statistique 4
1.2 Un modèle d'apprentissage formel 9
1.3 Un dilemme ! Tout se paye 11

2.1 Contexte et définition

Les ordinateurs nous surpassent quand il s'agit de calculer, en effet, il est facile à un ordinateur de calculer la racine d'un nombre réel ou de multiplier deux matrices entre elles. En d'autres termes toute opération pouvant s'interpréter formellement ou de façon logique peut être traitée à grande vitesse par les processeurs modernes.

Cependant, nous effectuons tous les jours des tâches simples ; mais extraordinairement difficiles à programmer directement, par exemple :

— reconnaître un animal sur une image.
— comprendre les paroles d'une chanson.
— jouer à un jeu de stratégie comme les échecs, les dames, le go.
— conduire une voiture.
— traduire un texte d'une langue vers une autre.
— effectuer un diagnostic médical.

L'apprentissage automatique ("machine learning" en anglais) est le domaine d'étude qui permet aux ordinateurs d'apprendre sans être programmé explicitement. En outre, l'objectif est de mettre au point des algorithmes d'apprentissage automatique. Une pièce maîtresse pour réaliser un bon algorithme est d'avoir des données. Une des raisons de la popularité de ces méthodes est que nous vivons dans une ère de production massive de données, et les besoins d'automatisation de leur traitement sont de plus en plus urgents. Il existe deux grandes catégories d'algorithmes d'apprentissage :

— l'apprentissage supervisé
— l'apprenissage non supervisé

Il existe une troisième catégorie l'apprentissage par renforcement, mais est utilisé moins fréquemment. Nous avons montré avec le théorème tout se paye qu'il n'y a pas d'algorithme universel, en d'autres termes la conception d'un algorithme d'apprentissage automatique

est toujours lié à un problème donné, une définition fait consensus aujourd'hui (voir [7] chap 5).

Définition 2.1. Un algorithme apprend d'une expérience E par rapport à une tâche T mesurée par un indicateur P si la performance mesurée par P à l'expérience T s'améliore avec l'expérience E.

Exemple 2.1. Le jeu d'échecs
E : l'expérience de jouer plusieurs parties.
T : le fait de jouer.
P : la probabilité de gagner la partie.

Ainsi, une procédure capable d'augmenter ses chances de gagner à chaque partie est un algorithme d'apprentissage automatique. Le cadre de l'apprentissage statistique évoqué au chapitre 1 nous fournit les outils pour raisonner à propos de la mise en oeuvre effective de cela. Dans ce mémoire, nous avons pris le parti de présenter l'apprentissage supervisé, car c'est le domaine générant le plus de valeur commerciale aujourd'hui [4], par ailleurs la plupart des réseaux de neurones constituent des algorithmes d'apprentissage supervisé.

2.2 L'apprentissage supervisé

L'apprentissage supervisé, qu'est-ce que c'est ? Il s'agit de permettre à l'ordinateur d'apprendre sous la supervision d'un être humain. Exemple, je veux élaborer un algorithme apprenant à trier les « spams » de ma boîte courriel. Il faut lui montrer des centaines voire des milliers d'exemples de « spams » pour qu'il puisse les reconnaître cf [5] on parle alors de classification.

Dans un autre contexte, on peut classifier des patients atteints d'une tumeur en groupe à risque en fonction de la taille de leur tumeur.Dans un autre contexte, on peut classifier des patients atteints d'une tumeur en groupe à risque Dans une certaine mesure, on cherche à permettre à l'ordinateur d'approcher notre jugement/expertise, d'où le terme d'apprentissage supervisé. De nombreuses catégories de problèmes peuvent être résolus par l'apprentissage supervisé, pour n'en citer que quelques-uns il y a les problèmes de classification et de régression. Nous reprendrons les notations établis précédemment.

2.2.1 La régression logistique

Dans cette sous section nous allons voir un algorithme d'apprentissage automatique spécialisé dans les tâches de classification binaire. Nous allons illustrer sa mise en forme avec un exemple.

Exemple 2.2. Classification des mangues
E : l'expérience de goûter plusieurs mangues.
T : classer une nouvelle mangue (prédire son goût au vu de ses caractéristiques).
P : la probabilité de se tromper.

Mise en forme du problème

— *Description de la réalité* : $\mathcal{X} \subset \mathbb{R}^d$, $\mathcal{Y} = \{0, 1\}$. On pose comme domaine $\mathcal{X} \times \mathcal{Y}$(d le nombre de caractéristiques associé à une mangue).
— *Un modèle de génération des données* : une loi jointe $\mathcal{D}$ sur $\mathcal{X} \times \mathcal{Y}$. Observons que dans ce contexte la loi conditionnelle $\mathcal{D}(y = 1|x)$ nous indique la probabilité qu'une mangue soit bonne étant donné ses caractéristiques (couleur, texture, parfum, etc).
— *Données étiquetées* : $S = \{(x^{(1)}, y^{(1)}), \cdots, (x^{(m)}, y^{(m)})\} \subset \mathcal{X} \times \mathcal{Y}$.

Définition 2.2. On définit ainsi la fonction sigmoïde :

$$\begin{aligned} \sigma : \mathbb{R} &\to \mathbb{R}_+ \\ x &\mapsto \frac{1}{1 + e^{-x}} \end{aligned}$$

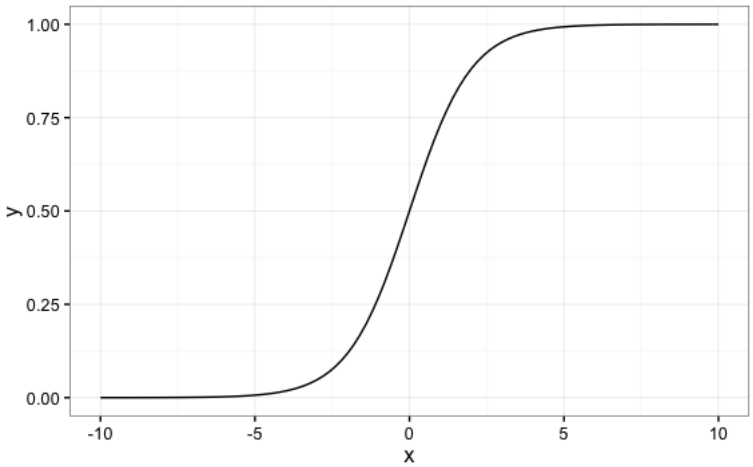

FIGURE 2.1 – Fonction sigmoïde

Remarque 2.1. Premier constat : $\begin{cases} x \geq 0 \ \Rightarrow \sigma(x) \geq \frac{1}{2} \\ x < 0 \ \Rightarrow \sigma(x) < \frac{1}{2} \end{cases}$

Dans la régression logistique, on cherche à estimer la distribution conditionnelle $\mathcal{D}(y = 1|x)$, puis on propose un prédicteur optimal pour cette distribution. La question qui suit est alors, comment représenter cette distribution ? Comme on l'a vu au chapitre 1 le théorème "tout se paye" implique que pour avoir un bon algorithme il faut avoir une connaissance préalable de la tâche à apprendre. Cela se matérialise dans la régression logistique par l'hypothèse que la distribution conditionnelle est paramétrique. Plus précisément :

$$\mathcal{D}(y = 1|x, w) = \sigma(< w, x >)$$

avec $< \cdot, \cdot >$ le produit scalaire usuel dans $\mathbb{R}^d$.
On pose $h_w(x) = \sigma(< w, x >)$ comme estimation de la distribution conditionnelle. Alors le prédicteur associé serait :

$$f_{\mathcal{D}}(x) : \begin{cases} 1 \ si \ h_w(x) \geq \frac{1}{2} \\ 0 \ sinon \end{cases}$$

Définition 2.3. Dans le cadre de la classification binaire on appelle fonction de décision la courbe permettant de séparer les données. Elle correspond à l'hypersurface suivante :

$$\{x \in \mathbb{R}^d,\, h_w(x) = \frac{1}{2}\}$$

La fonction coût l provenant de ce modèle est la suivante :

$$l(x, y, h_w) = -(y \cdot \log(h_w(x)) + (1 - y) \cdot \log(1 - h_w(x)))$$

Notons que cette fonction est convexe, nous reviendrons sur son origine dans une section suivante. Trouver la meilleure distribution conditionnelle au vu des données revient ici à minimiser le risque empirique sur $\mathbb{R}^d$, si nous notons h_S la meilleure estimation de cette loi, on a :

$$h_S = \min_{w \in \mathbb{R}^d} L_S(h_w) = \min_{w \in \mathbb{R}^d} \frac{1}{m} \cdot \sum_{i=1}^{m} l(x^{(i)}, y^{(i)}, h_w) \tag{2.1}$$

Nous trouverons ce minimum avec une méthode itérative, celle du gradient, que nous présenterons plus bas. Pour récapituler l'algorithme A cherche à minimiser le risque empirique. En faisant ainsi, on approxime la distribution conditionnelle sous-jacente, et on propose un prédicteur optimal pour la distribution choisie.

2.2.2 La régression linéaire

La tâche de régression est un outil statistique permettant de modéliser une relation entre des variables explicatives et une valeur réelle en sortie. La régression linéaire comme son nom l'indique suppose une relation linéaire entre variables explicatives et valeurs de sorties, c'est l'un des modèle les plus populaire, il est utilisé dans de nombreux secteurs ; banque, biologie, relations humaine, etc.

