Chapitre 1

Apprentissage symbolique de concepts

1.1 Introduction

Étant données une tâche T à effectuer par un programme informatique donné, et une mesure de performance par rapport à cette tâche P ; on dit que ce programme apprend à partir d'une expérience E si la mesure P augmente avec l'expérience.

Par exemple dans le domaine de la *Reconnaissance de manuscrits* : T est la tâche de reconnaissance et de classement des lettres manuscrites données en entrée. P est le pourcentage des mots classés correctement par le programme. E est une base de données contenant des mots avec une fonction qui permet d'effectuer la classification.

Pour mettre en œuvre un système d'apprentissage automatique, il faut choisir l'expérience qui permet au système d'apprendre ainsi que la fonction cible à apprendre. L'expérience est souvent exprimée en terme d'un ensemble d'exemples qui peuvent, dans certains domaines, être appuyés par une théorie qui décrit quelques aspects du domaine en question. La fonction cible est apprise à partir de l'ensemble d'exemples, chaque exemple serait représenté par le couple (b,f(b)), b étant les données de l'exemple et f(b) est la valeur de la fonction cible sur cette donnée.

Nous présentons dans la suite, un cas particulier de système d'apprentissage dans lequel la fonction cible étant l'appartenance ou non d'un exemple donné à un concept donné.

En d'autres mots, l'objet de ce chapitre est de présenter deux algorithmes (Find-S et Candidate-Elimination) qui permettent de décrire des concepts appris à partir d'un ensemble contenant d'exemples positifs et négatifs. Cette description sera formalisée en utilisant la terminologie des espaces d'hypothèses et de versions. Ce chapitre permettra à l'élève d'avoir un premier aperçu de l'apprentissage automatique à partir de données en utilisant un langage simple de description de concept qu'est celui des hypothèses.

1.2 L'apprentissage de concepts

La plus part des systèmes d'apprentissage ont comme objectif d'apprendre de concepts généraux à partir d'exemples spécifiques. Chaque concept peut être représenté (par extension) par le sous-ensemble d'exemples qu'il représente, ou (par intension) par une fonction booléenne qui décrit l'appartenance au concept en fonction de certaines caractéristiques.

Par conséquence, on peut dire que l'apprentissage de concept consiste à inférer une fonction booléenne à partir d'un ensemble d' exemples contenant chacun les entrée et la sortie correspondante.

1.2.1 Notations

Nous adoptons dans la suite la terminologie suivante qui nous permettra de décrie la tâche de l'apprentissage de concept.

Une instance étant donné un ensemble d'attribut, chacun ayant un domaine de valeurs, une instance est une valuation possible de cet ensemble d'attributs dans leurs domaines respectifs.

La fonction concept Soit X un ensemble donné d'instances, la fonction concept : $c(X) \rightarrow \{0, 1\}$ est une fonction qui permet d'effectuer une appartenance à une instance $x \in X$ au concept en cours d'apprentissage.

L'ensemble d'apprentissage Est un ensemble d'instances munis de leurs valeurs d'appartenance au concept.

L'espace d'hypothèses Une hypothèse est définie comme étant une conjonction de contraintes sur la liste d'attributs : <val1,..,valN>. Une contrainte sur un attribut peut être une valeur appartenant au domaine de l'attribut ou, peut prendre la valeur ? ou ϕ. La valeur ? signifie que l'attribut en question peut prendre n'importe quelle valeur tandis que la valeur ϕ signifie que l'attribut ne peut prendre aucune valeur dans son domaine. L'espace d'hypothèses comprend toutes les hypothèses possibles pour décrire un domaine donné.

Par exemple, la table 1.1 nous montre un ensemble d'apprentissage. Une instance est décrite par les attributs : $\{CIEL, TEMP, HUMI, VENT\}$ définis respectivement sur les domaines suivants : $D(CIEL) = \{ensoleille, couvert, pluvieux\}$, $D(TEMP) = \{eleve, moyenne, basse\}$, $D(HUMI) = \{forte, normale, moyenne\}$,$D(VENT) = \{oui, non\}$. Cette table comporte deux exemples positifs (ayant c(x)=1) et deux exemples négatifs (avec c(x)=0).

Une hypothèse possible serait h=<ensoleillé, élevé,forte, ?> qui couvre les deux exemples négatifs dans l'ensemble d'apprentissage.

L'hypothèse < ?, ?, ?, ?> est appelée l'hypothèse la plus générale car elle représente tous les exemples possibles tandis que toute hypothèse contenant la valeur ϕ représente l'ensemble vide, autrement dit elle ne satisfait[1] aucun exemple.

[1] Un hypothèse h satisfait un exemple x ssi $h(x) = 1$

NUM	CIEL	TEMP.	HUMI.	VENT	Concept
1	ensoleillé	élevé	forte	non	0
2	ensoleillé	élevé	forte	oui	0
3	couvert	élevé	normale	non	1
4	pluvieux	moyenne	normale	non	1

TAB. 1.1 – Description des conditions météorologiques et du concept JouerFoot

Schéma de description d'une tâche La tâche à apprendre par le système peut être décrite par le schéma suivant : étant donné un ensemble d'instances X, munie de la fonction cible $c(X) \rightarrow \{0, 1\}$, appelé ensemble d'apprentissage et contenant des exemples positifs et négatifs, il s'agit de déterminer $h \in H$ où H est l'espace d'hypothèses possible tel que h(x)=c(x) $\forall x \in X$

Autrement dit, il faut que l'hypothèse inférée classe correctement les exemples positifs et rejette les exemples négatifs.

