Chapitre 1

Modèles d'Apprentissage Artificiel

La cognition humaine repose sur cette propension unique à extraire des connaissances générales à partir de très peu d'exemples. Même si les animaux peuvent discriminer les objets sur la base d'indices perceptuels, seuls les humains et peut-être quelques autres mammifères sont capables d'apprendre des connaissances qui sous-tendent la généralisation de propriétés. Considérons par exemple comment un enfant peut saisir le sens d'un nom commun tel que "cheval". étant donné quelques instances de chevaux montrés par ses parents, l'enfant est capable d'accomplir un bond inductif qui dépasse largement les données observées : il pourra ensuite prédire si une nouvelle entité est un cheval ou non, et fera vraisemblablement très peu d'erreurs, excepté les occasionnels poneys, ânes, ou zèbres. La capacité à généraliser à partir de peu de données est cruciale, non seulement pour apprendre la signification des mots, mais aussi pour apprendre les propriétés des objets, les relations de cause à effet, les règles sociales, et bien d'autres domaines de la vie courante (Woodmward et Needham, 2008).

Avec le potentiel qu'offre l'apprentissage humain, il n'est pas étonnant de constater que, dès les débuts de l'informatique, les chercheurs se sont demandés si les machines étaient capables d'apprendre. Le premier algorithme d'apprentissage a été inventé en 1958 par Rosenblatt. Cet algorithme, appelé *Perceptron*, apprend une fonction linéaire discriminante à partir d'une séquence d'exemples décrits par leurs attributs. Même si l'analyse discriminante avait été découverte auparavant (Fisher, 1930), le Perceptron était le premier algorithme capable de *généraliser*, c'est à dire d'induire un concept à partir d'un échantillon d'exemples, sans savoir à l'avance la distribution de probabilité sur tous les exemples. Depuis, la recherche en apprentissage a fait des progrès considérables, à la fois sur le plan expérimental, où de nouveaux algorithmes sont apparus pour apprendre des tâches de plus en plus complexes, et sur le plan théorique, où des modèles ont été inventés pour établir les fondements de l'apprentissage.

Une analyse détaillée des modèles d'apprentissage artificiel nécessiterait aujourd'hui tout un ouvrage. Après avoir introduit une architecture pour les agents apprenants, nous présenterons un cadre formel d'apprentissage *supervisé*, à partir duquel nous déclinerons trois modèles : *l'apprentissage exact*, *l'apprentissage enligne*, *l'apprentissage statistique*. Nous présenterons ensuite un cadre non supervisé d'apprentissage, appelé *apprentissage par renforcement*, dans lequel nous examinerons une adaptation de l'apprentissage statistique. Nous conclurons cette étude en pointant quelques thèmes de recherche actuels sur la modélisation de l'apprentissage.

0. par ANTOINE COURNUÉJOLS, Université de Paris-Sud, FRÉDÉRIC KORICHE, Université Montpellier II, LAURENT MICLET, Université Rennes I, et RICHARD NOCK, Université des Antilles et de la Guyane

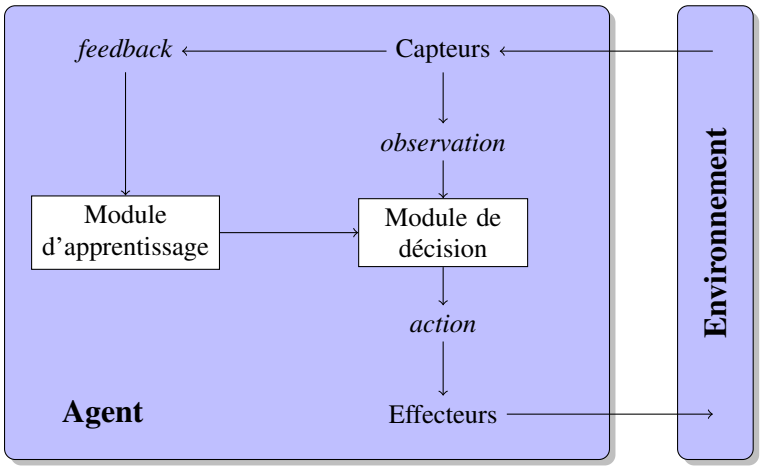

FIGURE 1: Une architecture simplifiée d'agent apprenant

1.1 Agents Apprenants

Comme le suggèrent Russell et Norvig (2003) ainsi que Pitrat (2009), la plupart des problèmes d'intelligence artificielle se modélisent naturellement à partir du paradigme orienté *agent*. Un agent est une entité qui perçoit et agit dans son environnement en utilisant respectivement ses capteurs et ses effecteurs. Par exemple, un agent robotique perçoit son environnement physique à partir de capteurs optiques, sonores ou inertiels, et agit dans son environnement en déclenchant diverses commandes. Un agent logiciel perçoit son environnement à partir des frappes du clavier, les mouvements de la souris, ou encore les messages du réseau, et agit dans son environnement en affichant sur l'écran, écrivant des fichiers, ou transmettant des messages par le réseau. Le comportement d'un agent est décrit par une fonction ou *stratégie* qui associe une action à un percept, ou plus généralement une séquence de percepts. Cette stratégie est implémentée par un programme appelé *module de décision*.

En se basant sur ce paradigme, un problème de décision est spécifié par un espace de percepts accessibles par l'agent, un ensemble d'actions que l'agent peut accomplir, et une *mesure de performance* qui associe à toute stratégie une quantité, que l'on appelle souvent *utilité*. Pour des tâches relativement simples comme les problèmes de classification, la performance de l'agent peut être mesurée en comptant le nombre de fois que l'agent s'est trompé de décision. Cependant, pour des tâches plus complexes comme les problèmes de décision séquentielle, chaque action peut avoir une incidence à long terme et donc la performance doit mesurer l'utilité de toute la séquence d'actions accomplies par l'agent en fonction des données perçues.

Nous disons qu'un agent *apprend* s'il améliore sa performance à résoudre le problème de décision à partir de l'expérience acquise sur son environnement. L'architecture d'un agent apprenant (figure 1) comprend, en plus du module de décision, un *module d'apprentissage* permettant à l'agent de modifier les paramètres du module de décision en fonction de l'expérience acquise. Le processus d'apprentissage peut être vu comme une séquence de cycles d'interaction entre l'agent et l'environnement. Durant chaque cycle, l'agent reçoit une *observation* et choisit une *action* en utilisant son module de décision. Après s'être engagé sur son action, l'agent reçoit un *feedback* lui permettant d'évaluer son action et donc de réviser sa stratégie.

Exemple 1. Pour illustrer ces notions sur un exemple concret, considérons la conception d'un filtre anti-spam "adaptatif". Le problème consiste à déterminer si chaque courriel entrant est un message utile à lire ou non. Même si le développement des filtres anti-spam a fait des progrès notables, la conception de filtres adaptatifs est un problème important puisque l'utilité d'un message est très relative d'un individu à l'autre (Chan et Lippmann, 2006). Pour des raisons de simplicité, nous imaginons ici chaque mot comme un attribut booléen, et représentons chaque message comme un vecteur booléen. Ainsi, l'espace des observations est l'ensemble $\{0,1\}^n$ où la dimension n peut atteindre 10^5 dans les applications réelles. Comme il s'agit d'un problème de classification binaire, l'ensemble des actions peut être donné par $\{-1,1\}$, avec la convention que -1 soit un message normal et $+1$ soit un spam.

Le protocole d'apprentissage est le suivant : à chaque étape $t = 1, 2, \ldots$, un courriel entrant est présenté à l'agent. A partir de son module de décision, l'agent prédit si ce message est un spam ou non ; si sa prédiction est positive, il déplace le message dans le dossier "spam". L'utilisatrice lui signale alors une erreur si elle déplace le message depuis le dossier "spam" vers la boîte de réception, ou inversement. La performance de l'agent est mesurée par le nombre d'erreurs de prédiction qu'il fait sur toute une série de courriels.

Intuitivement, la "difficulté" d'un problème d'apprentissage est caractérisée par deux sources de complexité. La première correspond au nombre de cycles durant lesquels la performance de l'agent reste sous-optimale pour la tâche de décision donnée. La seconde correspond aux ressources de calcul nécessaires durant chaque cycle à l'agent pour réviser sa stratégie et choisir une action. Par *modèle d'apprentissage*, nous entendons un cadre formel donnant une mesure de ces deux sources de complexité. Avant d'entrer dans les détails des modèles, il est utile de donner une indication de la façon dont les observations, les actions et le feedback peuvent influer sur la difficulté de l'apprentissage.

Observations. Dans la plupart des applications réelles, l'espace des observations accessibles à l'agent est immense, voire infini. Ainsi, pour apprendre rapidement un agent doit être capable d'*extrapoler*, c'est-à-dire inférer à partir de l'expérience acquise sur un nombre restreint d'observations perçues, une stratégie de décision pouvant s'appliquer sur éventail beaucoup plus large d'observations possibles. En plus du problème lié à la dimension des observations, les valeurs de certains attributs peuvent être imprécises, erronées, ou encore absentes. Dans ces environnements "partiellement" observables, l'agent doit être aussi capable d'*interpoler*, c'est-à-dire inférer à partir d'une situation incertaine un ensemble d'observations sur lesquelles il peut appliquer sa stratégie et combiner les résultats pour en dériver une action.

Actions. Un problème de décision est dit *unidimensionnel* si les actions sont des décisions simples pouvant être représentées comme des valeurs d'un domaine (fini ou infini). Il est dit *multidimensionnel* si les actions sont des décisions complexes représentées par des vecteurs dont chaque entrée est un composant de l'action. Dans certains problèmes de décision multidimensionnels, l'espace des décisions possède une structure combinatoire ; les décisions peuvent prendre la forme d'arbres, de graphes, ou encore d'hypergraphes. Le problème est alors dit *structurel*. Par exemple, l'alignement de mots depuis une phrase source (observation) vers une phrase cible (décision) en traduction automatique du langage (Matusov *et al.*, 2004), l'appariement de formes en reconnaissance d'images (Belongie *et al.*, 2002), sont des problèmes de classification structurelle considérés comme particulièrement difficiles en apprentissage.

Même des actions simples peuvent avoir un impact sur la difficulté de l'apprentissage selon la manière dont elles influent l'environnement. En reprenant la terminologie de Russell et Norvig (2003), un problème de décision est dit *épisodique* si, durant chaque cycle d'interaction, l'action choisie par l'agent n'a aucun effet sur l'observation suivante. Il est dit *séquentiel* si chaque action peut influer le cours des observations et donc avoir une conséquence à long terme. Par exemple, les problèmes de classification sont souvent épisodiques. En revanche, les problèmes de mouvement, les stratégies de placement financier, ou encore les jeux de plateau, sont des exemples caractéristiques de tâches de décision séquentielle.

Feedback. Le type de feedback définit le mode d'apprentissage. En apprentissage *supervisé*, le feedback correspond à l'action qu'aurait du choisir l'agent selon l'observation courante. Ainsi l'environnement est un superviseur qui corrige les erreurs de l'agent en fournissant la bonne "étiquette" à chaque observation (Kearns et Vazirani, 1994). En apprentissage *non supervisé*, l'agent ne reçoit aucun feedback. Au départ du processus d'apprentissage, l'agent démarre avec un lot d'observations non étiquetées et construit une représentation qui permet de structurer ces observations. Cette représentation peut ensuite être utilisée comme stratégie pour étiqueter de nouvelles observations selon la structure induite (Xu et Wunsch, 2008; Koller et Friedman, 2009). Il existe tout un éventail de modes d'apprentissage entre ces deux extrêmes. Par exemple, en apprentissage *semi-supervisé*, certaines observations sont associées avec des étiquettes, alors que d'autres sont sans étiquette ; ces dernières pouvant néanmoins aider l'agent à prédire la distribution des données (Chapelle *et al.*, 2006).

Enfin, dans l'apprentissage *par renforcement*, l'agent reçoit systématiquement un feedback de l'environnement, mais il est limité à un signal donnant une indication sur la qualité de sa décision (Sutton et Barto, 1998). Dans ce cadre s'ajoute une difficulté supplémentaire caractérisée par le fameux dilemme *exploration-exploitation* (Fel'dbaum, 1961). D'un coté, l'agent a besoin d'explorer son environnement pour découvrir l'effet de ses actions et mesurer la performance des stratégies envisageables ; de l'autre coté, l'agent a intérêt d'exploiter ses connaissances acquises afin d'appliquer les actions qu'il sait être performantes. Bien entendu, toute la difficulté est de trouver un bon compromis entre ces deux attitudes.

1.2 Apprentissage Supervisé

Après un tour d'horizon sur les problèmes d'apprentissage, nous allons à présent introduire un cadre plus formel pour l'apprentissage supervisé. Ce cadre sera exploité dans les trois prochaines sections pour définir des modèles particuliers d'apprentissage supervisé.

Un problème d'apprentissage supervisé comprend un espace $\mathcal{X}$ d'observations, appelées aussi *instances* ou *entrées*, et un espace $\mathcal{Y}$ d'actions ou *sorties*. Notons $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$. Tout élément $z \in \mathcal{Z}$ est appelée *exemple* et toute séquence d'exemples $\boldsymbol{z} = (z_1, \cdots, z_m)$ est appelée *ensemble d'entraînement*.[1] Dans ce chapitre, les entrées et sorties sont décrites comme des vecteurs de dimension non nulle et *finie*.