Exemple 2.3. Régression

- Proposer un score (credit-scoring) pour un individu demandant un crédit à la consommation connaissant son historique bancaire.
- Prédire le poids d'un bébé en fonction de son âge et de son poids de naissance.
- Prédire le prix d'un appartement dans une ville donnée, connaissant son nombre de chambres.

Comme précédemment, on a en général un jeu de données S et l'on cherche à tirer un maximum d'information de ce dernier, le problème se met en forme de la manière suivante :

Mise en forme du problème

- *Description de la réalité* : $\mathcal{X} \subset \mathbb{R}^d$, $\mathcal{Y} \subset \mathbb{R}$.
- Une classe de prédicteurs $\mathcal{H} = \{u + b :\, u \in \mathcal{L}(\mathbb{R}^d, \mathbb{R}), b \in \mathbb{R}\}$
- Un jeu de donnée $S = \{(x^{(1)}, y^{(1)}), \cdots, (x^{(m)}, y^{(m)})\} \subset \mathcal{X} \times \mathcal{Y}$
- Une fonction coût : $l(x, y, h) = (h(x) - y)^2$

On parle de fonctions linéaires, mais en réalité c'est un abus de language, il s'agit plutôt de fonctions affines, ainsi une telle fonction est caractérisée par un paramètre $w \in \mathbb{R}^d$ et

un réel $b \in \mathbb{R}$, plus précisément on peut l'exprimer ainsi :

$$\begin{aligned} h_{w,b} : \mathbb{R}^d &\to \mathbb{R} \\ x &\mapsto < w, x > + b \end{aligned}$$

Pour trouver le meilleur prédicteur on cherche donc à minimiser le risque empirique :

$$h_S = \min_{h \in \mathcal{H}} L_S(h) = \min_{w \in \mathbb{R}^d, b \in \mathbb{R}} \frac{1}{m} \sum_{i=1}^{m} l(x^{(i)}, y^{(i)}, h_{w,b}) \tag{2.2}$$

Avant d'aller plus loin sur la méthode d'optimisation, il serait intéressant de dire un mot sur l'origine des fonctions coût.

Note sur les fonctions coûts

Dans le cas de la régression linéaire la fonction coût est le carré de la norme euclidienne, cela fait sens car le problème d'optimisation est dans $\mathbb{R}$ un espace euclidien, on aurait très bien pu utiliser d'autres normes usuelles, cependant l'erreur quadratique est souvent la plus utilisé dans la littérature.
Concernant la régression logistique, nous cherchons à estimer une loi de probabilité paramétrique, la fonction coût provient directement de l'estimateur du maximum de vraisemblance. Cet estimateur est souvent utilisé en apprentissage automatique car on peut montrer qu'il est asymptotiquement ($m \to \infty$) sans biais et efficace. La vraisemblance de la loi conditionnelle s'écrit ainsi :

$$\mathcal{L}(y|x^{(1)}, \cdots, x^{(m)}; w) = \prod_{i=1}^{m} \mathcal{D}(y|x^{(i)}, w)$$

Maximiser la vraisemblance revient à minimiser la négative log vraisemblance, (la fonction -log est convexe). Ainsi :

$$\max_{w \in \mathbb{R}^d} \mathcal{L}(y|x^{(1)}, \cdots, x^{(m)}; w) = \min_{w \in \mathbb{R}^d} \sum_{i=1}^{m} \log(\mathcal{D}(y|x^{(i)}, w))$$

Par hypothèse $\mathcal{D}(y = 1|x^{(i)}, w) = \sigma(< w, x^{(i)} >$ et $\mathcal{D}(y = 0|x^{(i)}, w) = (1 - \sigma(< w, x^{(i)} >)$
Au final le problème d'optimisation est le suivant :

$$\min_{w \in \mathbb{R}^d} \sum_{i=1}^{m} y^{(i)} \cdot \log(\sigma(< w, x^{(i)} >) + (1 - y^{(i)}) \cdot \log(1 - \sigma(< w, x^{(i)} >)$$

Ce problème est équivalent à celui formulé en (2.1). À ce point le lecteur peut se demander ce qui différencie l'apprentissage automatique de l'optimisation. La différence est importante, et sera présentée dans la sous-section consacrée à la généralisation. Mais avant d'en venir à cela, il nous faut aussi développer l'aspect numérique de la discipline. En effet, nous avons mis en forme le problème et formulé un problème d'optimisation, là où plusieurs méthodes peuvent être utilisées, nous allons présenter l'algorithme du gradient, qui peut être utilisée systématiquement dans la plupart des problèmes d'apprentissage.

2.3 Méthode d'optimisation : l'algorithme du gradient

La méthode du gradient ou de la plus forte pente est un algorithme d'optimisation différentiable utilisée très fréquemment en apprentissage automatique. En général on cherche à minimiser une fonction à entrée multiple $f : \mathbb{R}^n \to \mathbb{R}$, pour cela on utilise les dérivées partielles $\frac{\partial f}{\partial x}$ qui mesurent le taux de variation de f dans les n directions. Le gradient de f en x s'exprimant $\nabla f(x) = (\frac{\partial f}{\partial x_1}, \cdots, \frac{\partial f}{\partial x_n})$ est le vecteur des taux de variations de la fonction f dans chacune des n directions.
En calcul différentiel les points critiques d'une fonction sont ceux annulant le gradient, ainsi on cherche l'ensemble suivant :

$$\{x \in \mathbb{R}^n : \|\nabla f(x)\| = 0\} = \{x \in \mathbb{R}^n : \frac{\partial f(x)}{\partial xi} = 0, i = 1, \cdots, n\}$$

Dans le cas ou la fonction à minimiser est convexe, il n'y a pas de points selles et le minimum est global. Alors minimiser f revient à chercher la direction dans laquelle f décroit le plus rapidement. La méthode de la plus forte descente est une méthode itérative consistant à choisir une valeur initiale et à prendre la direction opposée au gradient. On peut le résumer par l'algorithme suivant :

Data : $f : \mathbb{R}^n \to \mathbb{R}, x \in \mathbb{R}^n, \varepsilon, \alpha \in \mathbb{R}$;
initialization x = 0;
while $\|\nabla f(x)\| > \varepsilon$ **do**
  $x \coloneqq x - \alpha \nabla f(x)$;
end

Algorithme 1 : Méthode de la plus forte pente

Ici ε contrôle la précision attendue, et α est le coefficient d'apprentissage, c'est la taille du pas que l'on fait vers le minimum, en apprentissage automatique ce type de coefficient est appelé un hyper-paramètre (nous reviendrons sur cette notion d'hyper-paramètre dans la section traitant de la généralisation).

La méthode que nous venons de décrire permet de résoudre de manière itérative des problèmes d'optimisation convexe sans se poser de question. Dans le cas ou la fonction à minimiser n'est pas convexe, il n'y a pas de garantie de trouver un minimum global, à ce moment la démarche devient plus complexe, et d'autres techniques entrent en jeu : optimisation sous contrainte, méthodes d'optimisation du second ordre, etc.

2.4 Généralisation

Le défi principal en apprentissage automatique est d'avoir un prédicteur performant sur un jeu de données inconnu. Cette propriété fondamentale pour un algorithme d'apprentissage est appelé généralisation, on dit qu'un algorithme généralise bien. Dans un problème d'apprentissage typique on a accès à un jeu d'apprentissage S_1, l'erreur commise par le prédicteur sur ces données est appelé erreur d'apprentissage. La différence majeure entre les problèmes d'optimisation et ceux d'apprentissage automatique est que l'on cherche un prédicteur minimisant l'erreur d'apprentissage mais qui également généralise bien, en d'autres termes, on veut que l'erreur réelle soit faible. Comme on l'a vu au chapitre 1, cette erreur est mesuré par l'espérance d'une fonction coût : $E_{(x,y)\sim\mathcal{D}}[l(x,y,h)]$ ou les données (x,y) sont tirées selon une distribution de probabilité inconnue $\mathcal{D}$. Nous illustrerons toutes ces notions avec un exemple.

2.4.1 Le procédé de validation

L'erreur de Bayes est la plus petite erreur possible sur un problème, $\varepsilon_{Bayes} = \min_{h\in\mathcal{F}} L_{\mathcal{D}}(h)$, elle dépend de la distribution inconnue $\mathcal{D}$, et est inconnue. Le théorème "tout se paye" énoncé au chapitre 1 nous rappelle que notre algorithme doit être sélectionné en tenant compte d'une connaissance préalable du problème. Cette connaissance préalable peut être modélisée par le choix d'une classe d'hypothèses $\mathcal{H}$, avec cette classe vient une erreur minimale associé, l'erreur d'approximation $\varepsilon_{app} = \min_{h\in\mathcal{H}} L_{\mathcal{D}}(h)$.

Supposons que notre algorithme exhibe un prédicteur h_S en se basant sur le jeu de données $S \sim \mathcal{D}^m$, l'erreur réelle se décompose ainsi, comme vu au chapitre un $L_{\mathcal{D}}(h_S) = \varepsilon_{app} + \varepsilon_{est}$.