Ce type d'apprentissage appartient à l'apprentissage par induction qui exprime l'hypothèse suivante : *une hypothèse trouvée à partir d'un ensemble d'exemples contenant des exemples suffisamment représentatifs du domaine doit classer assez correctement les nouveaux exemples*.

Relation d'ordre sur l'espace d'hypothèses Il existe une relation d'ordre partielle sur l'espace d'hypothèses qui nous permettra dans la suite de trouver un treillis d'hypothèses consistantes. Cette relation est basée sur la relation PGE (Plus général ou Égal) qu'on définira ainsi : Soit h_j,h_k deux hypothèses définies sur X. h_j est PGE que h_k : $(h_j \geq h_k)$ ssi $(\forall x \in X)[h_k(x) = 1 \rightarrow h_j(x) = 1]$.

De même,h_j est PG (plus général strictement) que h_k $(h_j > h_k)$ ssi $(h_j \geq h_k) \wedge (h_k \ngeq h_j)$.

1.3 L'algorithme Find-S

Nous présentons dans cette section l'algorithme Find-S dont l'objectif est de trouver l'hypothèse la plus spécifique qui satisfait tous les exemples positifs dans l'ensemble d'apprentissage.

1. Initialiser h à $< \phi, .., \phi >$, l'hypothèse la plus spécifique.
2. Pour chaque exemple positif x faire
   - Pour toute valeur a_i de contrainte dans h
     - Si a_i est satisfait par x Alors (Ne rien faire).
     - Sinon, remplacer a_i dans h par la contrainte suivante la plus générale qui satisfait x.
3. Retourner h.

Par exemple, en appliquant Find-S sur les quatre exemples présenté au dessus , nous initialisons h par $h =< \phi, \phi, \phi, \phi >$, après lecture de l'exemple numéro 3, nous aurions : $h =< couvert, eleve, forte, non >$; et après lecture de l'exemple 4, nous aurions : $h =<?, ?, normale, non >$.

L'inconvénient majeur de cet algorithme est qu'il ignore complètement les exemples négatifs. Par conséquence, il est possible d'avoir une hypothèse qui satisfait tous les exemples positifs mais qui

ne rejette pas obligatoirement tous les exemples négatifs car aucune vérification n'est effectuée dans l'algorithme. Autrement dit, l'algorithme Find-S est incapable dans ce cas de retourner une hypothèse vide.

D'un autre côté, Find-S trouve l'hypothèse la plus spécifique qui couvre les exemples positifs sachant qu'il peut exister d'autres hypothèses plus générales qui seront consistantes avec l'ensemble d'apprentissage. Nous présentons dans la suite un algorithme permettant de construire l'espace des hypothèses consistantes avec un ensemble donné d'exemples. Cet espace est appelé *espace de version*.

1.4 L'espace de version

La construction de l'espace de version est fondée sur la recherche des *hypothèses consistantes* avec l'ensemble d'apprentissage D dans H. Une hypothèse h est dite *consistante* avec un ensemble d'apprentissage donné ssi $h(x) = c(x)$ pour chaque exemple (x,c(x)) dans D. L'espace de version est donné par

$$VS_{H,D} = \{h \in H | Consistant(h, D)\}$$

Nous définissons dans la suite la *limite générale* et *la limite spécifique* d'un espace d'hypothèses H par rapport à un ensemble d'apprentissage D, notés respectivement par G et S.

La limite générale G est l'ensemble des hypothèses les plus générales qui seront consistantes avec D et est donnée par

$$G = \{g \in H | Consistant(g, D) \wedge (\nexists g' \in H)[(g' > g) \wedge Consistant(g', D)]\}$$

De même, la limite spécifique S est l'ensemble des hypothèses les plus spécifiques qui seront consistantes avec D et est donnée par

$$S = \{s \in H | Consistant(s, D) \wedge (\nexists s' \in H)[(s' < s) \wedge Consistant(s', D)]\}$$

Théorème 1.4.1 *L'espace de version est donnée par :*

$$VS_{H,D} = \{h \in H | (\exists s \in S)(\exists g \in G) | (s \leq h \leq g)\}$$

Soit h une hypothèse appartenant à l'ensemble $VS_{H,D}$ soit $x \in X$ un exemple positif, par définition s satisfait x, et puisque $h > s$ nous avons $h(x) = 1$. De même, soit $y \in X$ un exemple négatif, g ne satisfait pas y : $g(y) = 0$, par conséquence h(y) ne peut prendre que la valeur 0 car si $h(y) = 1$ alors $g(y) = 1$ (car g est plus général que h).