En plus des entrées et sorties, un problème d'apprentissage supervisé comprend généralement deux classes de fonctions de $\mathcal{X}$ dans $\mathcal{Y}$, la *classe d'hypothèses* $\mathcal{H}$ et la *classe cible* $\mathcal{H}^*$. Intuitivement, l'apprenant cherche à prédire le comportement de son environnement à partir de

1. Techniquement, un ensemble d'entraînement est donc une "liste" d'exemples.

stratégies h sélectionnées dans $\mathcal{H}$. La classe cible $\mathcal{H}^*$ sert de référence pour mesurer la performance des hypothèses ; l'objectif est de trouver par entraînement une hypothèse $h \in \mathcal{H}$ dont la performance est proche de la "meilleure" fonction cible $h^* \in \mathcal{H}^*$. Bien qu'il semble naturel de supposer que $\mathcal{H} = \mathcal{H}^*$, nous verrons que dans certaines circonstances, il est important du point de vue calculatoire que ces deux ensembles restent distincts.

La performance des hypothèses est mesurée par une fonction $\ell : \mathcal{H} \times \mathcal{X} \times \mathcal{Y} \to \mathbb{R}_+$, appelée *fonction de perte*. Elle associe à toute hypothèse h dans $\mathcal{H}$, toute entrée x dans $\mathcal{X}$ et toute sortie y dans $\mathcal{Y}$ un réel non négatif $\ell(h; x, y)$ évaluant l'écart entre la prédiction $h(x)$ et la sortie y. La *performance* d'une hypothèse h sur une séquence d'exemples $\boldsymbol{z} = (z_1, \cdots, z_m)$ selon ℓ est définie par la perte cumulée $\sum_{t=1}^{m} \ell(h; z_t)$. Parmi les fonctions de perte les plus utilisées dans les problèmes unidimensionnels, nous pouvons mentionner la fonction de perte *discrète* ou *zero-un*, et la fonction de perte *quadratique*, respectivement définies par :

$$\ell_{\text{DIS}}(h; x, y) = \mathbb{I}_{h(x) \neq y} \quad \text{et} \quad \ell_{\text{SQ}}(h; x, y) = \frac{1}{2}\|h(x) - y\|^2$$

Dans le cadre plus général des problèmes multidimensionnels ou structurels, les fonctions de perte sont souvent définies par des fonctions convexes sur $\mathcal{H}$ qui évaluent la distance entre les structures $h(x)$ et y (Bakir *et al.*, 2007).

Pour des raisons d'efficacité, les hypothèses qui nous intéressent en intelligence artificielle sont celles qui peuvent être représentées par un nombre "raisonnable" de symboles. Nous appelons *schéma de représentation* la donnée d'un ensemble Ω appelé *classe de représentation*, d'une *mesure de complexité*[2] $f : \Omega \to \mathbb{R}_+$ qui associe à chaque représentation $\omega \in \Omega$ un réel non négatif mesurant sa "complexité" ou "longueur" $f(\omega)$, et d'une fonction surjective qui associe à chaque représentation $\omega \in \Omega$ une hypothèse h_ω dans $\mathcal{H}$. Dans la suite, nous omettrons la fonction surjective lorsque sa définition ne fait pas d'ambiguïté, et nous spécifierons donc un schéma de représentation simplement par sa classe Ω et sa mesure de complexité f. L'espace des hypothèses engendrées par Ω est dénoté $\mathcal{H}_\Omega$, c'est à dire $\mathcal{H}_\Omega = \{h_\omega : \omega \in \Omega\}$. La *longueur de description* d'une hypothèse $h \in \mathcal{H}_\Omega$ selon f, notée $f(h)$, est définie par la longueur $f(\omega)$ de la plus petite représentation $\omega \in \Omega$ pour laquelle $h_\omega = h$.

En résumé, les principaux composants d'un problème d'apprentissage supervisé sont :

- un espace d'instances $\mathcal{X}$,
- un espace de décisions $\mathcal{Y}$,
- un schéma de représentation (Ω, f) engendrant l'espace $\mathcal{H}_\Omega$ mesuré par f,
- un espace de fonctions cibles $\mathcal{H}^*$, et
- une fonction de perte ℓ.

Les paramètres du problème sont la dimension des entrées, la dimension des sorties, la longueur de description des hypothèses, et la fonction de perte.

Exemple 2. Reprenons notre scénario sur le filtrage anti-spam. Rappelons que l'espace des entrées est $\mathcal{X} = \{0, 1\}^n$ et l'espace des sorties est $\mathcal{Y} = \{-1, +1\}$. Un schéma de représentation souvent utilisé dans la littérature pour apprendre à classer les messages est celui des fonctions linéaires à seuil. La classe Ω est un ensemble de paires (w, θ) où w est un vecteur de $\mathbb{R}^n$ et θ un scalaire de $\mathbb{R}$. L'hypothèse associée à $\omega = (w, \theta)$ est définie par $h_\omega(x) = \text{sgn}(\langle w, x \rangle - \theta)$ où $\langle w, x \rangle$ désigne le produit scalaire entre w et x et $\text{sgn}(z)$ est la fonction qui retourne le signe

2. En apprentissage statistique, f est souvent appelée *fonction de régularisation*.

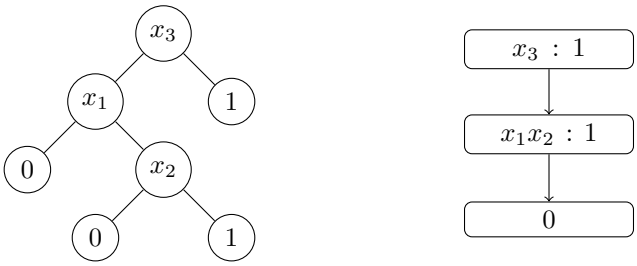

FIGURE 2: Un arbre de décision et une liste de décision pour la fonction $x_1x_2 \vee x_3$

du scalaire z. En d'autres termes, $h_w(x) = +1$ si et seulement si le produit scalaire entre w et x est supérieur au seuil θ. Une des mesures de complexité les plus utilisées en apprentissage linéaire est $f(w) = \frac{1}{2}\|\omega\|$, où $\|\omega\|$ est la norme Euclidienne de $\omega \in \mathbb{R}^{n+1}$.

Enfin, la performance de l'agent est le nombre d'erreurs de prédiction qu'il fait sur toute séquence de courriels $\boldsymbol{z} = ((x_1, y_1), \cdots, (x_m, y_m))$. Spécifiquement, ce nombre d'erreurs est mesuré par la perte discrète cumulée : $\sum_{t=1}^{m} \ell_{\text{DIS}}(h_{\omega_t}; x_t, y_t)$ où ω_t est la fonction linéaire à seuil maintenue par l'agent au tour t.

Classes de concepts. Afin d'analyser et comparer les modèles d'apprentissage étudiés dans ce chapitre, nous utiliserons souvent des hypothèses booléennes, appelées aussi *concepts*. Dans ce contexte, l'espace des entrées est l'hypercube $\{0,1\}^n$ et l'ensemble de sortie est $\{0,1\}$ ou bien $\{-1,+1\}$ selon le contexte. étant donné un ensemble de variables booléennes $\{x_1, \cdots, x_n\}$, rappelons qu'un littéral est une variable x_i ou sa négation $\overline{x}_i$, un terme (ou monômial) est une conjonction de littéraux et une clause est une disjonction de littéraux. Une formule en forme normale disjonctive, appelée aussi DNF, est une disjonction de termes et une formule en forme normale conjonctive, appelée CNF, est une conjonction de clauses. Une formule est *monotone* si tous ses littéraux sont positifs.

Comme nous l'avons vu dans l'exemple 2, une fonction $h : \{0,1\}^n \to \{-1,+1\}$ est dite *linéaire à seuil* si elle representable par une paire (w, θ) où $w \in \mathbb{R}^n$ et $\theta \in \mathbb{R}$, telle que $h(x) = \text{sgn}(\langle w, x\rangle - \theta)$. La fonction linéaire h est dite *booléenne à seuil* si $w \in \{0,1\}^n$ et $\theta \in \mathbb{N}$. Par exemple, la fonction de majorité qui associe à toute entrée x la classe $+1$ si et seulement si x contient une majorité de 1, est définie par $w = \mathbf{1}$ et $\theta = \lfloor \frac{n}{2} \rfloor$.

Un *arbre de décision* (booléen) un arbre binaire dont chaque feuille est étiquetée par 0 ou 1 et chaque noeud interne est étiqueté par un index $i \in \{1, \cdots, n\}$ pointant sur deux fils. La classification d'une instance x selon l'hypothèse associée à l'arbre de décision ω est déterminée en partant de la racine de ω et en appliquant récursivement la règle de décision suivante : (1) si le noeud courant est un index i, alors brancher à gauche si $x_i = 0$ et à droite si $x_i = 1$; (2) si le noeud courant est une feuille alors retourner la valeur 0 ou 1 indiquée la feuille. La taille d'un arbre de décision est définie par le nombre de ses feuilles.

Une *liste de décision* est une séquence de règles $\langle t, v\rangle_d = ((t_1, v_1), \cdots, (t_d, v_d))$ où t_i est un terme et v_i est une valeur dans $\{0,1\}$. La classification d'une instance x selon l'hypothèse associée à la liste de décision ω est déterminée en partant de la première règle et en parcourant itérativement la liste jusqu'à ce qu'une règle soit déclenchée. La règle (t_i, v_i) est déclenchée par x si x est un modèle de t_i ; dans ce cas la valeur retournée est v_i.

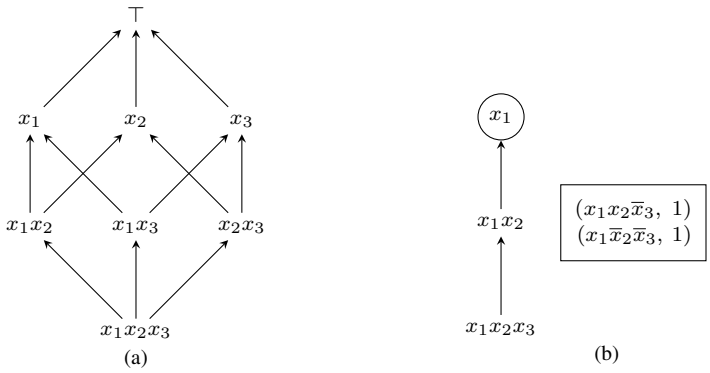

FIGURE 3: Apprentissage de termes monotones

1.3 Apprentissage Exact

Appelé aussi *apprentissage avec requêtes* (Angluin, 1988), l'apprentissage exact est un modèle d'apprentissage supervisé qui a été très étudié en intelligence artificielle pour éliciter des connaissances en interagissant avec un utilisateur. L'apprentissage exact peut être vu comme une variante du jeu de "Mastermind" faisant intervenir une séquence de questions-réponses entre l'apprenant et son environnement. L'objectif de l'apprenant est de découvrir la fonction cachée de l'environnement en posant un nombre minimum de questions. Dans cette section, nous allons illustrer l'apprentissage avec requêtes sur des problèmes de classification binaire ($\mathcal{Y} = \{0, 1\}$), mais gardons à l'esprit que ce modèle peut se généraliser à des tâches multi-classes ou multidimensionnelles.

Un problème d'apprentissage de concepts consiste en un espace d'instances $\mathcal{X}$, un espace d'hypothèse $\mathcal{H}_\Omega$ engendré à partir d'une classe de représentation Ω, et un espace de concepts cibles $\mathcal{H}^*$. Par exemple, un problème d'apprentissage de formules de Horn consiste en un espace d'assignations booléennes $\{0, 1\}^n$ et de l'ensemble de toutes les fonctions booléennes sur $\{0, 1\}^n$ représentables par des formules de Horn. Un exemple est ici une paire (x, y) où $x \in \mathcal{X}$ et $y \in \{0, 1\}$. L'exemple est dit *positif* si $y = 1$ et *négatif* si $y = 0$. Une hypothèse h est dite *cohérente* avec un exemple (x, y) si $h(x) = y$.

étant donnés deux concepts h_1 et h_2 dans $\mathcal{H}_\Omega$, nous disons que h_1 est plus *spécifique* que h_2 (ou h_2 est plus *générale* que h_1), noté $h_1 \leq h_2$, si $h_1(x) \leq h_2(x)$ pour toute instance $x \in \mathcal{X}$. Par exemple, la figure 3a représente le graphe associé à l'espace de tous les termes monotones (conjonctions de littéraux positifs) définis sur les variables booléennes $\{x_1, x_2, x_3\}$; un arc est associé entre deux termes ω_1 et ω_2 si $h_{\omega_1} \leq h_{\omega_2}$.

étant donné un ensemble d'entraînement $\boldsymbol{z}$, nous appelons *espace des versions* de $\boldsymbol{z}$ par rapport à $\mathcal{H}_\Omega$ l'ensemble de toutes les hypothèses h de $\mathcal{H}_\Omega$ qui sont cohérentes avec *tous* les exemples z_t de $\boldsymbol{z}$. Intuitivement, l'espace des versions d'une série d'exemples est la collection de toutes les hypothèses possibles qui peuvent représenter le concept cible. Une hypothèse h dans l'espace des versions de $\boldsymbol{z}$ est dite *maximalement spécifique* s'il n'existe aucune autre hypothèse dans cet espace qui est plus spécifique que h. La notion d'hypothèse *maximalement*

générale se définit de manière analogue.

Les requêtes principalement utilisées en apprentissage exact sont les *requêtes d'appartenance* et les *requêtes d'équivalence*. Soit h^* l'hypothèse cible de l'environnement.

- Une requête d'appartenance (MQ) associe à une instance x posée par l'apprenant la réponse *oui* si $h^*(x) = 1$, et *non* sinon.
- Une requête d'équivalence (EQ) associe à une hypothèse h posée par l'apprenant la réponse *oui* si $h = h^*$, et *non* sinon. En cas de réponse négative, l'adversaire renvoie un *contre-exemple* à l'apprenant ; le contre exemple est de la forme $(x, 1)$ si $h^*(x) = 1$ mais $h(x) = 0$, et de la forme $(x, 0)$ si $h^*(x) = 0$ mais $h(x) = 1$.