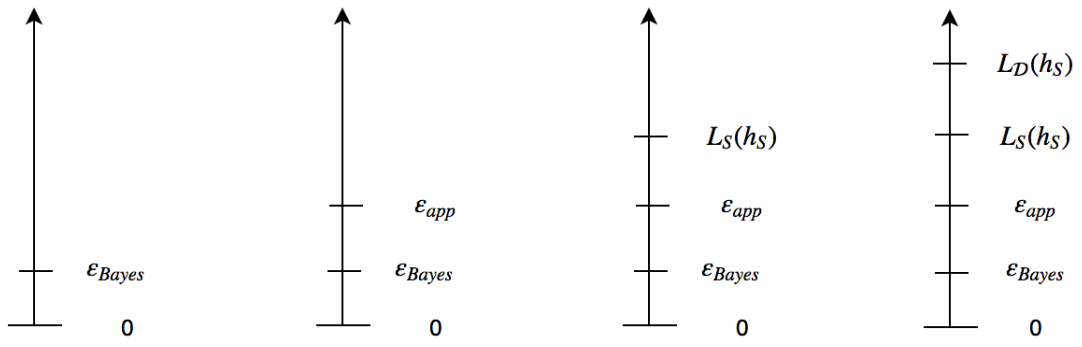

FIGURE 2.2 – Comparaison des erreurs

Cependant $L_S(h_S) \leq L_{\mathcal{D}}(h_S)$, en effet, le prédicteur h_S est adapté au données S, et à

priori il n'a aucune raison qu'il généralise bien. Au contraire il a toute les chances de commettre de nombreuses erreurs sur des données inconnues. C'est pour cela que l'on cherche à minimiser l'erreur réelle. La question est maintenant : comment évaluer l'erreur réelle d'un prédicteur donné ?

En pratique on utilise un jeu d'essai T i.i.d suivant $\mathcal{D}$ différent de S et on mesure l'erreur empirique $L_T(h_S)$ cette approche se justifie en montrant que $L_{\mathcal{D}}(h_S) \approx L_T(h_S)$, i.e l'erreur empirique commise sur un jeu test est approximativement égale à l'erreur réelle commise par h_S. Nous allons préciser tout ceci avec les résultats suivants.

2.4.2 Inégalité de Hoeffding et conséquences

Lemme 2.1. *(Inégalité de Hoeffding)*
Soient $\theta_1, \cdots, \theta_m$ des variables aléatoires i.i.d avec $E[\theta_i] = \mu$ et $\mathbb{P}(a \leq \theta_i \leq b) = 1$, alors $\forall \varepsilon > 0$ on a

$$\mathbb{P}\left(\left|\frac{1}{m}\sum_{i=1}^{m}\theta_i - \mu\right| > \varepsilon\right) \leq 2 \cdot exp\left(\frac{-2m\varepsilon^2}{(b-a)^2}\right)$$

Dans notre étude les variables aléatoires correspondent aux $l(x^{(i)}, h_S)$ avec $l(\cdot,\cdot)$ la fonction coût et $S = (x^{(1)}, \cdots, x^{(m)}) \sim \mathcal{D}^m$. Pour simplifier les expressions dans la suite on va supposer que la fonction l est à valeur dans $[0, 1]$, ainsi $(b - a) = 1$.

Proposition 2.1. *Soit g un prédicteur, $\forall \varepsilon, \delta > 0$*

$$m \geq \frac{1}{2\varepsilon^2} \cdot ln(\frac{2}{\delta}) \Rightarrow \mathcal{D}^m\{T : |L_T(g) - L_{\mathcal{D}}(g)| \leq \varepsilon\} \geq 1 - \delta$$

Démonstration. On fixe $\varepsilon, \delta > 0$ et $T \sim \mathcal{D}^m$, alors $L_T(g) = \frac{1}{m}\sum_{i=1}^{m} l(x^{(i)}, g)$, et $L_{\mathcal{D}}(g) = E_{x\sim\mathcal{D}}[l(x, g)]$. On a alors d'après l'inégalité de Hoeffding :

$$\mathbb{P}\left(\left|\frac{1}{m}\sum_{i=1}^{m} l(x^{(i)}, g) - L_{\mathcal{D}}(g)\right| > \varepsilon\right) = \mathcal{D}^m\{T : |L_T(g) - L_{\mathcal{D}}(g)| > \varepsilon\} \leq 2e^{-2m\varepsilon^2}$$

δ étant fixé, pour m assez grand $\mathcal{D}^m\{T : |L_T(g) - L_{\mathcal{D}}(g)| > \varepsilon\} \leq \delta$ il suffit pour cela que

$$2e^{-2m\varepsilon^2} \leq \delta \Leftrightarrow \frac{2}{\delta} \leq e^{2m\varepsilon^2} \Leftrightarrow ln(\frac{2}{\delta}) \leq 2m\varepsilon^2 \Leftrightarrow \frac{1}{2\varepsilon^2} ln(\frac{2}{\delta}) \leq m$$

Pour conclure, il suffit de prendre l'évènement contraire et on obtient

$$\mathcal{D}^m\{T : |L_T(g) - L_{\mathcal{D}}(g)| \leq \varepsilon\} \geq 1 - \delta$$

□

Cette proposition nous dit en somme que pour toute précision exigée ε et toute probabilité δ si m est assez grand, on a pour tout $T \sim \mathcal{D}^m$ on est sûr avec une probabilité

supérieure à $1-\delta$ que l'erreur empirique de g sur T proche de l'erreur réelle de g à epsilon près.

En pratique on partitionne notre jeu de données en un jeu d'apprentissage S et un jeu d'apprentissage T . Une partition possible est de prendre 70% pour S et 30 % pour T. Évidemment, cela fonctionne seulement si nous avons assez de données, ($i.e \geq 100$ exemples) sinon on utilise une méthode appelée "k-fold-validation". Le corollaire suivant donne la précision que l'on peut espérer pour un jeu de données de taille m fixé.

Corollaire 2.1. *Soit g un prédicteur et $\delta > 0$, alors avec une probabilité supérieure ou égale à $1-\delta$ sur le choix d'un échantillon T de taille m on a*

$$|L_{\mathcal{D}}(g) - L_T(g)| \leq \sqrt{\frac{1}{2m} ln(\frac{2}{\delta})}$$

Démonstration. Soit $m(\varepsilon) = \frac{1}{2\varepsilon^2} ln(\frac{2}{\delta})$, on remarque que

$$m(\varepsilon) \leq m \Leftrightarrow \sqrt{\frac{1}{2m} ln(\frac{2}{\delta})} \leq \varepsilon$$

Alors d'après la proposition 2.1 $\forall \delta, \varepsilon > 0$

$$m(\varepsilon) \leq m \Rightarrow \mathcal{D}^m \left\{ T : |L_{\mathcal{D}}(g) - L_T(g)| > \sqrt{\frac{1}{2m} ln(\frac{2}{\delta})} \right\} \leq \delta$$

Pour conclure la preuve il suffit alors de prendre l'évènement contraire comme précédemment. □

Nous savons donc comment obtenir une approximation de l'erreur réelle d'un prédicteur. En pratique on cherche un prédicteur avec une erreur faible à la fois sur le jeu d'entraînement mais aussi sur n'importe quel jeu d'essai, alors seulement nous aurons un prédicteur qui généralise bien. La question qui suit est : que faire si l'apprentissage échoue ? Avant d'aller plus loin donnons un exemple pour illustrer toutes ces notions.

Exemple 2.4. Nous allons modéliser l'apprentissage effectué par un étudiant pour son examen avec les notions introduites jusqu'à présent.

Mise en forme

- Description de la réalité : $\mathcal{X}$ l'ensemble des questions possibles, $\mathcal{Y} = \{0, 1\}$(Faux/Vrai).
- Un modèle de génération des données : $\mathcal{D}$ une loi sur $\mathcal{X}$ représentant la distribution des questions que donne le professeur. Ainsi $\mathcal{D}(x)$ est la probabilité que la question x sorte. Cette distribution est inconnue pour l'étudiant.
- On modélise l'expérience E par un jeu de question/réponse $S = \{(x^{(1)}, y^{(1)}), \cdots, (x^{(m)}, y^{(m)})\}$ Toutes ces questions et leurs réponses ont été données par le professeur durant son cours.
- Résultat de l'apprentissage : un ensemble de règles de raisonnement $\mathcal{H} = \{h; h : \mathcal{X} \to \mathcal{Y}\}$, ainsi pour une question $x \in \mathcal{X}$, $h(x) = 0$ signifie que l'étudiant n'a pas su résoudre la question x par opposition.
- Un algorithme $A : \mathcal{P}(\mathcal{X} \times \mathcal{Y}) \to \mathcal{H}$ est une règle d'apprentissage. Ainsi si l'étudiant travaille sur une jeu de questions/réponses S A exhibe une règle de raisonnement h_S.
- L'indicateur de performance est le calcul de l'erreur commise par une règle de raisonnement.