Nous avons démontré que toute hypothèse appartenant à $VS_{H,D}$ est consistante avec l'ensemble d'apprentissage D. Il reste à démontrer que toute hypothèse consistante appartient à l'ensemble $VS_{H,D}$; cette démonstration est laissée à la charge du lecteur intéresse ! □

1.5 L'algorithme Candidate-Elimination

L'algorithme Candidate-Elimination calcule l'espace de version en trouvant les deux limites G et S. Les deux limites sont initialisées aux hypothèses $<?,..,?>$ et $<\phi,..,\phi>$ respectivement. A chaque lecture d'un exemple positif x la limite S est généralisée d'une façon minimale pour que l'exemple positif soit consistant avec les éléments de S ; de même, les éléments de G non consistants avec x sont supprimés. Un comportement symétrique par rapport à (G, S) est effectué lors de la lecture d'un exemple négatif.

Nous donnons dans la suite l'algorithme détaillé :

1. Initialiser G à $\{<?,..?>\}$
2. Initialiser S à $\{<\phi,..\phi>\}$
3. Si d est un exemple positif
   - Supprimer de G les hypothèses inconsistantes avec d.
   - Pour chaque hypothèse $s \in S$ qui n'est pas consistante avec d faire
     - Supprimer s de S.
     - Ajouter à S toutes le généralisations minimales h de s tel que h soit consistante avec d et il existe une hypothèse $g \in G$ plus général que h.
     - supprimer de S chaque hypothèse plus générale que d'autre hypothèse incluse également dans S.
4. Si d est un exemple négatif
   - Supprimer de S les hypothèses inconsistantes avec d.
   - Pour chaque hypothèse $g \in G$ qui n'est pas consistante avec d faire
     - Supprimer g de G.
     - Ajouter à G toutes le spécialisations minimales h de g tel que h soit consistante avec d et il existe une hypothèse $s \in S$ plus spécifique que h.
     - supprimer de G chaque hypothèse plus spécifique qu'autre hypothèse incluse également dans G.

1.6 Exercices

EXERCICE 3.1 En analysant des actifs financiers côtés en bourse, nous voulons apprendre le concept-cible l'actif distribue des dividendes. Pour ce faire nous avons les exemples du tableau suivant :

No	Prix	Bénéfices	Secteur	Actif	Bourse	Croissance	Dividende
1	Élevé	Élevés	Manufacture	Action	NYSE	Forte	1
2	Élevé	Élevés	Services	Action	NYSE	Forte	1
3	Faible	Faibles	Services	Action	NYSE	Faible	0
2	Élevé	Élevés	Services	Action	Nasdaq	Faible	1

1. Appliquer les algorithmes FIND-S et CANDIDAT-ELIMINATION et discuter les étapes de chaque algorithme.

2. Calculer le nombre d'éléments de l'espace d'hypothèses et de l'espace d'instances.
3. Donner ces mêmes nombres si on rajoute un attribut nouveau qui a k valeurs différentes.
4. Ranger les exemples dans un autre tableau en ordre inverse et appliquer de nouveau l'algorithme CANDIDAT-ELIMINATION. Expliquer pour quelles raisons l'espace des versions final est le même dans les deux cas.
5. Pouvez-vous envisager une stratégie pour ordonner les exemples, afin de réduire le nombre d'opérations nécessaires lors de l'apprentissage du concept ?

EXERCICE 3.2 Nous voulons apprendre le concept voiture familiale japonaise. Pour cela nous avons les exemples suivants :

Pays	Constructeur	Couleur	Année	Type	Exemple
Japon	Honda	Bleue	2000	Familiale	1
Japon	Toyota	Verte	1997	Sportive	0
Japon	Toyota	Bleue	1999	Familiale	1
États-Unis	Chrysler	Rouge	2000	Familiale	0
Japon	Honda	Blanche	2000	Familiale	1

1. Appliquer les algorithmes FIND-S et CANDIDAT-ELIMINATION et discuter les résultats.
2. Ajouter les deux nouveaux exemples suivants :

Pays	Constructeur	Couleur	Année	Type	Exemple
Japon	Toyota	Verte	2000	familiale	1
Japon	Honda	Rouge	1999	Familiale	0

   En appliquant l'algorithme CANDIDAT-ELIMINATION évaluer l'évolution des ensembles S(X) et G(X).

EXERCICE 3.3 Considérons les exemples suivants qui décrivent le concept-cible « couple de gens qui vivent dans la même maison ».

No	Couple	Ensemble
1	(homme, chatain, grand, USA) (femme, noire, petite, USA)	1
2	(homme, chatain, petit, France) (femme, noire, petite, USA)	1
3	(femme, chatain, grande, Allemagne) (femme, noire, petite, Inde)	0
4	(homme, chatain, grand, Irlande) (femme, chatain, petite, Irlande)	1

1. Appliquer l'algorithme CANDIDAT-ELIMINATION.
2. Considérons l'exemple suivants :
   Indiquer l'évolution des ensembles S(X) et G(X).

No	Couple	Ensemble
5	(homme, noir, petit, Portugal) (femme, blonde, grande, Inde)	1