Intuitivement, les requêtes d'équivalence correspondent à une forme d'apprentissage *passif* où l'apprenant cherche à maintenir son hypothèse jusqu'à ce qu'elle soit réfutée par un contre-exemple. Les requêtes d'appartenance, pour leur part, correspondent à une forme d'apprentissage *actif* permettant à l'apprenant de demander la valeur d'un exemple de son choix. Bien entendu, l'apprenant doit poser un nombre raisonnable de requêtes pour que le modèle puisse être applicable.

Definition 1 (Apprentissage exact)**.** Soit $\mathcal{X}$ un espace d'entrées de dimension n, Ω une classe de représentation munie d'une mesure f et $\mathcal{H}^*$ une classe de concepts cibles sur $\mathcal{X}$. Nous disons que $\mathcal{H}^*$ est *apprenable avec requêtes d'équivalence et d'appartenance* par $\mathcal{H}_\Omega$, s'il existe un algorithme A tel que, pour tout concept cible $h^* \in \mathcal{H}^*$, A trouve une representation de h^* dans Ω, en posant un nombre de requêtes EQ et MQ polynômial en n et $f(h)$.

Nous disons que la classe cible $\mathcal{H}^*$ est *identifiable* si la classe de représentation Ω utilisée par l'apprenant coïncide avec $\mathcal{H}^*$ c'est à dire $\mathcal{H}^* = \mathcal{H}_\Omega$. Ce critère est important en représentation des connaissances lorsque nous cherchons à identifier un concept par une représentation précise qui sera ensuite utilisée pour diverses taches, telles que la déduction ou le diagnostic.

D'autre part, nous disons que $\mathcal{H}^*$ est *efficacement* apprenable si le temps de calcul de l'algorithme A borné par un polynôme de n et $f(h)$. Enfin, nous disons que $\mathcal{H}^*$ est *fortement* apprenable (par rapport au nombre d'attributs n) si la complexité des requêtes est polynômiale en $f(h)$ mais seulement poly-logarithmique en n. Ce dernier critère révèle toute son intérêt lorsque les données contiennent un grand nombre d'attributs dont, finalement, très peu s'avèrent pertinents pour séparer les exemples. Par exemple, dans le filtrage anti-spam, la difficulté majeure de l'apprenant est de trouver parmi des dizaines de milliers d'attributs, les quelques variables qui, combinées ensemble, permettent de prédire si un message est un spam.

Exemple 3 (Apprendre des termes)**.** Soit $\mathcal{H}_{\text{MM}} = \bigcup_{n\in\mathbb{N}} \mathcal{H}^n_{\text{MM}}$, où $\mathcal{H}^n_{\text{MM}}$ désigne l'espace de toutes les hypothèses représentables par des termes monotones sur les variables booléennes $\{x_1, \cdots, x_n\}$. Il est possible d'identifier efficacement toute hypothèse $h \in \mathcal{H}^n_{\text{MM}}$ avec au plus n requêtes d'équivalence : l'algorithme démarre avec le terme ω formé par toutes les variables, et, à chaque réception d'un contre-exemple (x, y), l'algorithme élimine dans ω tous les littéraux qui sont faux dans x. Notons que l'algorithme maintient systématiquement l'hypothèse la plus spécifique de son espace des versions. Ainsi chaque contre-exemple reçu est nécessairement positif ($y = 1$). étant donné que l'hypothèse courante h_ω est toujours plus spécifique que l'hypothèse cible h et que chaque contre-exemple suivant élimine au moins un littéral, le nombre de requêtes avec réponse négative est au plus n. Le comportement de cet algorithme est illustré dans la figure 3b, où le concept cible est x_1 et l'apprenant reçoit deux contre-exemples.

Exemple 4 (Apprendre des DNF monotones)**.** Rappelons qu'une formule en forme normale disjonctive (DNF) est une disjonction de termes. Une DNF est dite monotone si ses termes sont tous monotones. Soit $\mathcal{H}_{\text{MDNF}} = \bigcup_{n\in\mathbb{N}} \mathcal{H}^n_{\text{MDNF}}$ l'espace de toutes les hypothèses représentables par des DNF monotones sur $\{x_1, \cdots, x_n\}$. Il est démontré que cette classe n'est pas apprenable de manière purement passive : pour identifier certaines formules de taille polynômiale en n, le nombre de requête d'équivalence peut croître exponentiellement en n (Angluin, 1990).

En revanche, il est possible d'identifier les DNF monotones de manière active en utilisant à la fois des requêtes d'équivalence et des requêtes d'appartenance (Angluin, 1988). La stratégie d'apprentissage réside dans le fait que toute hypothèse $h \in \mathcal{H}_{\text{MDNF}}$ admet exactement une représentation minimale (par inclusion de termes). L'algorithme démarre avec la formule vide $\omega = \bot$ et, à chaque réception d'un contre-exemple (nécessairement positif) $(x, 1)$, l'algorithme identifie le terme t de la représentation minimale pour lequel x est un modèle de t en utilisant des requêtes d'appartenance. Pour cela l'idée est de poser une requête sur chaque voisin de x obtenu en permutant la valeur d'une seule variable, et de prendre la conjonction des variables x_i pour lesquelles la permutation de valeur a donné une réponse négative. Comme l'hypothèse courante h_ω est toujours plus spécifique que l'hypothèse cible et que chaque contre-exemple permet l'identification d'un nouveau terme dans la représentation minimale de h, il suffit donc de $k + 1$ requêtes d'équivalence et de kn requêtes d'appartenance pour identifier le concept cible, où k est le nombre de termes dans la représentation minimale de h.

Les recherches en apprentissage ont démontré que de nombreuses classes de concepts sont apprenables avec des requêtes d'équivalence et d'appartenance. En particulier, les fonctions de Horn (Angluin *et al.*, 1992) et les fonctions linéaires à seuil (Maass et Turan, 1994), sont efficacement identifiables. En se basant sur des propriétés de fermeture de ces classes, Blum *et al.* (1995) ont démontré le résultat suivant.

Théorème 1. Les DNF monotones, les formules de Horn, les fonctions à lecture unique et les formules à seuil sont *fortement* identifiables avec requêtes d'équivalence et d'appartenance.

Bshouty (1995) a démontré que les arbres de décision sont efficacement apprenables (mais non nécessairement identifiables) avec des requêtes d'équivalence et d'appartenance. Enfin, les listes de décision dont la taille des termes est bornée sont efficacement identifiables avec seulement des requêtes d'équivalence (Castro et Balcázar, 1995; Simon, 1995).

Avec une panoplie de tels résultats, on pourrait être tenté de croire que la plupart des classes de fonctions booléennes sont apprenables avec requêtes d'équivalence et d'appartenance. Ce n'est malheureusement pas le cas ; concernant la question des circuits booléens, Kharitonov (1993) a démontré qu'ils ne sont pas apprenables avec requêtes d'équivalence et d'appartenance, même dans le cadre ou l'hypothèse maintenue par l'apprenant n'est pas forcément un circuit booléen. Une autre question qui a fait l'objet de recherches intensives depuis l'introduction du modèle d'Angluin est celle de l'apprenabilité des DNF (non monotones). Sur ce point, Feldman (2009) a récemment fait une découverte capitale :

Théorème 2. A moins que $\text{NP} = \text{RP}$, les DNF ne sont pas efficacement apprenables avec requêtes d'équivalence et d'appartenance, même si l'hypothèse maintenue par l'apprenant est une disjonction de formules à seuil (cette dernière classe incluant celle des DNF).

Gardons à l'esprit que l'apprentissage exact ne se limite pas aux classes de fonctions booléennes. Ce modèle a été utilisé avec succès pour identifier certaines logiques de description

Algorithme 1: Apprentissage Enligne

Paramètres : espace d'entrées $\mathcal{X}$, espace de sorties $\mathcal{Y}$, classe d'hypothèses $\mathcal{H}$, function de perte ℓ

Initialisation : L'apprenant choisit une hypothèse initiale $h_1 \in \mathcal{H}$

Tours : pour chaque tour de jeu $t = 1, 2, \ldots$
(1) l'environnement présente une instance $x_t \in \mathcal{X}$
(2) l'apprenant effectue la prédiction $\hat{y}_t = h_t(x_t)$
(3) l'environnement révèle la response $y_t \in \mathcal{Y}$ et l'apprenant subit $\ell(h_t; x_t, y_t)$
(4) l'apprenant choisit une nouvelle hypothèse h_{t+1}

(Frazier et Pitt, 1996), ou encore divers fragments de la logique du premier ordre (Haussler, 1989; Khardon, 1999; Arias et Khardon, 2002). De manière orthogonale, le cadre ne se limite plus aujourd'hui aux requêtes d'équivalence et d'appartenance. Par exemple, certains résultats récents en inférence grammaticale ont été obtenus en utilisant des *requêtes d'édition* pour apprendre des boules de mots (Becerra-Bonache *et al.*, 2008). Un autre exemple est l'utilisation de *requêtes d'injection* pour apprendre certaines classes de circuits booléens (Angluin *et al.*, 2009). Une application pratique de ces requêtes d'injection est l'identification de réseaux sociaux à partir du comportement des agents (Angluin *et al.*, 2010).

1.4 Apprentissage Enligne

Parmi les paradigmes d'apprentissage supervisé étudiés dans la littérature, l'apprentissage enligne est certainement un des cadres qui se rapproche le plus de l'apprentissage naturel. Comme le décrit la figure 1, ce cadre peut-être vu comme un jeu répétitif à somme nulle entre l'apprenant et son environnement. A chaque tour de jeu, l'apprenant reçoit une instance du problème à résoudre. Il prédit ensuite une solution pour cette instance en utilisant son hypothèse et reçoit enfin la bonne réponse. L'apprenant subit alors une perte qui mesure l'écart entre la prédiction et la réponse. En se basant sur ces informations, l'apprenant est autorisé à mette à jour son hypothèse avant de procéder au tour suivant.

Dans l'apprentissage enligne, nous ne faisons aucune hypothèse sur la séquence des exemples choisis par l'environnement. Cette séquence peut être déterministe, stochastique, ou bien encore choisie par un environnement actif se comportant comme un adversaire. Comme l'environnement révèle la réponse y_t seulement après la prédiction $\hat{y}_t$, il peut évidemment induire l'apprenant en erreur à chaque tour de jeu. Dans ce cas, la perte cumulée de l'apprenant est maximale. Afin de prendre en compte cette éventualité, la performance de l'apprenant n'est pas mesurée de manière absolue par sa perte cumulée, mais de manière relative, par la différence entre la perte cumulée de l'apprenant et celle de la meilleure hypothèse de la classe cible $\mathcal{H}^*$. Cette différence est appelée *regret*, car elle capture le regret de l'apprenant à ne pas choisir la meilleure hypothèse s'il avait connu à l'avance la séquence d'exemples.

De manière formelle, un algorithme d'apprentissage enligne peut-être identifié par une fonction A qui associe à toute séquence d'exemples $\boldsymbol{z} = (z_1, \cdots, z_t)$ une hypothèse $h_t \in \mathcal{H}$.

Le regret à l'horizon m de A par rapport à une hypothèse $h^* \in H^*$ est défini par

$$regret_A(h^*, m) = \sup_{\boldsymbol{z} \in \mathcal{Z}^m} \left(\sum_{t=1}^{m} \ell(h_t; z_t) - \ell(h^*; z_t) \right)$$

Intuitivement, un algorithme d'apprentissage A atteint une performance non-triviale pour une classe cible $\mathcal{H}^*$ si, pour tout $h^* \in \mathcal{H}^*$, son regret par rapport à h^* est *sous-linéaire* par rapport à m, c'est-à-dire $regret_A(h^*, m) = o(m)$. Cette condition implique qu'à l'horizon infini, le comportement de l'apprenant est similaire à celui de la meilleure hypothèse.

$$\lim_{m \to \infty} \left(\frac{1}{m} regret_A(h^*, m) \right) = 0$$

Definition 2 (Apprentissage enligne)**.** Soient $\mathcal{X}$ un espace d'entrées de dimension n et $\mathcal{Y}$ un espace de sorties de dimension d. Soit Ω une classe de représentation associée avec sa mesure de complexité f. Enfin, soit $\mathcal{H}^*$ un sous-ensemble cible de l'espace d'hypothèses $\mathcal{H}_\Omega$. Nous disons que $\mathcal{H}^*$ est *apprenable enligne* à partir de $\mathcal{H}_\Omega$ et par rapport à ℓ s'il existe un algorithme d'apprentissage A pour Ω tel que, pour toute hypothèse cible $h^* \in \mathcal{H}^*$ et tout horizon $m \geq 1$, $regret_A(h^*, m)$ est sous-linéaire en m et polynômial en n, d, et $f(h^*)$.

Comme pour l'apprentissage exact, $\mathcal{H}^*$ est *identifiable* enligne si $\mathcal{H}^* = \mathcal{H}_\Omega$. Nous disons que $\mathcal{H}^*$ est *efficacement* apprenable enligne si, durant chaque tour t, le temps calcul par A pour prédire (étape 2) et réviser son hypothèse (étape 4) est polynômial en n, d et $f(h^*)$. Enfin, $\mathcal{H}^*$ est *fortement* apprenable enligne (par rapport au nombre d'attributs n) si $regret_A(h^*, m)$ est seulement poly-logarithmique en n.

Lorsque la classe d'hypothèses $\mathcal{H}^*$ est particulièrement difficile à apprendre, il est utile de contraindre les séquences d'exemples. Nous disons que $\mathcal{H}^*$ est apprenable enligne dans le cas *réalisable* s'il existe une fonction cible h^*, inconnue de l'apprenant, et pour laquelle toute séquence d'exemples $\langle x_t, y_t \rangle$ fournie par l'environnement est cohérente avec h^*.