2.4.3 Que faire en cas d'échec de l'apprentissage ?

Soit le scénario suivant : nous sommes confrontés à un problème d'apprentissage, on cherche à apprendre une distribution $\mathcal{D}$. Nous choisissons un algorithme A et une classe d'hypothèses $\mathcal{H}$, puis à l'aide d'un jeu de données S on produit un prédicteur h_S, mais ce dernier commet une erreur importante sur un jeu test T, que faire ? Plusieurs actions peuvent être tentées :

— Obtenir un jeu d'apprentissage plus important.
— Élargir la classe $\mathcal{H}$.
— Réduire la classe $\mathcal{H}$.
— Changer de classe d'hypothèse.
— Changer la méthode d'optimisation.

Chaque action doit être intentée en fonction de l'analyse de l'erreur. Rappelons-nous $L_{\mathcal{D}}(h_S) = \varepsilon_{app} + \varepsilon_{est}$. En pratique il est très difficile d'estimer ε_{app}, or on sait par le procédé de validation que $L_{\mathcal{D}}(h_S) \approx L_T(h_S)$ avec $T \sim \mathcal{D}^m$ Donc on utilise le schéma suivant : $L_T(h_S) = (L_T(h_S) - L_S(h_S)) + L_S(h_S)$. Intuitivement lorsque le terme $(L_T(h_S) - L_S(h_S))$ est grand on a sur-ajustement, et lorsque le terme $L_S(h_S)$ est grand, il y a sous-ajustement.

Illustrons le diagnostic de l'erreur dans le cadre de l'exemple précédent.

1. Supposons que l'étudiant décide de travailler sur un jeu de questions/réponses spécifique, à savoir une annale de l'année passée $S \sim \mathcal{D}^{20}$ (On travaille avec 20 questions pour fixer les idées). Convaincu que l'examen sera identique, il travaille sur l'ensemble des raisonnements vus en cours $\mathcal{F}$. Il en dégage une règle de raisonnement par l'opération d'optimisation suivante $h_S = min_{h \in F} L_S(h)$. À ce point l'étudiant connaît parfaitement le "sujet" S et parvient à ne commettre aucune erreur i.e $L_S(h_S) = 0$. Cependant, le professeur pour évaluer le degré de connaissance de ses élèves propose un examen $T \sim \mathcal{D}^{20}$ de n questions, avec $T \neq S$. Dans ces conditions, il y a fort à parier que l'étudiant sur-ajuste e.g $L_T(h_S) = 8/20$. Ici l'étudiant parvient à une note de $12/20$, alors qu'il avait une note de $20/20$ sur le jeu d'apprentissage ; c'est l'erreur d'estimation ε_{est} qui est importante.
   — Pour obtenir de meilleure performances on peut travailler sur un jeu d'apprentissage plus important, pour mieux saisir les nuances de $\mathcal{D}$.
   — La classe $\mathcal{H}$ est peut-être trop large, si l'étudiant apprend à résoudre des questions de détails, il a des chances de parvenir à une règle de raisonnement h_S trop complexe et ainsi peu robuste à la généralisation. Une solution est de réduire la classe $\mathcal{H}$ ou de régulariser (section suivante)
2. Si l'étudiant travaille sur le même jeu S, mais décide de ne se concentrer que sur les raisonnements les plus généraux. Alors la classe de règles de raisonnements $\mathcal{H}$ considérée sera plus réduite. Après avoir travaillé sur S il dégage une règle h_S et parvient à ne commettre que 8 erreurs sur 20 questions $L_S(h_S) = 8/20$. Le jour de l'examen il commet 10 erreurs et à la moyenne i.e $L_T(h_S) = 10/20$. Ici l'erreur d'estimation n'est pas très importante, l'étudiant sous-ajuste.
   — Pour obtenir de meilleures performances, augmenter le jeu d'apprentissage ne sera pas d'un grand secours, il faudrait plutôt élargir la classe $\mathcal{H}$ pour envisager plus de complexité.

En pratique $L_S(h_S)$ n'est pas un bon estimateur de ε_{app}, on peut avoir une erreur empirique élevée et une erreur d'approximation faible, dans ce cas c'est la méthode d'optimisation qui est peu efficace et qui devrait être changé ou amélioré. En particulier si le problème d'optimisation n'est pas convexe, le choix de la méthode de minimisation peut avoir un impact crucial.

2.4.4 Régulariser

La régularisation est une heuristique très utilisée en pratique pour réduire le surajustement, on l'applique donc quand la différence entre l'erreur commise par un prédicteur h sur un jeu d'apprentissage S est très inférieure à l'erreur commise sur un jeu d'essai T, i.e $L_S(h_S) \ll L_T(h_S)$. Comme on l'a vu au chapitre 1, il est possible en théorie de diminuer l'erreur d'estimation en réduisant la classe d'hypothèse $\mathcal{H}$. Cependant cette vision est simpliste et peu pratique, alternativement on peut modifier un algorithme en introduisant une préférence pour certaines hypothèses en introduisant une contrainte. La question est comment mettre en place cette contrainte ? En introduisant un terme régularisateur.

Exemple 2.5. Dégradation des pondérations
Dans les algorithmes de régression linéaire et logistique nous avions mis en avant des problèmes de minimisation, (équations 2.1 et 2.2) nous allons introduire un nouveau terme à ces expressions.

— Régression logistique : $h_S = \min_{w\in\mathbb{R}^d} \frac{1}{m} \cdot \sum_{i=1}^{m} l(x^{(i)}, y^{(i)}, h_w) + \lambda||w||_2$

— Régression linéaire : $h_S = \min_{w\in\mathbb{R}^d, b\in\mathbb{R}} \frac{1}{m} \sum_{i=1}^{m} l(x^{(i)}, y^{(i)}, h_{w,b}) + \lambda||w||_2$
Avec λ un hyper-paramètre.

Ainsi on impose à l'algorithme d'expliquer les données avec une hypothèse plus simple. Nous discuterons du rôle des hyper-paramètres à la fin du chapitre. La recherche de simplicité est due au principe heuristique et philosophique du "rasoir d'Ockham" autrement appelé principe de parcimonie que l'on peut énoncer ainsi :

"Les hypothèses suffisantes les plus simples sont les plus vraisemblables"

Ce principe bien que général trouve une justification mathématiques dans le cadre de l'apprentissage statistique voir le chapitre 7 de [1]. On peut informellement interpréter la complexité ou la simplicité d'une hypothèse par ses degrés de liberté.

Exemple 2.6. Appliquons la technique de dégradation des pondérations au problème suivant :
Soit $f(x) = sin(\pi x)$ avec $\mathcal{X} = [-1, 1]$, $\mathcal{U}$ la loi uniforme sur $\mathcal{X}$. Considérons un jeu de deux points sélectionnés uniformément sur $\mathcal{X}$; $S = \{x_1, x_2\} \sim \mathcal{U}^2$ et $\mathcal{H}$ l'ensemble des fonctions linéaires. Nous voulons expliquer la fonction cible f à l'aide de plusieurs jeux de données S, i.e pour chaque couple de point on trace une fonction linéaire. Dans la figure ci dessus on compare l'ensemble des courbes tracé avec une régression linéaire (gauche) et une régression linéaire régularisé (droite).

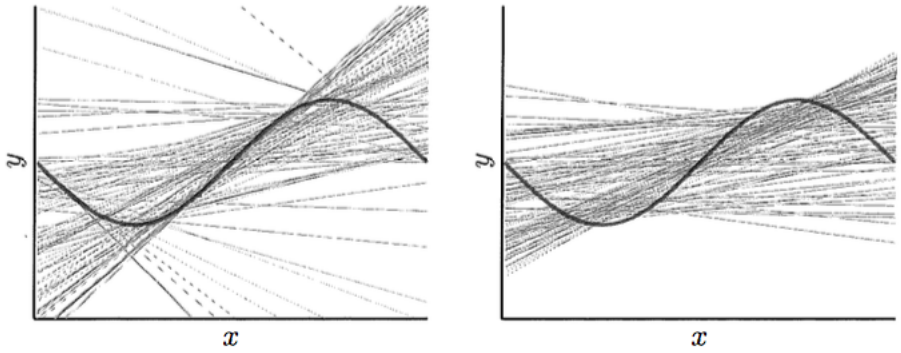

FIGURE 2.3 – Apprentissage standard vs régularisé

On constate que la version régularisée varie moins car ici les poids $w \in \mathbb{R}^2$ ont été choisis de sorte à minimiser $||w||_2$. L'exemple a été tiré de [6]

Le régularisateur vu comme stabilisateur
On peut également interpréter le régularisateur comme le stabilisateur d'un algorithme d'apprentissage. Intuitivement une règle d'apprentissage est stable si un petit changement dans les données d'entrée ne modifie que peu le résultat de sortie. Dans ce contexte, la dégradation des pondérations présentée plus haut introduit le stabilisateur de Thikonov. On peut montrer que si la fonction coût est convexe et lipschitzienne ou convexe dérivable et positive alors la fonction coût régularisée est fortement convexe. En particulier la règle de minimisation du risque régularisé (MRR) est stable, voir [1] chapitre 13.