Dans les applications où la fonction cible peut changer au cours du temps, l'objectif de l'apprenant est de "poursuivre" ou "traquer" l'évolution de la fonction. Ce cadre dynamique nous incite à généraliser la notion de regret. Soit $\boldsymbol{h}^* = \langle h_1^*, h_2^*, \ldots \rangle$ une séquence infinie de fonctions dans $\mathcal{H}^*$ et $\boldsymbol{h}_m^*$ la sous-séquence obtenue à l'horizon m. Un *changement* dans $\boldsymbol{h}_m^*$ correspond à un tour $t \leq m$ pour lequel $h_{t+1}^* \neq h_t^*$. Le regret à m pour un algorithme A par rapport à la séquence $\boldsymbol{h}^*$ est défini par

$$regret_A(\boldsymbol{h}^*, m) = \sup_{\boldsymbol{z} \in \mathcal{Z}^m} \left(\sum_{t=1}^{m} \ell(h_t; z_t) - \ell(h_t^*; z_t) \right)$$

Definition 3 (Tracking enligne)**.** Soient $\mathcal{X}$ un espace d'entrées de dimension n et $\mathcal{Y}$ un espace de sorties de dimension d. Soit Ω une classe de représentation associée avec sa mesure de complexité f. Enfin, soit $\mathcal{H}^*$ un sous-ensemble cible de l'espace d'hypothèses $\mathcal{H}_\Omega$. Nous disons que $\mathcal{H}^*$ est *poursuivable* à partir de $\mathcal{H}_\Omega$ et par rapport à ℓ s'il existe un algorithme d'apprentissage A pour Ω tel que, pour toute séquence d'hypothèses $\boldsymbol{h}^*$ dans $\mathcal{H}^*$ et tout horizon $m \geq 1$, $regret_A(\boldsymbol{h}^*, m)$ est sous-linéaire en m et polynômial en n, d, $f(h^*)$ et le nombre de changements dans $\boldsymbol{h}_m^*$.

Comme le souligne Blum (1997), la plupart des algorithmes enligne "apprennent très bien des choses simples". En effet, lorsque l'espace cible $\mathcal{H}^*$ est suffisamment petit, ces algorithmes convergent très rapidement, même lorsque les données sont éparpillées ou bruitées, ou encore lorsque la fonction cible évolue au cours du temps. Le trait commun de ces algorithmes réside dans l'optimisation convexe : l'apprenant optimise à la volée une fonction objective convexe qui est mise à jour à la fin de chaque tour.

Parmi les classes de représentation "simples", celles dont le cardinal est borné par un polynôme des dimensions d'entrée et de sortie sont efficacement apprenables, même dans le cas non-réalisable où l'apprenant est confronté à du bruit. Les algorithmes enligne pouvant apprendre ces classes d'hypothèses sont regroupés sous le paradigme d'apprentissage avec *conseil d'experts* (Littlestone et Warmuth, 1989; Vovk, 1990; Cesa-Bianchi *et al.*, 1997). L'idée générale est de considérer chaque fonction cible $h \in \mathcal{H}^*$ comme un expert et de prédire en se basant sur la performance des experts.

Exemple 5 (Weighted Majority)**.** L'algorithme d'apprentissage avec conseil d'experts le plus connu est "Weighted Majority", introduit par Littlestone et Warmuth (1989) et généralisé ensuite sous le nom de "Hedge" par Freund et Schapire (1997). Considérons un espace $\mathcal{H}^*$ de fonctions cibles de $\mathcal{X}$ dans $\mathcal{Y}$, appelées experts. Nous supposons que $\mathcal{H}^*$ est fini et de taille N. Dans l'algorithme de la majorité pondérée, l'apprenant démarre avec un vecteur de poids w_1 qui associe à chaque expert $i \in \{1, \cdots, N\}$ le poids $w_{1,i} = 1$. Durant chaque tour t, l'apprenant prédit avec la règle

$$\hat{y}_t = \frac{\sum_{i=1}^{N} w_{t,i} h_i^*(x_t)}{\sum_{j=1}^{N} w_{t,j}} \tag{1.1}$$

Ainsi, l'algorithme calcule la moyenne pondérée des prédications de chaque expert. Dans le cas où $\mathcal{Y} = \{-1, +1\}$, l'algorithme retourne le signe de la moyenne pondérée.

L'idée clé de l'algorithme réside dans la mise à jour multiplicative des poids. étant donné un paramètre d'apprentissage η, chaque expert $i \in \{1, \cdots, N\}$ est mis à jour par la règle

$$w_{t+1,i} = w_{t,i} e^{-\eta \ell(h_i^*; x_t, y_t)} \tag{1.2}$$

où $\ell(h_i^*; x_t, y_t)$ est la perte de l'expert h_i sur l'exemple (x_t, y_t). Intuitivement, un expert qui prédit incorrectement verra son poids diminuer à vitesse exponentielle. Ainsi, avec une telle règle, l'apprenant se focalise rapidement sur le meilleur expert. Notons qu'en normalisant la règle 1.2, l'algorithme de majorité pondérée apprend à rechercher le meilleur expert dans $\mathcal{H}^*$ en maintenant des hypothèses dans l'enveloppe convexe de $\mathcal{H}^*$.

Supposons que ℓ soit fonction de perte convexe prenant des valeurs dans $[0, 1]$. Alors d'après le théorème 2.2. dans (Cesa-Bianchi et Lugosi, 2006), le regret par rapport à tout expert de l'algorithme de majorité pondérée en utilisant $\eta = \sqrt{8(\ln N)/m}$ est borné par :

$$2\sqrt{\frac{m \ln N}{2}}$$

Notons que si l'algorithme ne connaît pas à l'avance le nombre de tours m, le paramètre η peut être rendu adaptatif en utilisant à chaque tour $\eta_t = \sqrt{8(\ln N)/t}$. Le regret augmente alors légèrement en ajoutant le terme $\sqrt{(\ln N)/8}$. Dans le cas particulier où $\mathcal{Y} = \{-1, +1\}$ et la

fonction de perte $\ell(h; x_t, y_t)$ est donnée par la valeur absolue $|h(x_t) - y_t|$, le nombre maximum d'erreurs que peut faire l'algorithme de majorité pondérée est borné par :

$$\min_i M_i + 2 \ln N + \sqrt{2(\ln N) \min_i M_i}$$

où M_i est le nombre d'erreurs faites par l'expert i. Par exemple, le nombre maximum d'erreurs que peut faire l'algorithme de majorité pondérée pour apprendre un monotone monomial de taille au plus k est en $O(k \ln n + \min_i M_i)$. En se basant sur les propriétés de l'algorithme Weighted Majority, nous pouvons donc déduire le résultat suivant.

Théorème 3. Toute classe d'hypothèses dont la taille est un polynôme des dimensions d'entrée et de sortie est apprenable enligne à partir de son enveloppe convexe, et par rapport à toute fonction de perte convexe et bornée.

Ce dernier résultat peut même être généralisé au tracking d'experts, en utilisant des variantes de Weighted Majority (Herbster et Warmuth, 1998).

C'est certainement dans la reconnaissance de fonctions linéaires, et plus généralement les fonctions polynômiales à seuil, que les algorithmes d'apprentissage enligne ont eu le plus de succès. Les exemples suivants examinent deux des algorithmes les plus connus en classification linéaire. Nous supposons ici que $\mathcal{X} = \mathbb{R}^n$ et $\mathcal{Y} = \{-1, +1\}$.

Exemple 6 (Perceptron)**.** Introduit par Rosenblatt (1958), le Perceptron est un classifieur linéaire avec mises à jour additives de poids. Dans sa version la plus simple, l'algorithme utilise un seuil nul et maintient un vecteur de poids $w \in \mathbb{R}^n$. Initialement, le vecteur est $w_1 = 0$. Durant chaque tour t, le Perceptron prédit avec la règle

$$\hat{y}_t = \operatorname{sgn} \langle w_t, x_t \rangle \tag{1.3}$$

En d'autres termes, la valeur prédite est le signe du produit scalaire du vecteur de poids w_t et de l'instance x_t. Si la prédiction est correcte alors $w_{t+1} = w_t$. Sinon, l'algorithme utilise la mise à jour

$$w_{t+1} = w_t + \eta y_t x_t \tag{1.4}$$

où η est un paramètre d'apprentissage. La performance du Perceptron a fait l'objet de nombreuses études dans la littérature (voir e.g. Novikov, 1962; Freund et Schapire, 1999; Shalev-Shwartz et Singer, 2007). Cette performance est généralement établie en utilisant une fonction de perte convexe appelée *perte charnière* (hinge loss) et définie de la manière suivante

$$\ell_\gamma(w; x, y) = \max [0, \gamma - y \langle w, x \rangle] \tag{1.5}$$

Cette fonction pénalise toute fonction linéaire pour laquelle la marge $y \langle w, x \rangle$ est plus petite que γ. Notons que si $y \neq \operatorname{sgn} \langle w, x \rangle$ alors $\ell_\gamma(w; x, y) \geq \gamma$. Par conséquent, la perte charnière cumulée sur le nombre d'exemples est une borne supérieure de γM, où M est le nombre total d'erreurs faites par l'algorithme. Dans le cas du Perceptron, nous utilisons $\gamma = 1$. Soit $((x_1, y_1), \cdots, (x_m, y_m))$ une séquence arbitraire d'exemples et $X_2 = \max_t \|x_t\|$, où $\|x_t\|$ est la norme Euclidienne de x_t. Soit u un vecteur de $\mathbb{R}^n$ et $L = \sum_{t=1}^m \ell_\gamma(u; x_t, y_t)$. Alors d'après

le résultat récent de Shalev-Shwartz et Singer (2007), le nombre total d'erreurs faites par le Perceptron est borné par

$$L + \frac{1}{2}\|u\|X_2^2 + \|u\|X_2\sqrt{L} \tag{1.6}$$

Ce résultat se généralise aisément aux fonctions linéaires à seuil en supposant que le dernier élément de chaque instance x_t possède toujours la valeur -1.[3] Dans le cadre booléen où $\mathcal{X} = \{0,1\}^n$, nous obtenons le résultat suivant.

Théorème 4. Les fonctions linéaires à seuil sont apprenables enligne par rapport à la fonction de perte charnière avec $\gamma = 1$.

Exemple 7 (Winnow)**.** L'algorithme Winnow, introduit par Littlestone (1988), ressemble au Perceptron par sa simplicité, mais utilise une mise à jour multiplicative, plutôt qu'additive, des poids. Nous examinons ici la version présentée dans (Grove *et al.*, 2001). Comme pour le Perceptron, Winnow maintient un vecteur de poids $w \in \mathbb{R}^n$. Initialement, le vecteur est $w_1 = 1$. Durant chaque tour t, Winnow prédit avec la règle 1.3. Si la prédiction est correcte alors $w_{t+1} = w_t$. Sinon l'algorithme utilise, pour chaque $i \in \{1, \cdots, n\}$, la mise à jour

$$w_{t+1,i} = w_{t,i} e^{\eta y_t x_{t,i}} \tag{1.7}$$

où η est un paramètre d'apprentissage. Comme pour le Perceptron, la convergence de Winnow a fait l'objet de nombreuses investigations (Littlestone, 1988, 1989; Grove *et al.*, 2001; Shalev-Shwartz et Singer, 2007). Nous présentons ici le résultat de Shalev-Shwartz et Singer. Soit $((x_1, y_1), \cdots, (x_m, y_m))$ une séquence arbitraire d'exemples et $X_\infty = \max_t \max_i |x_i|$. Soit u un vecteur de $\mathbb{R}^n$, γ une charnière entre 0 et 1, et $L = \sum_{t=1}^m \ell_\gamma(u; x_t, y_t)$. Alors, le nombre total d'erreurs faites par Winnow est borné par

$$\frac{1}{\gamma}L + \frac{2X_\infty^2}{\gamma^2}(\log n) + \frac{X_\infty}{\gamma^{\frac{3}{2}}}\sqrt{2(\log n)L} \tag{1.8}$$

Le regret est dépendant du choix de γ, mais il est seulement logarithmique en la dimension d'entrée. A nouveau, le résultat peut s'étendre aux fonctions linéaires à seuil en supposant que le dernier attribut de chaque instance possède toujours la valeur -1. Dans le cas où $\mathcal{X} = \{0,1\}^n$, rappelons qu'une fonction *booléenne* à seuil est une fonction linéaire à seuil pour laquelle les poids sont restreints à des booléens, et le seuil est un entier. Une fonction booléenne à seuil (w, θ) est dite (k, r) si $\sum_i |w_i| = k$ et $\theta = r$.

Théorème 5. Les fonctions booléennes à seuil (k, r) sont fortement apprenables enligne par rapport à la fonction de perte charnière avec $\gamma = \frac{1}{2}/(k + r + \frac{1}{2})$.

Ce théorème implique, entre autres, que les clauses sont fortement apprenables enligne. Notons que le résultat peut être généralisé au tracking de fonctions booléennes à seuil, en utilisant des variantes de Winnow (Auer et Warmuth, 1998; Herbster et Warmuth, 2001).

L'apprentissage enligne ne se limite évidemment pas à la classification binaire. Les familles d'algorithmes additifs et multiplicatifs, incluant respectivement le Perceptron et Winnow, comprennent des versions multi-classes étudiées dans (Crammer et Singer, 2003b; Crammer *et al.*,

3. Si ce n'est pas le cas, nous rajoutons simplement un autre élément à x.

2006). De manière analogue, ces familles couvrent aussi les problèmes de régression, avec des algorithmes populaires comme la descente de gradient (Windrow et Hoff, 1960) ou le gradient exponentiel (Kivinen et Warmuth, 1997). Concernant l'apprentissage multidimensionnel, les algorithmes enligne ont été utilisés avec succès dans l'ordonnancement de préférences (e.g. Cohen *et al.*, 1999; Crammer et Singer, 2003a), et l'appariement de structures (e.g. Lafferty *et al.*, 2001; Taskar *et al.*, 2006; Collins *et al.*, 2008).

1.5 Apprentissage Statistique

Le dernier modèle d'apprentissage supervisé présenté dans ce chapitre a été introduit en statistique avec les travaux de Vapnik et Chervonenkis (1971, 1974), puis en informatique théorique avec ceux de Valiant (1984). L'intérêt fondamental de ce modèle est de fournir des bornes sur la capacité de généralisation des algorithmes d'apprentissage.

Dans ce modèle, nous supposons que l'environnement est une source d'exemples qui, à chaque appel de l'apprenant, fournit un exemple tiré aléatoirement selon une distribution fixe, mais inconnue de l'apprenant. Cette hypothèse nous permet de voir une séquence d'exemples fournis par l'environnement comme un échantillon, et donc, de considérer le problème d'apprentissage comme un problème d'inférence statistique.

En termes plus formels, considérons une distribution de probabilités $\mathcal{D}$ sur $\mathcal{X} \times \mathcal{Y}$, où $\mathcal{X}$ est un espace d'instances et $\mathcal{Y}$ un espace de décisions. Nous appelons *échantillon* une séquence $\boldsymbol{z} = (z_1, \cdots, z_m)$ d'exemples tirés selon la distribution jointe $\mathcal{D}^m$. Etant donné une classe de représentation Ω, un algorithme d'apprentissage statistique pour Ω est une fonction A qui prend en entrée un échantillon $\boldsymbol{z}$ tiré aléatoirement selon $\mathcal{D}^m$ et retourne en sortie une hypothèse $A(\boldsymbol{z})$ dans la classe $\mathcal{H}_\Omega$. En général, nous souhaiterions que A puisse produire, avec une forte probabilité sur $\mathcal{D}^m$, une hypothèse h capable de prédire correctement sur des exemples de test fournis par la même source que les exemples d'entraînement. La performance de l'hypothèse h est mesurée par son *risque*, noté $risk_{\mathcal{D}}(h)$; il est défini par la perte espérée de h sur un exemple z tiré aléatoirement selon $\mathcal{D}$,

$$risk_{\mathcal{D}}(h) = \mathbb{E}_{z \sim \mathcal{D}}[\ell(h; z)]$$

Soit $\mathcal{H}^*$ une classe d'hypothèses cibles. Le but de l'apprenant A est de minimiser son regret probabiliste par rapport à toute hypothèse $h^* \in \mathcal{H}^*$, c'est-à-dire

$$\mathbb{P}_{\boldsymbol{z} \sim \mathcal{D}^m}\left[risk_{\mathcal{D}}(A(\boldsymbol{z})) \geq risk_{\mathcal{D}}(h^*) + \epsilon\right] \leq \delta$$

pour deux paramètres $\delta \in (0, 1)$ et $\epsilon > 0$, appelés respectivement la *confiance* et la *précision*. En se basant sur ces notions, le modèle d'apprentissage "agnostique" introduit dans (Haussler, 1992; Kearns *et al.*, 1994) fournit un cadre général à l'apprentissage statistique.

Definition 4 (Apprentissage agnostique)**.** Soient $\mathcal{X}$ un espace d'entrées de dimension n et $\mathcal{Y}$ un espace de sorties de dimension d. Soit Ω une classe de représentation associée avec sa mesure f. Enfin, soit $\mathcal{H}^*$ un sous-ensemble cible de l'espace d'hypothèses $\mathcal{H}_\Omega$. Nous disons que $\mathcal{H}^*$ est *agnostiquement apprenable* par $\mathcal{H}_\Omega$ s'il existe un algorithme A et un polynôme p tels que, pour toute distribution $\mathcal{D}$ sur $\mathcal{X} \times \mathcal{Y}$, toute hypothèse $h^* \in \mathcal{H}^*$ et tous paramètres $\delta \in (0, 1)$ et $\epsilon > 0$, après avoir reçu $p(n, d, f(h^*), \frac{1}{\delta}, \frac{1}{\epsilon})$ exemples tirés selon $\mathcal{D}$, A retourne une hypothèse $h \in \mathcal{H}_\Omega$ telle qu'avec une probabilité $1 - \delta$, $risk_{\mathcal{D}}(h) \leq risk_{\mathcal{D}}(h^*) + \epsilon$.

Dans le modèle "agnostique", la distribution $\mathcal{D}$ est *arbitraire*, ce qui implique qu'il n'existe a priori aucune dépendance fonctionnelle entre une instance x et une décision y dans un exemple tiré dans $\mathcal{D}$. En revanche, dans le modèle d'apprentissage *probablement approximativement correct* (PAC) de Valiant (1984), nous supposons qu'il existe une dépendance fonctionnelle gouvernée par une fonction cible $h^* \in \mathcal{H}^*$. Dans ce cadre réalisable, $\mathcal{D}$ est une distribution sur l'ensemble $\mathcal{X}$; chaque exemple fourni par l'environnement est une paire $z = (x, h^*(x))$ où x est tiré aléatoirement selon $\mathcal{D}$. Le risque est donc défini par

$$risk_{\mathcal{D}}(h) = \mathbb{E}_{x\sim\mathcal{D}}[\ell(h; x, h^*(x))]$$

Definition 5 (Apprentissage PAC)**.** Soient $\mathcal{X}$ un espace d'entrées de dimension n et $\mathcal{Y}$ un espace de sorties de dimension d. Soit Ω une classe de représentation associée avec sa mesure f. Enfin, soit $\mathcal{H}^*$ un sous-ensemble cible de l'espace d'hypothèses $\mathcal{H}_\Omega$. Nous disons que $\mathcal{H}^*$ est *PAC apprenable* par $\mathcal{H}_\Omega$ s'il existe un algorithme A et un polynôme p tels que, pour toute distribution $\mathcal{D}$ sur $\mathcal{X}$, toute hypothèse $h^* \in \mathcal{H}^*$ et tous paramètres $\delta \in (0, 1)$ et $\epsilon > 0$, après avoir reçu $p(n, d, f(h^*), \frac{1}{\delta}, \frac{1}{\epsilon})$ exemples tirés selon $\mathcal{D}$ et étiquetés par h^*, A retourne une hypothèse $h_\omega \in \mathcal{H}_\Omega$ telle qu'avec une probabilité $1 - \delta$, $risk_{\mathcal{D}}(h_\omega) \leq \epsilon$.

De manière analogue aux autres modèles d'apprentissage, nous disons $\mathcal{H}^*$ est *efficacement* PAC apprenable si la complexité de A est polynomiale en la taille de son échantillon. Nous disons aussi que $\mathcal{H}^*$ est PAC *identifiable* si $\mathcal{H}^* = \mathcal{H}_\Omega$.

Une des approches les plus connues pour satisfaire le critère d'apprenabilité "statistique" est d'utiliser le principe de convergence uniforme. Etant donné un échantillon $\boldsymbol{z}$ dont les exemples sont tirés aléatoirement et indépendamment selon une distribution fixe $\mathcal{D}$, le *risque empirique* d'une hypothèse h sur $\boldsymbol{z}$ selon une fonction de perte ℓ est défini par

$$risk_{\text{EMP}}(h) = \frac{1}{m}\sum_{t=1}^{m} \ell(h; z_t)$$

Intuitivement, le principe de convergence uniforme signifie que, pour toute hypothèse $h \in \mathcal{H}$ et toute distribution $\mathcal{D}$, le risque empirique de h est, avec une forte probabilité, "proche" du vrai risque de h. Afin de mesurer précisément la proximité entre ces deux risques, diverses dimensions ont été proposées dans la littérature de l'apprentissage statistique (Vapnik et Chervonenkis, 1971; Vapnik, 1998; Antos *et al.*, 2002; Bartlett et Mendelson, 2002). Nous allons nous focaliser sur une dimension indépendante des données, due à Vapnik et Chervonenkis, et abrégée sous le nom de VC-dimension.

Pour des raisons de clarté, nous supposons ici que $\mathcal{Y} = \{0, 1\}$. [4] Etant donné un échantillon $\boldsymbol{z} = (z_1, \cdots, z_m)$ dans $\mathcal{Z}^m$ et une hypothèse $h \in \mathcal{H}$, le *vecteur de perte* induit par h sur $\boldsymbol{z}$ est défini par $(\ell(h; z_1), \cdots, \ell(h; z_m))$ où $\ell(h; z_i)$ est la perte discrète de h sur l'exemple $z_i = (x_i, y_i)$. La *croissance* de $\mathcal{H}$ sur $\boldsymbol{z}$, notée $\Pi_{\mathcal{H}}(\boldsymbol{x})$, est le nombre de vecteurs de perte distincts sur $\boldsymbol{z}$ induits par $\mathcal{H}$,

$$\Pi_{\mathcal{H}}(\boldsymbol{z}) = |\{(\ell(h; z_1), \cdots, \ell(h; z_m) : h \in \mathcal{H}\}|$$

Notons que pour tout échantillon $\boldsymbol{z}$ de taille m, $\Pi_{\mathcal{H}}(\boldsymbol{z}) \leq 2^m$. Nous disons que $\boldsymbol{z}$ est *pulvérisé* par $\mathcal{H}$ si $\Pi_{\mathcal{H}}(\boldsymbol{z}) = 2^m$. De manière plus générale, la *fonction de croissance* de $\mathcal{H}$ est le nombre

4. Voir (Vapnik, 1998) pour une généralisation de la VC-dimension aux fonctions multivaluées ou réelles.

maximum de vecteurs de perte distincts induits par $\mathcal{H}$ sur un échantillon de taille donnée,

$$\Pi_{\mathcal{H}}(m) = \max_{\boldsymbol{z} \in \mathcal{Z}^m} \Pi_{\mathcal{H}}(\boldsymbol{z})$$

La *VC-dimension* de $\mathcal{H}$, notée $\mathrm{VCdim}(\mathcal{H})$, est définie par la taille du plus grand échantillon qui peut être pulvérisé par $\mathcal{H}$.

$$\mathrm{VCdim}(\mathcal{H}) = \sup\{m \in \mathbb{N} : \Pi_{\mathcal{H}}(m) = 2^m\}$$

En se basant sur cette notion, une condition suffisante pour établir la convergence uniforme est que la VC-dimension de $\mathcal{H}$ soit finie. Le résultat suivant est démontré dans (Long, 1999).

Lemma 1 (Convergence uniforme)**.** Soit $\mathcal{H}$ un espace d'hypothèses de $\mathcal{X}$ dans $\{0, 1\}$. Si la VC-dimension de $\mathcal{H}$ est finie alors il existe une constante c telle que pour toute distribution $\mathcal{D}$ sur les exemples, tout échantillon de m exemples tirés aléatoirement et indépendamment selon $\mathcal{D}$, et tout paramètre de confiance $\delta \in (0, 1)$, avec une probabilité $1 - \delta$

$$\sup_{h \in \mathcal{H}} |risk_{\text{EMP}}(h) - risk_{\mathcal{D}}(h)| \leq c\sqrt{\frac{\mathrm{VCdim}(\mathcal{H}) + \ln(1/\delta)}{m}}$$

A partir de ce résultat, il suffit donc que la VC-dimension de notre classe d'hypothèses soit bornée par un polynôme de la dimension d'entrée, et que notre algorithme retourne une hypothèse minimisant le risque empirique.

Théorème 6. Soient $\mathcal{X}$ un espace d'entrées de dimension n et $\mathcal{H}_\Omega$ un espace d'hypothèses de $\mathcal{X}$ dans $\{0, 1\}$ engendrées par une classe de représentation Ω. Si $\mathrm{VCdim}(\mathcal{H}_\Omega)$ est un polynôme de n, alors toute classe cible $\mathcal{H}^* \subseteq \mathcal{H}_\Omega$ est agnostiquement apprenable par $\mathcal{H}_\Omega$ en retournant $\hat{h} = \mathrm{arginf}_{h \in \mathcal{H}_\Omega} risk_{\text{EMP}}(h)$ avec une taille d'échantillon d'au moins $\left(\frac{2c}{\epsilon}\right)^2 \left(d + \ln \frac{1}{\delta}\right)$.

Si l'on fait abstraction des ressources de calcul, de nombreuses classes de concepts sont agnostiquement apprenables en utilisant le principe de convergence uniforme. Nous pouvons mentionner la classe des termes monotones (sur n variables) dont la VC-dimension est de n, et les formules kDNF dont la VC-dimension est en $\Theta(n^k)$. Plus généralement toute classe de concepts $\mathcal{H}^*$ dont la taille est bornée $2^{p(n)}$, où p est un polynôme, est apprenable puisque $\mathrm{VCdim}(\mathcal{H}^*) \leq \log_2 |\mathcal{H}^*|$. Parmi les classes de taille infinie, les fonctions linéaires à seuil sont agnostiquement apprenables car leur VC-dimension est de $n + 1$.

Cependant, d'un point de vue calculatoire, une des difficultés centrales associées au principe de convergence uniforme est de produire des algorithmes capables de retourner en temps polynômial une hypothèse minimisant le risque empirique sur les données d'entraînement. Avec la fonction de perte discrète, les termes monotones, les kDNF et les fonctions linéaires à seuil sont efficacement identifiables dans le modèle PAC. Ce n'est malheureusement pas le cas dans le modèle agnostique où les exemples ne sont plus étiquetés par une fonction cible : à moins que $\mathrm{NP} = \mathrm{RP}$, les termes monotones (et donc les kDNF) ne sont pas efficacement identifiables Kearns *et al.* (1994), ainsi que les fonctions linéaires à seuil (Höffgen *et al.*, 1995). Ce dernier résultat est à comparer avec le théorème 4, où un simple algorithme comme le Perceptron peut apprendre efficacement des fonctions linéaires, même en présence de bruit, dès lors que la fonction de perte n'est plus la fonction discrète, mais un substitut convexe de celle-ci.