Note sur les hyper-paramètres

Dans l'exemple de régression linéaire avec régularisation, il y a au total quatre paramètres : $w \in \mathbb{R}^d; b, \alpha, \lambda \in \mathbb{R}$, avec w et b des paramètres d'apprentissage, en effet, ils sont utilisés pour expliquer les données et nous permettre d'évaluer la distribution inconnue $\mathcal{D}$. Par contre α et λ sont des hyper-paramètres (α concerne la méthode du gradient, et λ module la régularisation) dans le sens suivant : ils ne servent pas à expliquer pas les données. Dans ce contexte, comment trouver les paramètres et hyper-paramètres ?
La procédure clé est la validation vu plus haut, on partitionne le jeu de données initial en trois jeux : un jeu d'apprentissage S, un jeu de validation V, et un jeu test T.

- Le jeu S sert à apprendre les paramètres $w \in \mathbb{R}^d$ et $b \in \mathbb{R}$ et à obtenir une hypothèse $h_{w,b}$
- Le jeu V sert à tester $L_V(h_{w,b})$ en faisant varier α dans la méthode d'optimisation, et λ pour la régularization.
- Le jeu test T sert à évaluer l'erreur réelle commis par notre algorithme d'apprentissage.

Le lecteur peut se demander à juste titre pourquoi ne pas se contenter d'un jeu d'apprentissage S et d'un jeu test T. Le jeu de validation faisant tampon est nécessaire, car sinon les hyper-paramètres seraient influencés par les données de T et $L_T(h_{w,b})$ ne serait plus une bonne estimation de l'erreur réelle $L_{\mathcal{D}}(h_{w,b})$. Ainsi, on utilise le jeu de validation V pour moduler les hyper-paramètres, mais lorsque l'on veut estimer l'erreur réelle, il faut évaluer l'algorithme sur des données qu'il n'a jamais "vu", c'est le seul moyen de s'assurer qu'il généralise bien.

Note bibliographique Pour rédiger ce chapitre nous avons utilisé plusieurs références, [1], [6], [5] qui présentent bien l'apprentissage automatique dans ses aspects théoriques et pratiques, nous avons fait le choix d'être le plus succinct possible pour rester dans le cadre d'une introduction à la discipline, un ouvrage complet sur les réseaux de neurones [7] propose également une introduction à l'apprentissage automatique au chapitre 5.

2.5 Schéma de synthèse

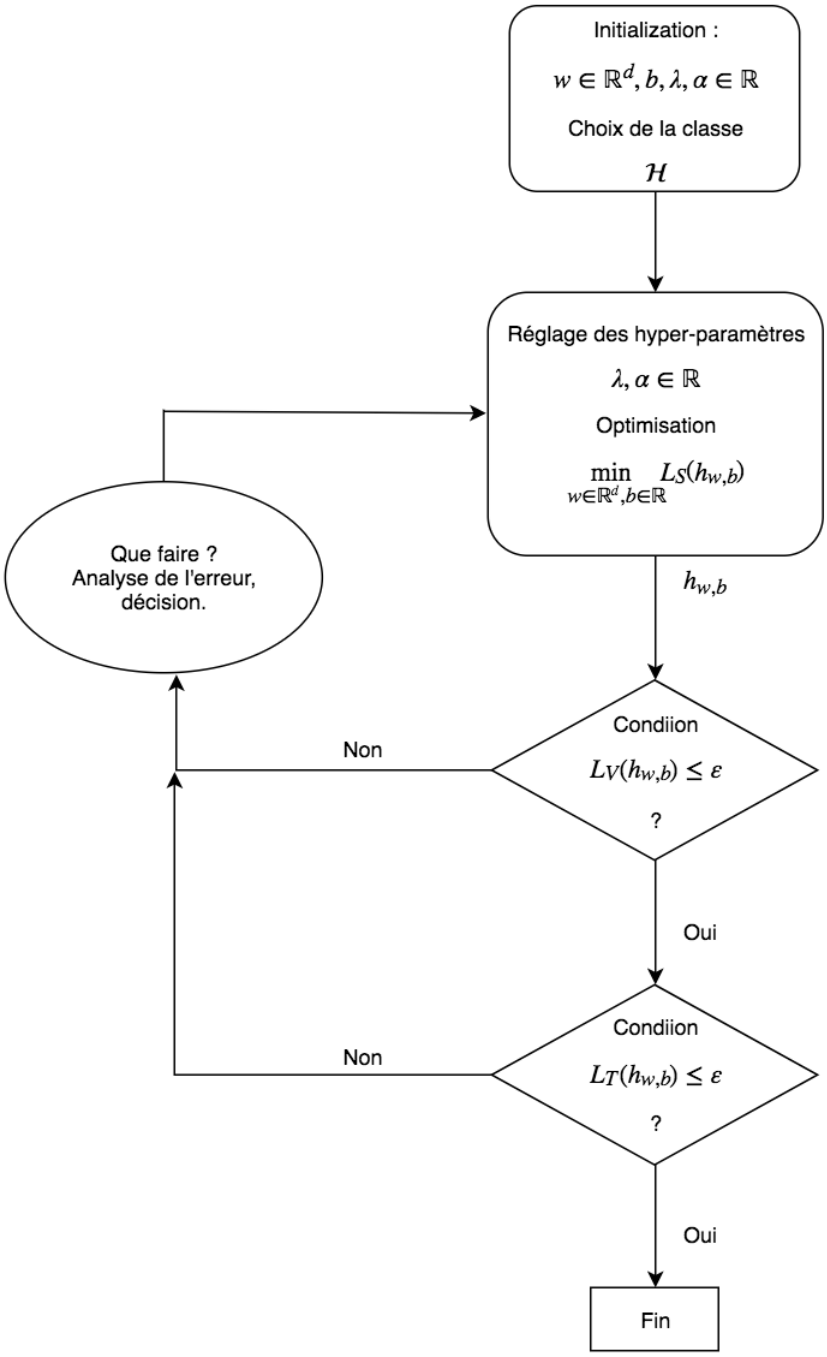

FIGURE 2.4 – Apprentissage automatique

Chapitre Trois

Réseaux de neurones artificiels

Sommaire

2.1 Contexte et définition 19
2.2 L'apprentissage supervisé 20
2.3 Méthode d'optimisation : l'algorithme du gradient 24
2.4 Généralisation 25
2.5 Schéma de synthèse 32

3.1 Introduction et représentation

La performance des algorithmes d'apprentissage dépend beaucoup des représentations que l'on se fait des données. En effet, aujourd'hui, nous pouvons aisément effectuer des opérations arithmétiques avec des chiffres arabes, alors qu'il était bien plus difficile de faire de même avec des chiffres romains par exemple. Ainsi, la résolution de problème d'apprentissage supervisé peut être grandement facilitée si l'on a une bonne représentation des données.

Cependant, pour beaucoup de tâches, il est difficile de savoir quelles caractéristiques sont pertinentes. Pour illustrer cette difficulté prenons la tâche de reconnaissance d'un véhicule sur une photo. Une photo n'est rien d'autre qu'un jeu de pixels donc une suite de chiffres pour un ordinateur. Nous avons appris à reconnaître des caractéristiques de haut niveau : roues, carrosserie, marques, allure générale. Ainsi, il nous est naturel de distinguer une voiture d'un arbre ou d'un rhinocéros. Pourtant, tous ces objets sont des suites de nombres pour la machine. Comment faire ?

Considérons les modèles présentés précédemment, la régression linéaire et la régression logistique, ces modèles sont intéressant, car l'apprentissage se déroule en minimisant une fonction convexe, et ce faisant offrent de bonnes propriétés de convergence, d'unicité du minimum etc. Néanmoins, la plupart des problèmes du monde réel ne peuvent pas être résolus uniquement avec des prédicteurs linéaire. En effet, la capacité de représentation de tels modèles est insuffisante. Comment faire dans ce cas ? Nous allons voir deux options.

— Dans un premier temps on peut chercher à la main une transformation non-linéaire pour expliquer les données en cas d'échec des modèles linéaire. Supposons que nous ayons accès à un jeu de données brutes voir Figure 3.1. Si nous cherchons à séparer les données avec une droite, nous échouerons. Par contre si nous effectuons une

transformation aux données, en l'occurrence en passant des coordonnées cartésiennes aux coordonnées polaires, on obtient une chance de séparer les données par une droite. Cependant, dans la réalité, les données auxquelles les chercheurs font face sont souvent multidimensionnelles et la recherche de représentations pertinentes est souvent longue et fastidieuse.

— La deuxième option est d'apprendre directement les représentations, et c'est là que les réseaux de neurones se sont révélés d'une étonnante efficacité. En effet, un réseau de neurones est capable d'apprendre des représentations complexes directement à partir des données.