Dans le cadre réalisable, une des questions qui est restée ouverte depuis plus d'une vingtaine d'année a été de savoir si les DNF de taille polynômiale en n sont efficacement PAC apprenables. Un premier résultat a été obtenu par Jackson (1997) avec l'algorithme "Harmonic Sieve". Cet algorithme apprend en temps polynômial une représentation de Fourier de la DNF cible en utilisant un échantillon tiré selon la distribution uniforme et des requêtes d'appartenance pour affiner son exploration. Malheureusement, l'espoir de généraliser ce résultat au véritable modèle PAC (distributions arbitraires) est très limité puisque, récemment, Alekhnovich *et al.* (2008) et Feldman (2009) ont démontré le résultat suivant.

Théorème 7. A moins que NP $=$ RP, les DNF ne sont pas efficacement PAC apprenables, même si l'apprenant à accès aux requêtes d'appartenance, et même si l'hypothèse qu'il retourne est une disjonction de formules à seuil.

Dans cette section, nous avons examiné l'apprentissage statistique sous le regard de la convergence uniforme. Même si ce principe est très utilisé dans la littérature, gardons à l'esprit que d'autres principes permettent de dériver des bornes sur le risque en se focalisant sur des algorithmes satisfaisant certaines propriétés. Parmi ces principes, nous pouvons citer le boosting d'apprenants faibles (Schapire, 1990; Freund, 1995), les algorithmes auto-limitatifs (Langford et Blum, 2003), et la stabilité algorithmique (Bousquet et Warmuth, 2002).

1.6 Apprentissage par Renforcement

Contrairement aux modèles étudiés jusqu'à présent, l'apprentissage par renforcement est un cadre d'apprentissage dans lequel le feedback communiqué à l'apprenant se résume à une "récompense" ou "pénalité". Notons que l'apprentissage par renforcement, dans son paradigme général, couvre tout un éventail de problèmes étudiés en théorie des jeux et en recherche opérationnelle. Parmi les problèmes les plus connus, nous pouvons citer les *bandits multi-bras* (Robbins, 1952; Banos, 1998; Gittings, 1989; Auer *et al.*, 2002a,b; Bubeck et Cesa-Bianchi, 2012), le *monitoring partiel* (Rustichini, 1999; Piccolboni et Schindelhauer, 2001; György *et al.*, 2007) et, bien entendu, l'apprentissage de *processus de décision séquentielle* (Andreae, 1969; Witten, 1977; Watkins, 1989; Watkins et Dayan, 1992; Sutton et Barto, 1998; Kaelbling, 2010). Dans cette section, nous allons focaliser notre attention sur le dernier problème.

En apprentissage de processus de décision séquentielle, l'objectif est d'apprendre à maximiser sa récompense totale en interagissant avec un environnement qui, au départ, est inconnu. L'environnement en question est souvent modélisé comme un *processus de décision Markovien* (Puterman, 1994). Pour un ensemble E, nous notons $\boldsymbol{P}_E$ l'ensemble de toutes les distributions de probabilités sur E. En se basant sur cette notation, un processus de décision Markovien consiste en un tuple $M = (\mathcal{S}, \mathcal{A}, T, R, \gamma)$ où $\mathcal{S}$ est un *espace d'états*, $\mathcal{A}$ est un *espace d'actions*, $T : \mathcal{S} \times \mathcal{A} \to \boldsymbol{P}_{\mathcal{S}}$ est la *fonction de transition*, $R : \mathcal{S} \times \mathcal{A} \to \boldsymbol{P}_{[0,1]}$ est la *fonction de récompense* (bornée)[5], et γ est le *facteur de dévaluation* compris dans l'intervalle $[0, 1]$. Un processus de décision Markovien est *fini* si les espaces $\mathcal{S}$ et $\mathcal{A}$ sont tous deux finis.

Pour des raisons de clarté, nous supposerons que le modèle M de l'environnement est fini. Dans ce contexte, nous pouvons définir $T(\cdot \mid s, a)$ comme la distribution de probabilités

5. Par transformation linéaire, il est possible d'étendre l'intervalle $[0, 1]$ à tout intervalle dont les bornes sont constantes, sans changer les politiques optimales du processus (Ng *et al.*, 1999).

Algorithme 2: Apprentissage par Renforcement

Paramètres : espace d'états $\mathcal{S}$, espace d'actions $\mathcal{A}$, fonction (cachée) de transition T, fonction (cachée) de récompense R, facteur de dévaluation γ

Initialisation : L'environnement occupe un état $s_1 \in \mathcal{S}$ et le communique à l'agent

Tours : pour chaque tour de jeu $t = 1, 2, \ldots$
(1) l'agent perçoit l'état s_t et choisit une action $a_t \in \mathcal{A}$
(2) l'environnement retourne à l'agent la récompense r_t choisie aléatoirement selon $R(s_t, a_t)$ et occupe un nouvel état $s_{t+1} \in \mathcal{S}$ choisi aléatoirement selon $T(\cdot \mid s_t, a_t)$

associée à l'état $s \in \mathcal{S}$ et l'action $a \in \mathcal{A}$. Ainsi $T(s' \mid s, a)$ est la probabilité d'atteindre l'état s' si l'action a est accomplie dans l'état s.

Une *politique* est une stratégie pour choisir la prochaine action étant donné l'historique de tous les états observés jusqu'à présent. Une politique est *stationnaire* si elle choisit la prochaine action en se basant seulement sur l'état courant ; en d'autres termes une politique stationnaire est une fonction $\pi : \mathcal{S} \to \mathcal{A}$. La valeur d'un état s pour une politique stationnaire π, notée $V^\pi(s)$, est définie comme l'espérance de la récompense cumulative dévaluée obtenue est exécutant π à partir de l'état s ; en d'autres termes, $V^\pi(s) = \mathbb{E}[\sum_{t=1}^{\infty} \gamma^{t-1} r_t]$ où $r_t = \mathbb{E}[R(s_t, \pi(s_t))]$ est l'espérance de la t-ième récompense obtenue en suivant π à partir de s. De manière similaire, la valeur d'une paire état-action (s, a) pour une politique stationnaire π, notée $Q^\pi(s, a)$ est définie comme l'espérance de la récompense cumulative dévaluée obtenue en appliquant d'abord l'action a sur s, puis en suivant π à partir du nouvel état obtenu. Afin de maximiser ses récompenses, l'agent cherche à trouver une politique optimale π^* dont les fonctions de valeur, notées respectivement $V^*(s)$ et $Q^*(s, a)$, satisfont les conditions $V^* = \max_\pi V^\pi$ et $Q^* = \max_\pi Q^\pi$. Notons qu'une politique ne peut pas avoir une valeur au delà de $1/(1-\gamma)$ puisque la récompense maximale vaut 1.

Si le modèle M de l'environnement est communiqué dans son intégralité à l'agent, il est possible de trouver la fonction de valeur optimale ainsi que la politique optimale, en utilisant des algorithmes standards tels que la programmation linéaire, l'itération de valeur ou l'itération de politique (Puterman, 1994). Cependant, en apprentissage par renforcement, nous supposons que les fonctions de transition et de récompense dans M sont a priori inconnues de l'agent : il doit interagir avec son environnement pour acquérir des informations sur ces fonctions.

Le protocole d'apprentissage par renforcement, illustré dans la figure 2, est relativement similaire à celui de l'apprentissage enligne. A chaque tour t, l'agent perçoit l'état s_t et choisit une action a_t ; à partir de cette action, l'environnement retourne le feedback r_t choisi aléatoirement selon la fonction cachée de récompense R, et occupe un nouvel état s_{t+1} choisi aléatoirement selon la fonction cachée de transition T. Une *transition* est un tuple de la forme (s_t, a_t, r_t, s_{t+1}), qui peut être utilisé comme exemple pour apprendre les fonctions de recompense et de transition. Un *chemin* est une séquence de la forme $c_t = (s_1, a_1, r_1, s_2, \cdots, s_t)$ où chaque sous-séquence (s_i, a_i, r_i, s_{i+1}) est une transition. En se basant sur ces notions, un algorithme d'apprentissage par renforcement peut être vu comme une fonction A qui, à chaque étape t, retourne une politique *non stationnaire* $A_t : \{\mathcal{S} \times \mathcal{A} \times [0, 1]\}^* \times \mathcal{S} \to \mathcal{A}$. Les fonctions de valeur sont étendues de manière naturelle aux politiques non stationnaires. Spécifiquement,

étant donné une étape t et un chemin c_t, nous notons $V^{A_t}(c_t)$ l'espérance de la récompense cumulative dévaluée obtenue en exécutant la politique A_t à partir de la fin du chemin de c_t, c'est-à-dire $V^{A_t}(c_t) = \mathbb{E}[\sum_{i=0}^{\infty} \gamma^i r_{t+i}]$. Etant donné un paramètre de précision $\epsilon > 0$, nous disons que la politique A_t est *quasi-optimale* selon ϵ si $V^{A_t}(c_t) \geq V^*(s_t) - \epsilon$.

Nous avons à présent tout les notations en main pour définir le modèle PAC-MDP, introduit récemment par Strehl *et al.* (2009), et dont l'idée générale consiste à étendre le modèle statistique PAC à l'apprentissage par renforcement. Le modèle est construit sur la notion de *complexité de l'échantillonnage d'exploration* (ou plus simplement *complexité d'échantillonnage*), introduite par Kakade (2003), et qui mesure le temps nécessaire pour converger vers une politique quasi-optimale. De manière formelle, soit $c = (s_1, a_1, r_1, s_2, \ldots)$ un chemin aléatoire engendré en executant un algorithme d'apprentissage par renforcement A sur un processus de décision Markovien M. Notons c_t le sous-chemin de c obtenu depuis s_1 jusqu'à s_t. Pour tout $\epsilon > 0$, la complexité d'échantillonnage de A sur M selon ϵ est le plus petit entier T tel que, quelque-soit $t > T$, la politique A_t appliquée sur c_t est quasi-optimale selon ϵ.

Definition 6 (Apprentissage PAC-MDP)**.** Un algorithme A est dit *PAC-MDP* (probablement approximativement correct sur les processus de décision Markoviens) si, pour tout environnement $M = (\mathcal{S}, \mathcal{A}, T, R, \gamma)$, tout paramètre de précision $\epsilon > 0$ et tout paramètre de confiance $\delta \in [0, 1]$, avec une probabilité $1 - \delta$ la complexité d'échantillonnage de A sur M selon ϵ est polynômiale selon la taille de $\mathcal{S}$, la taille de $\mathcal{A}$, et selon les quantités $1/\epsilon$, $1/\delta$ et $1/(1-\gamma)$.

Par analogie avec le modèle statistique, nous disons que l'algorithme A est *efficacement* PAC-MDP si, pour chaque tour de jeu t, la complexité temporelle et la complexité spatiale de A requises pour choisir l'action a_t sont aussi polynômiales en $|\mathcal{S}|$, $|\mathcal{A}|$, $1/\epsilon$, $1/\delta$ et $1/(1-\gamma)$.

Afin de clarifier l'intérêt du modèle PAC-MDP, nous allons examiner deux algorithmes bien connus en apprentissage par renforcement, R-MAX et le Q-Learning retardé ; la borne de convergence du premier est meilleure sur le paramètre de précision et le facteur de dévaluation, alors que la borne de convergence du second est meilleure sur le nombre d'états.

Exemple 8 (R-MAX)**.** Introduit par Brafman et Tennenholtz (2002), R-MAX appartient à la famille des algorithmes d'apprentissage par renforcement *à base de modèles* ; ces algorithmes cherchent à apprendre la fonction de transition et la fonction de récompense de l'environnement $M = (\mathcal{S}, \mathcal{A}, T, R, \gamma)$, et utilisent leur modèle approximatif $\hat{M} = (\mathcal{S}, \mathcal{A}, \hat{T}, \hat{R}, \gamma)$ pour calculer une stratégie optimale. L'algorithme R-MAX construit les fonctions $\hat{T}$ et $\hat{R}$ de la manière suivante. Soit $n(s, a)$ le nombre de fois que l'agent applique l'action a dans l'état s. Notons $r[1], r[2], \cdots, r[n(s, a)]$ les récompenses obtenues à chaque fois. La fonction $\hat{R}$ est alors donnée par la récompense empirique :

$$\hat{R}(s, a) = \frac{1}{n(s, a)} \sum_{i=1}^{n(s,a)} r[i]$$

Soit $n(s, a, s')$ le nombre de fois que l'agent observe l'état s' après avoir appliqué l'action a dans l'état s. La fonction $\hat{T}$ est donnée par la distribution empirique de transition :

$$\hat{T}(s' \mid s, a) = \frac{n(s, a, s')}{n(s, a)}$$

A partir de ces deux fonctions empiriques, l'agent utilise la fonction de valeur $\hat{Q}$ pour construire sa stratégie. Spécifiquement, l'action choisie dans l'état s est donnée par $\operatorname{argmax}_{\mathcal{A}} \hat{Q}(s, a)$. La

mise à jour de la stratégie est obtenue en résolvant les équations de Bellman :

$$\hat{Q}(s,a) = \hat{R}(s,a) + \gamma \sum_{s' \in \mathcal{S}} \hat{T}(s' \mid s,a) \max_{a' \in \mathcal{A}} \hat{Q}(s',a') \qquad \text{si } n(s,a) \geq m,$$

$$\hat{Q}(s,a) = \frac{1}{1-\gamma} \qquad \text{sinon}$$

Ici, m est un paramètre de l'algorithme indiquant la quantité minimale d'échantillons nécessaires pour mettre à jour la fonction $\hat{Q}$. Rappelons que les équations de Bellman peuvent être résolues en utilisant des algorithmes de programmation dynamique. En se basant sur ces notions, Strehl *et al.* (2009) démontrent que R-MAX est efficacement PAC-MDP. Si l'on fait abstraction des termes logarithmiques, la complexité d'échantillonnage est en

$$\tilde{O}\left(\frac{|\mathcal{S}|^2|\mathcal{A}|}{\epsilon^3(1-\gamma)^6}\right)$$

Exemple 9 (Q-Learning retardé)**.** Le Q-Learning introduit dans la thèse de Watkins (1989) fait partie de la famille des algorithmes *sans modèle*, qui cherchent à apprendre une stratégie optimale sans construire le processus de décision Markovien de l'environnement. Nous présentons ici une version dite *retardée* de cet algorithme qui a été suggérée par Strehl *et al.* (2006). Comme son nom l'indique, l'algorithme du Q-Learning maintient une estimation $\hat{Q}$ de la fonction de valeur Q associée aux paires état-action du modèle M de l'environnement. En notant $\hat{V}(s) = \operatorname{argmax}_{\mathcal{A}} \hat{Q}(s,a)$, l'action choisie par l'algorithme dans l'état s est celle qui maximise $\hat{V}(s)$. Dans le Q-Learning retardé, la mise à jour de $\hat{Q}$ est gouvernée par deux paramètres, un entier m indiquant le nombre minimum d'échantillons nécessaires à la mise à jour de $\hat{Q}$, et un réel $\epsilon \in [0,1]$ indiquant le bonus d'exploration ajouté à chaque paire état-action lorsque $\hat{Q}$ est mise à jour. Pour chaque paire (s,a) dans $\mathcal{S} \times \mathcal{A}$, si l'action a a été accomplie au moins m fois dans l'état s, alors l'algorithme calcule la quantité suivante :

$$\hat{W}(s,a) = \frac{1}{m} \sum_{i=1}^{m} (r[i] + \gamma \hat{V}(s[i])) + \epsilon_1$$

où $r[1], \cdots, r[m]$ et $s[1], \cdots, s[m]$ sont les m récompenses et états consécutifs les plus récents observés lorsque l'agent a accompli l'action a dans l'état s. Si la différence entre l'estimation courante $\hat{Q}(s,a)$ et la quantité $\hat{W}(s,a)$ est supérieure à $2\epsilon_1$, l'agent choisit la nouvelle valeur $\hat{Q}(s,a) = \hat{W}(s,a)$. Dans le cas où au moins une des deux conditions précédentes est insatisfaite, l'agent ne met pas à jour $\hat{Q}(s,a)$. En se basant sur cette règle, Strehl *et al.* (2006) démontrent que le Q-Learning retardé est efficacement PAC-MDP avec une complexité d'échantillonnage en

$$\tilde{O}\left(\frac{|\mathcal{S}||\mathcal{A}|}{\epsilon^4(1-\gamma)^8}\right)$$

L'analyse de convergence des algorithmes d'apprentissage par renforcement (Szepesvári, 1997; Kearns et Singh, 2002; Even-Dar et Mansour, 2003; Kakade, 2003) a non seulement permis le développement du modèle PAC-MDP, mais ouvert la voie à d'autres paradigmes plus récents, tels que le modèle KWIK (*knows what it knows*) récemment suggéré par Li *et al.* (2011). Ce modèle utilise des notions provenant à la fois de l'apprentissage enligne et de l'apprentissage statistique ; il s'avère être particulièrement élégant pour modéliser l'exploration active en apprentissage par renforcement.

1.7 Discussion

Rappelons qu'un modèle d'apprentissage est un cadre formel permettant de définir une mesure de la "difficulté" à apprendre à résoudre un problème de décision. Dans ce chapitre, nous avons cherché à mettre en lumière la diversité des modèles d'apprentissage. Même si l'on se focalise sur l'apprentissage supervisé, il existe plusieurs modèles, chacun apportant un point de vue sur la manière dont un agent apprend en interagissant avec son environnement. Naturellement, certains de ces modèles peuvent être "simulés" par d'autres modèles. Par exemple, le modèle exact avec requêtes d'équivalence est identique au modèle enligne réalisable avec perte discrète (Littlestone, 1988). L'idée est de simuler chaque requête d'équivalence par une erreur de prédiction de l'apprenant : le nombre de requêtes d'équivalences est donc précisément le nombre d'erreurs de l'apprenant. Il est possible de convertir les modèles enligne en modèles statistiques : les méthodes de conversions sont établies dans (Cesa-Bianchi *et al.*, 2004) pour le cadre agnostique et dans (Angluin, 1988; Littlestone, 1988) pour le cadre PAC.

Pour des raisons évidentes de place, nous n'avons pas examiné les modèles propres à l'apprentissage non supervisé ou semi-supervisé. Ces modèles sont décrits dans plusieurs ouvrages (Duda *et al.*, 2001; Chapelle *et al.*, 2006; Koller et Friedman, 2009; Theodoridis et Koutroumbas, 2009; Cornuéjols *et al.*, 2010). Le chapitre **??** du présent ouvrage traite aussi en partie de ce cadre en présentant quelques algorithmes d'apprentissage non supervisé.

Enfin, parmi les nombreuses perspectives de recherche autour des modèles d'apprentissage, nous en mentionnerons trois :

Apprentissage relationnel. En apprentissage dit *attribut-valeur*, les observations perçues par l'agent sont des "objets" particuliers d'une même classe définie sur un ensemble d'attributs. L'apprentissage *relationnel* (De Raedt, 2008) offre un cadre plus général où chaque observation peut désigner un nombre arbitraire d'objets pouvant appartenir à des classes différentes inter-connectées par des relations. L'éventail des applications en apprentissage relationnel est immense, incluant la classification de molécules chimiques, la prédiction de liens dans les réseaux sociaux, l'analyse de citations dans les documents, ou encore la prise de décision séquentielle dans les environnements multi-objets. Les hypothèses utilisées pour ce cadre sont souvent des modèles graphiques contenant à la fois un composant relationnel et un composant probabiliste (Kersting, 2006; Getoor et Taskar, 2007). Par exemple, les réseaux de Markov logiques (Richardson et Domingos, 2006) sont des ensembles de clauses du premier ordre, chacune associée avec un poids non négatif, et dont la sémantique décrit une distribution de probabilités sur l'espace de Herbrand. L'apprenabilité de ces modèles probabilistes relationnels dans le cadre supervisé, non supervisé, ou par renforcement, reste encore largement inconnue.

Apprentissage structurel. Rappelons qu'en apprentissage structurel, l'espace des décisions d'un agent possède une structure combinatoire ; les décisions peuvent prendre la forme de permutations, d'arbres, de graphes, d'hypergraphes, etc. A nouveau, les applications de l'apprentissage structurel sont nombreuses incluant, par exemple, la décomposition analytique en traduction automatique de la langue, l'appariement de formes en reconnaissance d'images, ou le classement de produits dans les systèmes de recommendation. Certaines applications combinent à la fois l'apprentissage relationnel et la prédiction structurelle, comme la prédiction de repliement dans les protéines (Turcotte *et al.*, 2001). Même si des progrès considérables ont

été réalisés pour la prédiction de certaines classes de structures, comme les arbres (e.g. Koo *et al.*, 2007) et les permutations (e.g. Helmbold et Warmuth, 2009), il reste encore de multiples questions ouvertes sur les classes de graphes et d'hypergraphes (Vembu, 2009).

Apprentissage multi-agents. Un système multi-agents comporte plusieurs agents, chacun pouvant percevoir et agir dans un environnement commun (Shoham et Leyton-Brown, 2009). Les origines de l'apprentissage multi-agents remontent à celles de la théorie des jeux, avec des algorithmes comme *fictitious play* utilisés pour prédire des équilibres (Brown, 1951). Bien qu'il soit apparu assez récemment en intelligence artificielle, l'apprentissage multi-agents a fait l'objet d'une attention considérable cette dernière décennie, avec des applications incluant l'allocation de ressources, la gestion d'enchères, l'analyse financière et, bien entendu, les divers jeux de stratégie. Malgré ce florilège de résultats, Shoham *et al.* (2007) soulignent que de nombreuses questions restent ouvertes au niveau de la modélisation ; la difficulté réside dans le fait que pour chaque agent, l'environnement est, en général, perçu comme partiellement observable, dynamique et séquentiel. Considérons par exemple le modèle des *jeux stochastiques* ou *jeux de Markov* qui généralisent les processus de décision Markoviens au cadre multi-agents (Littman, 1994). Même dans le cas coopératif d'une équipe de joueurs ayant la même récompense, apprendre à jouer un équilibre n'est pas systématiquement la meilleure stratégie puisque certaines jeux d'équipe peuvent contenir de multiples équilibres dont certains sont sous-optimaux. Le défi augmente dans le cas non-coopératif où les joueurs n'ont pas toujours la même récompense : chaque agent doit à la fois découvrir le comportement des autres agents tout en cherchant à maximiser ses propres récompenses. Avec de telles considérations, nous ne pouvons qu'imaginer toute la difficulté d'étendre des modèles comme PAC-MDP (Strehl *et al.*, 2009) ou KWIK-MDP (Li *et al.*, 2011) aux jeux stochastiques.

References

ALEKHNOVICH, M., BRAVERMAN, M., FELDMAN, V., KLIVANS, A. et PITASSI, T. (2008). The complexity of properly learning simple concept classes. *J. Comput. Syst. Sci.*, 74(1):16–34.

ANDREAE, J. (1969). Learning machines : A unified view. *In* MEETHAM, A. et HUDSON, R., éditeurs : *Encyclopedia of linguistics, information, and control*. Pergamon Press.

ANGLUIN, D. (1988). Queries and concept learning. *Mach. Learn.*, 2(4):319–342.

ANGLUIN, D. (1990). Negative results for equivalence queries. *Mach. Learn.*, 5:121–150.

ANGLUIN, D., ASPNES, J., CHEN, J. et WU, Y. (2009). Learning a circuit by injecting values. *J. Comput. Syst. Sci.*, 75(1):60–77.

ANGLUIN, D., ASPNES, J. et REYZIN, L. (2010). Optimally learning social networks with activations and suppressions. *Theor. Comput. Sci.*, 411(29-30):2729–2740.

ANGLUIN, D., FRAZIER, M. et PITT, L. (1992). Learning conjunctions of Horn clauses. *Mach. Learn.*, 9:147–164.

ANTOS, A., KÉGL, B., LINDER, T. et LUGOSI, G. (2002). Data-dependent margin-based generalization bounds for classification. *J. Mach. Learn. Res.*, 3:73–98.

ARIAS, M. et KHARDON, R. (2002). Learning closed Horn expressions. *Inf. Comput.*, 178(1): 214–240.

AUER, P., CESA-BIANCHI, N. et FISCHER, P. (2002a). Finite-time analysis of the multiarmed bandit problem. *Mach. Learn.*, 47(2-3):235–256.

AUER, P., CESA-BIANCHI, N., FREUND, Y. et SCHAPIRE, R. (2002b). The nonstochastic multiarmed bandit problem. *SIAM J. Comput.*, 32(1):48–77.

AUER, P. et WARMUTH, M. (1998). Tracking the best disjunction. *Mach. Learn.*, 32(2):127–150.

BAKIR, G., T. HOFMANN, B. S., SMOLA, A., TASKAR, B. et VISHWANATHAN, S., éditeurs (2007). *Predicting Structured Data*. MIT Press.

BANOS, A. (1998). On pseudo-games. *Ann. Math. Stat.*, 39:1932–1945.

BARTLETT, P. et MENDELSON, S. (2002). Rademacher and gaussian complexities : Risk bounds and structural results. *J. Mach. Learn. Res.*, 3:463–482.

BECERRA-BONACHE, L., DE LA HIGUERA, C., JANODET, J.-C. et TANTINI, F. (2008). Learning balls of strings from edit corrections. *J. Mach. Learn. Res.*, 9:1841–1870.

BELONGIE, S., MALIK, J. et PUZICHA, J. (2002). Shape matching and object recognition using shape contexts. *IEEE T. Pattern Anal.*, 24(4):509–522.

BLUM, A. (1997). Empirical support for winnow and weighted-majority algorithms : Results on a calendar scheduling domain. *Mach. Learn.*, 26(1):5–23.

BLUM, A., HELLERSTEIN, L. et LITTLESTONE, N. (1995). Learning in the presence of finitely or infinitely many irrelevant attributes. *J. Comput. Syst. Sci.*, 50(1):32–40.

BOUSQUET, O. et WARMUTH, M. (2002). Tracking a small set of experts by mixing past posteriors. *J. Mach. Learn. Res.*, 3:363–396.

BRAFMAN, R. et TENNENHOLTZ, M. (2002). R-MAX - a general polynomial time algorithm for near-optimal reinforcement learning. *J. Mach. Learn. Res.*, 3:213–231.

BROWN, G. W. (1951). Iterative solution of games by fictitious play. *In* KOOPMANS, T., éditeur : *In Activity Analysis of Production and Allocation*, pages 374–376. Wiley.

BSHOUTY, N. (1995). Exact learning Boolean functions via the monotone theory. *Inform. Comput.*, 123(1):146–153.

BUBECK, S. et CESA-BIANCHI, N. (2012). Regret analysis of stochastic and nonstochastic multi-armed bandit problems. *Foundations and Trends in Machine Learning*, 5(1):1–122.

CASTRO, J. et BALCÁZAR, J. L. (1995). Simple PAC learning of simple decision lists. *In Proceedings of the 6th International Conference on Algorithmic Learning Theory (ALT'95)*, pages 239–248.