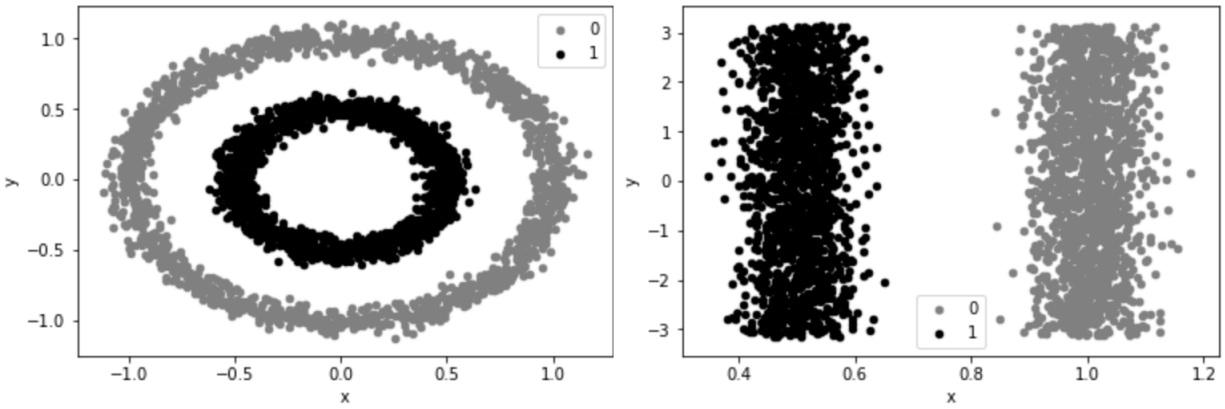

FIGURE 3.1 – Changement de coordonnées

3.2 Théorème d'approximation universelle

Comment font les réseaux de neurones pour apprendre des représentations complexes ? En partie car ils sont construits avec une classe de fonction dense dans l'espace des fonctions continues sur un compact $K \subset \mathbb{R}^d$. Ce résultat à été montré indépendamment par plusieurs auteurs. Nous avons choisi de présenter l'approche de Cygenco 1989 voir [8].

3.2.1 Résultats préliminaires

Définition 3.1. Soit $(E, \mathcal{B})$ un espace mesuré, une mesure signée μ est une application définie de $\mathcal{B}$ dans $\mathbb{R}$, σ-additive. En l'occurence :

— $\mu(\emptyset) = 0$

— $\forall (A_n)_{n>0} \subset \mathcal{B}$ tel que $A_n \cap A_m = \emptyset$ si $m \neq n$, on a $\mu(\bigcup_{n>0} A_n) = \sum_{n>0} \mu(A_n)$

Théorème 3.1. *(Représentation de Riez-Markov)*
Pour toute I une forme linéaire continue définie sur $C_0(\mathbb{R}^d)$ il existe une unique mesure signée μ telle que $\forall f \in C_0(\mathbb{R}^d)$, $I(f) = \int_{\mathbb{R}^d} f(x) d\mu(x)$.

Dans la suite on note $I_d = [0,1]^d$ le cube de dimension d, $C(I_d)$ l'ensemble des fonctions continues sur I_d, $||.||$ la norme de la convergence uniforme. $M(I_d)$ l'ensemble des mesures

de Borel finies et signées sur I_d. Notre objectif est de montrer dans quelles conditions les sommes de la forme :

$$g(x) = \sum_{i=1}^{n} a_i \cdot \sigma(< y_i, x > +b_i)$$

sont dense dans $C(I_d)$.

3.2.2 Résultat principal

Commençons par quelques définitions :

Définition 3.2. On appelle sigmoïde une fonction σ définie et à valeur dans $\mathbb{R}$ vérifiant :

$$\lim_{x \to -\infty} \sigma(x) = 0, \ \lim_{x \to \infty} \sigma(x) = 1$$

Définition 3.3. On dit qu'une fonction f est discriminante pour une mesure $\mu \in M(I_d)$, si

$$\forall y \in \mathbb{R}^d, b \in \mathbb{R}, \int_{I_d} f(< y, x > +b) d\mu(x) = 0 \Rightarrow \mu = 0$$

Lemme 3.1. *Soit σ une sigmoïde bornée continue mesurable, alors elle est discriminante. En particulier les sigmoïdes continues sont discriminantes.*

Théorème 3.2. *Soit σ une application continue discriminante, alors les fonctions de la forme*

$$g(x) = \sum_{i=1}^{n} a_i \cdot \sigma(< y_i, x > +b_i) \tag{3.1}$$

sont denses dans $C(I_d)$.

Démonstration. Posons G l'ensemble des fonctions définies par l'équation (3.1), on remarque que c'est un sous espace vectoriel de $C(I_d)$.
Supposons que $\overline{G} \neq C(I_d)$, alors par un corollaire du théorème de Hann-Banach géométrique $\exists L$ une forme linéaire continue non identiquement nulle du dual topologique de $C(I_d)$ telle que $\forall g \in G \, , L(g) = 0$. Or d'après le théorème de représentation de Riez-Markov

$$\exists ! \mu \in M(I_d) \text{ telle que } \forall f \in C(I_d), \ L(f) = \int_{I_d} f(x) d\mu(x)$$

Or on a $\forall g \in G \, , L(g) = \sum_{i=1}^{n} \int_{I_d} a_i \cdot \sigma(< y_i, x > +b_i) d\mu(x) = 0$.
Donc par le fait que σ est discriminante on en déduit que $\mu = 0$. Par conséquent $\forall f \in C(I_d), L(f) = 0$, on en déduit que L est identiquement nulle. Conclusion il y a absurdité et $\overline{G} = C(I_d)$. □

Corollaire 3.1. *Soit σ une sigmoïde continue alors, les sommes de la forme :*

$$g(x) = \sum_{i=1}^{n} a_i \cdot \sigma(< y_i, x > +b_i) \tag{3.2}$$

sont denses dans $C(I_d)$.

Ceci étant posé, introduisons un neurone formellement.

3.3 Neurone artificiel

3.3.1 Apprentissage à partir d'un exemple

Propagation

L'apprentissage se fait à partir de deux procédures, la *propagation* et la *rétropropagation*. Considérons un problème de classification binaire n'ayant qu'un exemple, soit la mise en forme suivante :

— $\mathcal{X} \subset \mathbb{R}^d$, $\mathcal{Y} = \{0, 1\}$

— $S = \{(x, y)\} \subset \mathcal{X} \times \mathcal{Y}$

Dans cette configuration un neurone ne se distingue d'une régression logistique que dans la représentation sous forme de graphe, cela nous permettant d'introduire au passage un vocabulaire spécifique.

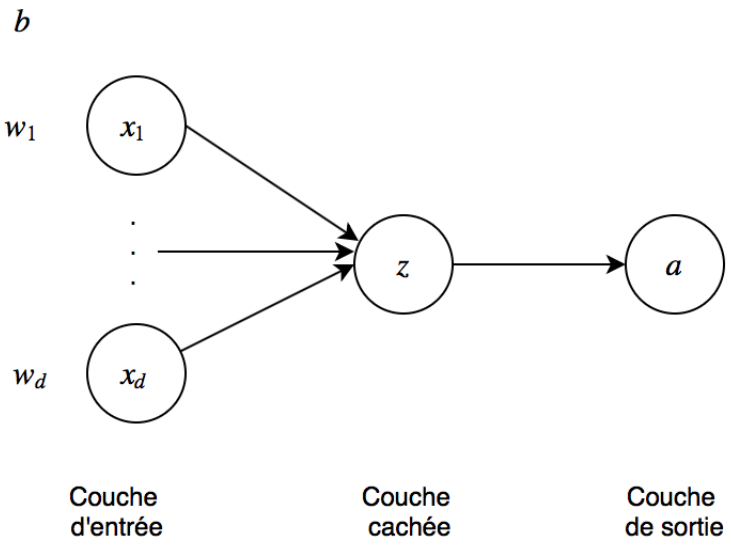

FIGURE 3.2 – Propagation

Ainsi dans tout réseaux de neurones, il y a une couche d'entrée correspondant aux données une couche cachée et une couche de sortie. La phase de calcul de a s'appelle la *propagation* et est déterminée par les équations suivantes :

$$z = \sum_{i=1}^{d} w_i x_i + b =< w, x > +b \text{ , avec } w \in \mathbb{R}^d, b \in \mathbb{R} \tag{3.3}$$

$$a = \sigma(z) \text{ , avec } \sigma \text{ la fonction sigmoïde} \tag{3.4}$$

Une fois que la prédiction est effectué (a est une prédiction de la classe de y) on calcule la fonction coût pour évaluer l'algorithme.

$$l(a, y) = -[y \cdot log(a) + (1 - y) \cdot log(1 - a)]$$

Rétropropagation

L'apprentissage se fait par la mise à jour des paramètres w, b par l'algorithme du gradient qui dans le cadre des réseaux de neurones s'appelle la *rétropopagation*. On cherche à minimiser la fonction coût l, ce réseau simple à $d+1$ paramètres qui ne sont pas apparent dans l'expression finale. La rétropropagation consiste à calculer la dérivée partielle par rapport à chaque variable en sens inverse jusqu'à arriver aux paramètres à apprendre. La règle principale utilisée est la règle de dérivation des fonctions composés

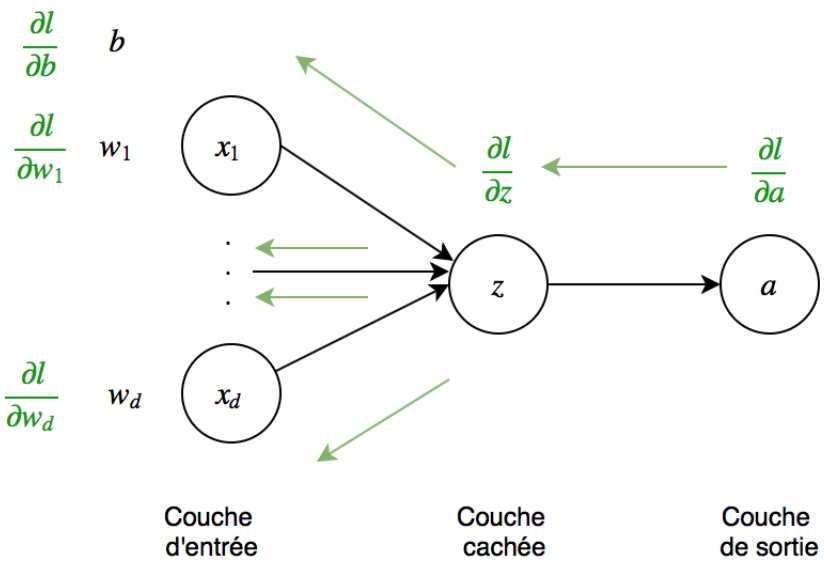

FIGURE 3.3 – Rétropropagation

Ainsi les dérivées partielles sont données par les équations suivantes :

$$\frac{\partial l}{\partial a} = \frac{-y}{a} + \frac{1-y}{1-a} \tag{3.5}$$

$$\frac{\partial l}{\partial z} = \frac{\partial l}{\partial a} \cdot \frac{\partial a}{\partial z} = a - y \tag{3.6}$$

$$\frac{\partial l}{\partial b} = \frac{\partial a}{\partial z} \cdot \frac{\partial z}{\partial b} = a - y \tag{3.7}$$

$$\frac{\partial l}{\partial w_i} = \frac{\partial a}{\partial z} \cdot \frac{\partial z}{\partial w_i} = x_i \cdot (a - y) \tag{3.8}$$

Remarque 3.1. Toutes les valeurs déterminées par la propagation et la rétropropagation sont numériques.

Dans la suite comme la différentiation se fera toujours par rapport à l la fonction coût : on notera $\partial a = \frac{\partial l}{\partial a}, \partial z = \frac{\partial l}{\partial z}, \partial w = \frac{\partial l}{\partial w}, \partial b = \frac{\partial l}{\partial b}$. On peut synthétiser tout cela avec l'algorithme suivant :

Data : $x \in \mathbb{R}^d, w \in \mathbb{R}^d, b \in \mathbb{R}, y \in \{0,1\}, l \in \mathbb{R}, \partial w \in \mathbb{R}^d, \partial b \in \mathbb{R}, \alpha \in \mathbb{R}$;
initialization $w = random, b = 0, \partial w = 0, \partial b = 0, l = 0, \alpha = 0.1$;
while $\|l\|$ *trop grand* **do**
  $z =< w, x > +b$;
  $a = \sigma(z)$;
  $l = -(y \cdot log(a) + (1-y) \cdot log(1-a))$;
  $\partial z = a - y$;
  $\partial w = x\partial z$; % ∂z est in scalaire et $x \in \mathbb{R}^d$
  $\partial b = \partial z$;
  $w = w - \alpha \partial w$;
  $b = b - \alpha \partial b$;
end

Algorithme 2 : Apprentissage d'un neurone

Rappelons que α est un hyper paramètre pour indiquer de quelle mesure on se déplace vers le minimum à chaque itération. Par ailleurs w est initialisé aléatoirement, par principe heuristique voir [7].

Remarque 3.2. L'apprentissage avec un exemple n'a qu'une portée introductive, en réalité on travaille avec de nombreux exemples et les réseaux de neurones sont complexes. Cela nous amène à considérer la *vectorisation*. La *vectorisation* est simplement le fait de privilégier des notations matricielles, (d'où l'utilisation massive d'algèbre linéaire en apprentissage automatique) cela a deux intérêts :

- Les notations sont plus concises.
- Des opérations vectorielles efficaces (tirant profit du parrallélisme des processeurs) ont été implémentés (en C, C++, Fortran) et sont disponibles dans les languages haut niveau tels que Python, Matlab etc, sous forme de librairies. Ainsi, utiliser ces opérations à la place des boucles classiques qui tournerait séquentiellement est un énorme avantage, car l'apprentissage profond/réseaux de neurones a un coût élevé en temps de calcul.

Ainsi on a intérêt à toujours chercher à vectoriser un maximum nos équations et calculs, c'est ce que nous feront dans ce qui suit.

3.3.2 Apprentissage à partir de plusieurs exemples

Considérons toujours le problème de classification binaire évoqué plus haut, la mise en forme serait :

— $\mathcal{X} \subset \mathbb{R}^d$, $\mathcal{Y} = \{0, 1\}$

— $S = \{(x^{(1)}, y^{(1)}), \cdots, (x^{(m)}, y^{(m)})\} \subset \mathcal{X} \times \mathcal{Y}$

Notons :

$$X = \begin{bmatrix} | & | & | \\ x^{(1)} & \cdots & x^{(m)} \\ | & | & | \end{bmatrix} \in \mathbb{R}^{d \times m} \qquad Y = [y^{(1)} \cdots y^{(m)}] \in \mathbb{R}^{1 \times m}$$

$$W = \begin{bmatrix} | \\ w_i \\ | \end{bmatrix} \in \mathbb{R}^{d \times 1} \qquad b = [b \cdots b] \in \mathbb{R}^{1 \times m}$$

Notons que le paramètre b a été vectorisé. Les équations de la propagation deviennent :

$$Z = W^T X + b \tag{3.9}$$

$$A = \sigma(Z) \tag{3.10}$$

$$L(A, Y) = \frac{1}{m} \sum_{i=1}^{m} l(a^{(i)}, y^{(i)}) \tag{3.11}$$

Les équations de la rétropropagation deviennent :

$$\partial Z = [a^{(1)} - y^{(1)} \cdots a^{(m)} - y^{(m)}] = A - Y \tag{3.12}$$

$$\partial W = \frac{1}{m} \Big(\begin{bmatrix} | \\ x^{(1)} \\ | \end{bmatrix} \partial z^{(1)} + \cdots + \begin{bmatrix} | \\ x^{(m)} \\ | \end{bmatrix} \partial z^{(m)} \Big) = \frac{1}{m} X \partial Z^T \tag{3.13}$$

$$\partial b = \frac{1}{m} (\partial z^{(1)} + \cdots + \partial z^{(m)}) = \frac{1}{m} \sum_{i=1}^{m} \partial z^{(i)} \tag{3.14}$$

Dans les équations (3.12) et (3.13) on moyenne les valeurs des dérivées partielles obtenues pour que l'apprentissage des paramètres tiennent compte de chaque exemples du jeu d'entraînement.

3.4 Réseau de neurones entièrement connectés

Un *réseau de neurone entièrement connecté* est paramétré par son nombre de couches (on compte uniquement les couches cachées), le nombre de neurones par couche, et par ses *fonctions d'activations.*

Commençons par définir ces nouveaux termes.

— Il existe plusieurs *architectures* de réseaux de neurones : les réseaux convolutifs, les réseaux récurrents, les réseaux antagonistes génératifs, etc. Le réseau le plus simple, mais fondamental qui permet ensuite d'appréhender les autres architectures est le réseau de neurones entièrement connectés.
— Une fonction d'activation est une non-linéarité, par exemple précédemment la fonction d'activation était σ. Mais il en existe d'autres comme les fonctions tangente hyperbolique, softmax, etc. Pour d'autres exemples voir [7].

3.4.1 Notations et exemple

L'un des principaux chalenges lorsque l'on décrit des réseaux de neurones est d'adopter un bon système de notations, celui que nous introduisons est fortement inspiré du cours d'Andrew Ng [9]. La définition suivante sera utile dans la description des équations de la rétropropagation

Définition 3.4. On appelle produit matriciel de Hadamard une opération binaire que l'on note $*$ entre deux matrices de même dimension. Soit $A, B \in \mathbb{R}^{n\times m}$ alors $A * B \in \mathbb{R}^{n\times m}$ et $(A * B)_{i,j} = A_{i,j} \times B_{i,j}$.

On note :
— L le nombre de couches.
— $n^{[l]}$ le nombre de neurone de la couche l.
— $W^{[l]}$ et $b^{[l]}$ les paramètres de la couche l.
— $Z^{[l]}$ la transformation linéaire obtenu de la couche $l-1$.
— $A^{[l]}$ l'activation obtenu à partir de $Z^{[l]}$.

Considérons le problème de classification binaire avec plusieurs exemples présenté en section (3.3.2). On posera $A^{[0]} = X \in \mathbb{R}^{d\times m}$ et $Y \in \mathbb{R}^{1\times m}$ comme précédemment. Présentons un réseau avec $L = 2$, $n^{[1]} = d$, $n^{[2]} = 1$, une fonction d'activation générale g pour la première couche, et la fonction sigmoïde σ pour la deuxième couche.

Remarque 3.3. On a opté pour la fonction sigmoïde en sortie de deuxième couche car on traite un problème de classification binaire. Au final en entrant un exemple x dans le réseau, on veut savoir si la probabilité qu'il appartienne à la classe 0 est supérieure ou inférieure à 0.5 revoir la figure 2.1.