CESA-BIANCHI, N., CONCONI, A. et GENTILE, C. (2004). On the generalization ability of on-line learning algorithms. *IEEE Transactions on Information Theory*, 50(9):2050–2057.

CESA-BIANCHI, N., FREUND, Y., HAUSSLER, D., HELMBOLD, D., SCHAPIRE, R. et WARMUTH, M. (1997). How to use expert advice. *Journal of the ACM*, 44(3):427–485.

CESA-BIANCHI, N. et LUGOSI, G. (2006). *Prediction, Learning, and Games*. Cambridge.

CHAN, P. et LIPPMANN, R. (2006). Mach. learn. for computer security. *J. Mach. Learn. Res.*, 7:2669–2672.

CHAPELLE, O., SCHÖLKOPF, N. et ZIEN, A., éditeurs (2006). *Semi-Supervised Learning*.

MIT Press.

COHEN, W., SCHAPIRE, R. et SINGER, Y. (1999). Learning to order things. *J. Artif. Intell. Res.*, 10:243–270.

COLLINS, M., GLOBERSON, A., KOO, T., CARRERAS, X. et BARTLETT, P. L. (2008). Exponentiated gradient algorithms for conditional random fields and max-margin markov networks. *J. Mach. Learn. Res.*, 9:1775–1822.

CORNUÉJOLS, A., MICLET, L. et HATON, J.-P. (2010). *Apprentissage Artificiel : Concepts et Algorithmes*. Eyrolles, 2 édition.

CRAMMER, K., DEKEL, O., KESHET, J., SHALEV-SHWARTZ, S. et SINGER, Y. (2006). Online passive-aggressive algorithms. *J. Mach. Learn. Res.*, 7:551–585.

CRAMMER, K. et SINGER, Y. (2003a). A family of additive online algorithms for category ranking. *J. Mach. Learn. Res.*, 3:1025–1058.

CRAMMER, K. et SINGER, Y. (2003b). Ultraconservative online algorithms for multiclass problems. *J. Mach. Learn. Res.*, 3:951–991.

DE RAEDT, L. (2008). *Logical and Relational Learning*. Springer.

DUDA, R., HART, P. et STORK, D. (2001). *Pattern Classification*. Wiley.

EVEN-DAR, E. et MANSOUR, Y. (2003). Learning rates for Q-learning. *J. Mach. Learn. Res.*, 5:1–25.

FEL'DBAUM, A. A. (1961). Dual-control theory. *Automation and Remote Control*, 21:874–880.

FELDMAN, V. (2009). Hardness of approximate two-level logic minimization and PAC learning with membership queries. *J. Comp. Syst. Sci.*, 75(1):13–26.

FISHER, R. (1930). *Statistical Methods for Research Workers*. Oliver and Boyd, London, third édition.

FRAZIER, M. et PITT, L. (1996). Classic learning. *Mach. Learn.*, 25(2-3):151–193.

FREUND, Y. (1995). Boosting a weak learning algorithm by majority. *Inform. Comput.*, 121(2):256–285.

FREUND, Y. et SCHAPIRE, R. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *J. Comput. Syst. Sci.*, 55(1):119–139.

FREUND, Y. et SCHAPIRE, R. (1999). Large margin classification using the perceptron algorithm. *Mach. Learn.*, 37(3):277–296.

GETOOR, L. et TASKAR, B. (2007). *Introduction to Statistical Relational Learning*. MIT Press.

GITTINGS, J. (1989). *Multi-Armed Bandit Allocation Indices*. Wiley.

GROVE, A. J., LITTLESTONE, N. et SCHUURMANS, D. (2001). General convergence results for linear discriminant updates. *Mach. Learn.*, 43(3):173–210.

GYÖRGY, A., LINDER, T., LUGOSI, G. et OTTUCSÁK, G. (2007). The on-line shortest path problem under partial monitoring. *J. Mach. Learn. Res.*, 8:2369–2403.

HAUSSLER, D. (1989). Learning conjunctive concepts in structural domains. *Mach. Learn.*, 4:7–40.

HAUSSLER, D. (1992). Decision theoretic generalizations of the PAC model for neural net and other learning applications. *Inform. Comput.*, 100(1):78–150.

HELMBOLD, D. et WARMUTH, M. (2009). Learning permutations with exponential weights. *J. Mach. Learn. Res.*, 10:1705–1736.

HERBSTER, M. et WARMUTH, M. (1998). Tracking the best expert. *Mach. Learn.*, 32(2):151–178.

HERBSTER, M. et WARMUTH, M. K. (2001). Tracking the best linear predictor. *J. Mach. Learn. Res.*, 1:281–309.

HÖFFGEN, K.-U., SIMON, H.-U. et HORN, K. V. (1995). Robust trainability of single neurons. *J. Comput. Syst. Sci.*, 50(1):114–125.

JACKSON, J. (1997). An efficient membership-query algorithm for learning DNF with respect to the uniform distribution. *J. Comput. Syst. Sci.*, 55(3):414–440.

KAELBLING, L. P. (2010). *Recent Advances in Reinforcement Learning*. Springer.

KAKADE, S. (2003). *On the Sample Complexity of Reinforcement Learning*. Thèse de doctorat, Gatsby Computational Neuroscience Unit, Univesity College London.

KEARNS, M., SCHAPIRE, R. et SELLIE, L. (1994). Toward efficient agnostic learning. *Mach. Learn.*, 17(2-3):115–141.

KEARNS, M. et SINGH, S. (2002). Near-optimal reinforcement learning in polynomial time. *Mach. Learn.*, 49(2-3):209–232.

KEARNS, M. et VAZIRANI, U. (1994). *An Introduction to Computational Learning Theory*. MIT Press.

KERSTING, K. (2006). *An Inductive Logic Programming Approach to Statistical Relational Learning*, volume 148 de *Frontiers in Artificial Intelligence and Applications*. IOS Press.

KHARDON, R. (1999). Learning function-free Horn expressions. *Mach. Learn.*, 37(3):241–275.

KHARITONOV, M. (1993). Cryptographic hardness of distribution-specific learning. *In Proceedings of the 25th Annual ACM Symposium on Theory of Computing (STOC'93)*, pages 372–381. ACM.

KIVINEN, J. et WARMUTH, M. (1997). Exponentiated gradient versus gradient descent for linear predictors. *Inform. Comput.*, 132(1):1–63.

KOLLER, D. et FRIEDMAN, N. (2009). *Probabilistic Graphical Models*. MIT Press.

KOO, T., GLOBERSON, A., CARRERAS, X. et COLLINS, M. (2007). Structured prediction models via the matrix-tree theorem. *In Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL'07)*, pages 141–150.

LAFFERTY, J., MCCALLUM, A. et PEREIRA, F. (2001). Conditional random fields : Probabilistic models for segmenting and labeling sequence data. *In Proceedings of the Eighteenth International Conference on Machine Learning (ICML'01)*, pages 282–289.

LANGFORD, J. et BLUM, A. (2003). Microchoice bounds and self bounding learning algorithms. *Mach. Learn.*, 51(2):165–179.

LI, L., LITTMAN, M., WALSH, T. et STREHL, A. (2011). Knows what it knows : a framework for self-aware learning. *Mach. Learn.*, 82(3):399–443.

LITTLESTONE, N. (1988). Learning quickly when irrelevant attributes abound : A new linear-threshold algorithm. *Mach. Learn.*, 2(4):285–318.

LITTLESTONE, N. (1989). From on-line to batch learning. *In Proceedings of the Second Annual Workshop on Computational Learning Theory (COLT'89)*, pages 269–284.

LITTLESTONE, N. et WARMUTH, M. K. (1989). The weighted majority algorithm. *In Proceedings of the 30th Annual IEEE Symposium on Foundations of Computer Science (FOCS'89)*, pages 256–261. IEEE.

LITTMAN, M. L. (1994). Markov games as a framework for multi-agent reinforcement learning. *In Proceedings of the Eleventh International Conference on Machine Learning (ICML'94)*, pages 157–163. Morgan Kaufmann.

LONG, P. (1999). The complexity of learning according to two models of a drifting environment. *Mach. Learn.*, 37(3):337–354.

MAASS, W. et TURAN, G. (1994). How fast can a threshold gate learn. *In Computational Learning Theory and Natural Learning System : Constraints and Prospects*, pages 381–414. MIT Press.

MATUSOV, E., ZENS, R. et NEY, H. (2004). Symmetric word alignments for statistical machine translation. *In Proceedings of the Twentieth International Conference on Computational Linguistics (COLING'04)*, pages 219–225.

NG, A., D. et RUSSELL, S. (1999). Policy invariance under reward transformations : Theory and application to reward shaping. *In Proceedings of the 16th International Conference on Machine Learning (ICML'99)*, pages 278–287.

NOVIKOV, A. (1962). On convergence proofs on Perceptrons. *In Proceedings of the Symposium of the Mathematical Theory of Automata*, volume XII, pages 615–622. Wiley.

PICCOLBONI, A. et SCHINDELHAUER, C. (2001). Discrete prediction games with arbitrary feedback and loss. *In Proceedings of the 14th Annual Conference on Computational Learning Theory (COLT'01)*, pages 208–223.

PITRAT, J. (2009). *Artificial Beings : The Conscience of a Conscious Machine*. ISTE Ltd and John Wiley & Sons.

PUTERMAN, M. (1994). *Markov Decision Processes : Discrete Stochastic Dynamic Programming*. Wiley.

RICHARDSON, M. et DOMINGOS, P. (2006). Markov logic networks. *Mach. Learn.*, 62(1-2):107–136.

ROBBINS, H. (1952). Some aspects of the sequential design of experiments. *Bulletin of the American Mathematical Society*, 55:527–535.

ROSENBLATT, F. (1958). The Perceptron : a probabilistic model for information storage and organization in the brain. *Psychological Review*, 65:386–408.

RUSSELL, S. et NORVIG, P. (2003). *Artificial Intelligence : A Modern Approach*. Prentice Hall Series in Artificial Intelligence, 3 édition.

RUSTICHINI, A. (1999). Minimizing regret : The general case. *Game. Econ. Behav.*, 29:224–243.

SCHAPIRE, R. (1990). The strength of weak learnability. *Mach. Learn.*, 5:197–227.

SHALEV-SHWARTZ, S. et SINGER, Y. (2007). A primal-dual perspective of online learning algorithms. *Mach. Learn.*, 69(2-3):115–142.

SHOHAM, Y. et LEYTON-BROWN, K. (2009). *Multiagent Systems : Algorithmic, Game-*

Theoretic, and Logical Foundations. Cambridge.

SHOHAM, Y., POWERS, R. et GRENAGER, T. (2007). If multi-agent learning is the answer, what is the question ? *Artif. Intell.*, 171(7):365–377.

SIMON, H.-U. (1995). Learning decision lists and trees with equivalence-queries. *In Proceedings of the 2nd European Conference on Computational Learning Theory (EuroCOLT'95)*, pages 322–336.

STREHL, A., LI, L. et LITTMAN, M. (2009). Reinforcement learning in finite MDPs : PAC analysis. *J. Mach. Learn. Res.*, 10:2413–2444.

STREHL, A., LI, L., WIEWIORA, E., LANGFORD, J. et LITTMAN, M. L. (2006). PAC model-free reinforcement learning. *In Proceedings of the Twenty-Third International Conference on Machine Learning (ICML'06)*, pages 881–888. ACM.

SUTTON, R. et BARTO, A. (1998). *Reinforcement Learning : An Introduction*. MIT Press.

SZEPESVÁRI, C. (1997). The asymptotic convergence-rate of q-learning. *In Advances in Neural Information Processing Systems 10, (NIPS'97)*, pages 1064–1070. MIT Press.

TASKAR, B., LACOSTE-JULIEN, S. et JORDAN, M. I. (2006). Structured prediction, dual extragradient and Bregman projections. *J. Mach. Learn. Res.*, 7:1627–1653.

THEODORIDIS, S. et KOUTROUMBAS, K. (2009). *Pattern Recognition*. Elsevier, 4 édition.

TURCOTTE, M., MUGGLETON, S. et STERNBERG, M. (2001). The effect of relational background knowledge on learning of protein three-dimensional fold signatures. *Mach. Learn.*, 43(1/2):81–95.

VALIANT, L. G. (1984). A theory of the learnable. *Commun. ACM*, 27(11):1134–1142.

VAPNIK, V. (1998). *Statistical Learning Theory*. Wiley.

VAPNIK, V. et CHERVONENKIS, A. (1971). On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and its Applications*, 16(2):264–280.

VAPNIK, V. et CHERVONENKIS, A. (1974). *Teoriya raspoznavaniya obrazov : Statisticheskie problemy obucheniya. (Russe)* Theory of pattern recognition : Statistical problems of learning. Moscow : Nauka.

VEMBU, S. (2009). *Learning to Predict Combinatorial Structures*. Thèse de doctorat, Department of Computer Science, University of Bonn, Germany.

VOVK, V. (1990). Aggregating strategies. *In Proceedings of the Third Annual Workshop on Computational Learning Theory (COLT'90)*, pages 371–386.

WATKINS, C. (1989). *Learning from Delayed Rewards*. Thèse de doctorat, Cambridge University.

WATKINS, C. et DAYAN, P. (1992). Q-learning. *Mach. Learn.*, 8:279–292.

WINDROW, B. et HOFF, M. (1960). Adaptive switching circuits. *In IRE WESCON Convention Record*, pages 96–104.

WITTEN, I. (1977). An adaptive optimal controller for discrete-time markov environments. *Information and Control*, 34(4):286–295.

WOODMWARD, A. et NEEDHAM, A. (2008). *Learning and the Infant Mind*. Oxford Univ. Press.

XU, R. et WUNSCH, D. (2008). *Clustering*. IEEE Press Series on Computational Intelligence. Wiley-IEEE Press.