Alors pour que les équations soient valides il faut nécessairement que :
— $W^{[1]} \in \mathbb{R}^{d\times d}$ et $b^{[1]} \in \mathbb{R}^{d\times m}$
— $W^{[2]} \in \mathbb{R}^{d\times 1}$ et $b^{[2]} \in \mathbb{R}^{1\times m}$
Les équations de la propagations sont alors :

$$Z^{[1]} = (W^{[1]})^T A^{[0]} + b^{[1]} \tag{3.15}$$
$$A^{[1]} = g(Z^{[1]}) \tag{3.16}$$
$$Z^{[2]} = (W^{[2]})^T A^{[1]} + b^{[2]} \tag{3.17}$$
$$A^{[2]} = g(Z^{[2]}) \tag{3.18}$$

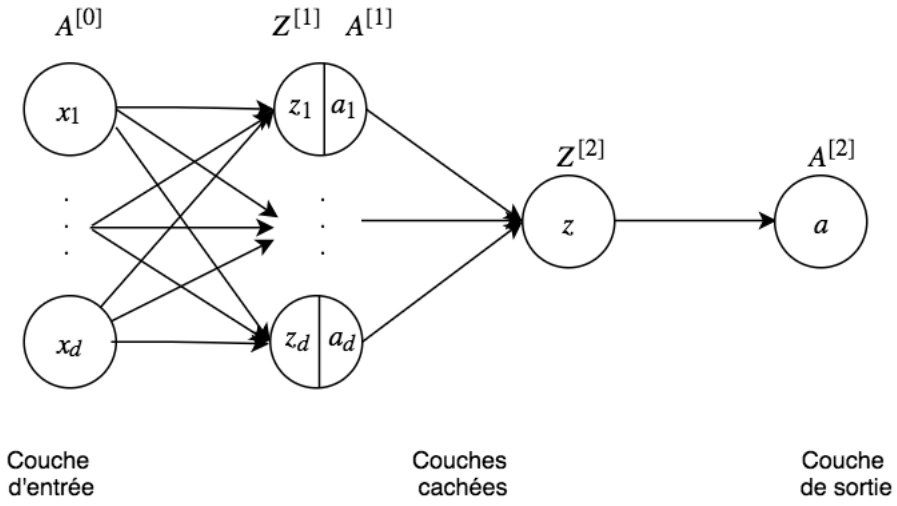

FIGURE 3.4 – Réseau de neurones à deux couches

Les équations de la rétropropagation :

$$\partial Z^{[2]} = A^{[2]} - Y \tag{3.19}$$

$$\partial W^{[2]} = \frac{1}{m} A^{[1]} (\partial Z^{[2]})^T \tag{3.20}$$

$$\partial b^{[2]} = \frac{1}{m} \sum_{i=1}^{m} \partial z_i^{[2]} \tag{3.21}$$

$$\partial Z^{[1]} = (W^{[2]})^T \partial Z^{[2]} * g'(Z^{[1]}) \tag{3.22}$$

$$\partial W^{[1]} = \frac{1}{m} A^{[0]} (\partial Z^{[1]})^T \tag{3.23}$$

$$\partial b^{[1]} = \frac{1}{m} \sum_{i=1}^{m} \partial z_i^{[1]} \tag{3.24}$$

La manière d'obtenir ces équations est identiques à ce que l'on a fait dans les sections précédentes. L'algorithme se déduit des équations.

3.4.2 Une discipline empirique

On a vu dans la section (3.2) qu'une combinaison linéaire de sigmoïdes peut approcher arbitrairement n'importe qu'elle fonction continue. C'est l'une des raisons qui font la force des réseaux de neurones. Cependant, la théorie n'est pas constructive. On ne sait pas combien de neurones sont nécessaires pour obtenir une bonne approximation. Le théorème tout se paye énoncé au chapitre 1 indique qu'il n'y a pas d'algorithme universel et que par conséquent la démarche d'apprentissage doit se baser sur une expérience/connaissance préalable de la tâche à apprendre. Les réseaux de neurones sont en fait la formulation d'une hypothèse autour de la fonction à apprendre, on suppose que cette dernière est une composition de plusieurs fonctions plus simple. En terme de représentation, cela suggère que l'on cherche un ensemble de facteurs de variation. Un réseau de neurones capture donc un

jeu de variation s'expliquant en termes de facteurs de variation plus simples jusqu'à revenir aux données brutes. La conséquence de cela est qu'un ensemble de réseaux de neurones spécialisés se sont "révélés" d'une grande efficacité pour apprendre des tâches précises.
Par exemple les réseaux de neurones convolutifs sont un type de réseaux spécialisés pour traiter des données ayant une forme de grille. Des séries temporelles admettant une grille d'une dimension, les images admettant des grilles de deux dimensions. Les réseaux convolutifs utilisent l'opération de convolution à travers les couches au lieu du produit matriciel classique. Les résultats sont spectaculaires. En 2017 une équipe de Stanford a entraîné un réseau convolutif pour apprendre à diagnostiquer la pneumonie grâce à des images obtenue aux rayons X à partir de radiographies et à obtenir de meilleures performances que des radiologues en activité voir [10].

Note bibliographique Ce chapitre est très inspiré de plusieurs sources notamment [8] pour le théorème d'approximation universelle, du livre de Yoshua Bengio, Ian Goodfellow et Aaron Courville [7] et du cours d'Andrew Ng [9].

Conclusion générale

Pour conclure, soulignons que l'apprentissage automatique et l'apprentissage profond sont des domaines de recherche très actifs. Le sous-domaine que nous avons présenté, l'apprentissage supervisé est celui qui marche le mieux, (qui a le plus de retombées financières) mais il existe de nombreux défis pour apprendre à partir de données non étiquetées (apprentissage non supervisé), ou apprendre en attendant une réponse de l'environnement (apprentissage par renforcement).

Par ailleurs ces disciplines sont appelées à prendre de plus en plus d'importance, en effet nous vivons dans une ère de production massive de données, que ce soit par les entreprises privées, les gouvernements, les centres de recherche, etc. Par ailleurs jamais la puissance de calcul n'a été aussi abordable, en effet, nos smartphones sont plus puissants que les super ordinateurs dont disposait la NASA dans sa mission Apollo qui envoya l'homme sur la Lune. Aujourd'hui, il est possible de louer de la puissance de calcul directement sur le "cloud". Il est difficile pour le Sénégal de devenir (par exemple) une puissance nucléaire juste avec la connaissance théorique, car cela demande de pouvoir développer des technologies complexes et onéreuses. Cependant l'apprentissage automatique ne demande que très peu de moyen à mettre en oeuvre pour être compris en théorie et en pratique. Cela dit, développer une expertise locale en apprentissage automatique permettrait de se positionner sur plusieurs marchés porteurs du XXIe siècle : les voitures autonomes, assistants digitaux, traitement de données massives "big data", robotique industrielle, etc. La discipline étant directement liée aux données, innover académiquement dans nos laboratoires pourrait générer des partenariats privés et permettre l'exportation d'un savoir-faire à haute valeur ajoutée dans les économies dites développées. Capter des flux de devises fortes soutenir l'émergence d'un écosystème de start-up et de PME spécialisées et enfin générer de la prospérité. Au-delà des conséquences de la création de champions forts sur les marchés extérieurs, le développement de ces techniques pourrait grandement réduire les inégalités. Il y a une pénurie de médecins dans certaines régions. Apprendre à une intelligence artificielle à diagnostiquer les maladies les plus communes et pouvant être accessible via un smartphone dans n'importe quel village aurait un grand impact sur le bien-être des populations en zone rurale. Ce manuscrit ainsi que toutes les considérations présentées ici ont pour but de sensibiliser sur le potentiel de cette discipline.

Bibliographie

[1] S Shalev-Shwartz, S Ben-David, *Understanding machine learning : From theory to algorithms*, Cambridge university press, 2014

[2] Sylvain Arlot, *Fondamentaux de l'apprentissage statistique*, hal-01485506, 2017

[3] Gérald GAVIN, *Etude du modèle d'apprentissage Probablement Approximativement Correct (PAC)*, Thèse de doctorat en informatique, Université Lumière - Lyon 2, Laboratoire ERIC, 2001

[4] Andrew Ng, *The State of AI*, MIT Technology Review, 2017

[5] Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar *Foundations of machine learning*, The MIT Press Cambridge, Massachusetts London, England, 2012

[6] Abu-Mostafa Y. S, Magdon-Ismail M, H Lin, *Learning from data : a short course*, AMLBook.com, 2012

[7] Ian Goodfellow and Yoshua Bengio and Aaron Courville, *Deep Learning*, MIT Press, 2016

[8] G Cybenko, *Approximation by superpositions of a sigmoidal function*, Mathematics of control, signals and systems - Springer, 1989

[9] Andrew Ng, *Neural Networks and Deep Learning*, deeplearning.ai - Coursera - Massive Open Online Course

[10] Rajpurkar, Pranav ; Irvin, Jeremy ; Zhu, Kaylie ; Yang, Brandon ; Mehta, Hershel ; Duan, Tony ; Ding, Daisy ; Bagul, Aarti ; Langlotz, Curtis ; Shpanskaya, Katie ; Lungren, Matthew P. ; Ng, Andrew Y. *CheXNet : Radiologist-Level Pneumonia Detection on Chest X-Rays with Deep Learning*, ARXIV, 2017