Modèles d'Apprentissage Artificiel

Master 2 : Intelligence Artificielle

Pr. Mustapha Bourahla

Introduction

- 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.

Le problème

- 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.
- 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'utilisateur 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.

Difficulté d’un problème d’apprentissage

- La difficulté d’un problème d’apprentissage est caractérisée par deux sources de complexité:
    1) La première correspond au nombre de cycles durant lesquels la performance de l’agent reste sous-optimale pour la tâche de décision donnée.
    2) 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.
- 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,
- 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
- l'appariement de formes en reconnaissance d'images sont des problèmes de classification structurelle considérés comme particulièrement difficiles en apprentissage.

Actions (continue)

- Même des actions simples peuvent avoir un impact sur la difficulté de l'apprentissage selon la manière dont elles influent l'environnement.
  - 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, sont des exemples caractéristiques de tâches de décision séquentielle.

Feedback

- En apprentissage supervisé, le feedback correspond à l'action qu'aurait du choisir l'agent selon l'observation courante (Le type de feedback définit le mode d'apprentissage).
- Ainsi l'environnement est un superviseur qui corrige les erreurs de l'agent en fournissant la bonne étiquette à chaque observation.
- 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.

Feedback (continue)

- Cette représentation peut ensuite être utilisée comme stratégie pour étiqueter de nouvelles observations selon la structure induite.
- 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.

Feedback (continue)

- 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.
- Dans ce cadre s'ajoute une difficulté supplémentaire caractérisée par le fameux dilemme exploration-exploitation:
  - 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.

Exercice n°1

a) Quelles sont les sources du problème d'apprentissage?

b) Donner une définition du modèle d'apprentissage

c) Expliquer comment les observations, les actions et le feedback peuvent influer sur la difficulté de l'apprentissage

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 pour définir des modèles particuliers d'apprentissage supervisé.

Apprentissage supervisé (continue)

- Un problème d'apprentissage supervisé comprend
  - un espace X d'observations, appelées aussi instances ou entrées,
  - et un espace Y d'actions ou sorties. Notons $Z = X \times Y$.
- Tout élément $z \in Z$ est appelée exemple et toute séquence d'exemples $z = (z_1 , \cdots , z_m)$ est appelée ensemble d'entraînement.
- Un ensemble d'entraînement est donc une "liste" d'exemples.
- les entrées et sorties sont décrites comme des vecteurs de dimension non nulle et finie.

Apprentissage supervisé (continue)

- En plus des entrées et sorties, un problème d'apprentissage supervisé comprend généralement deux classes de fonctions de X dans Y, la classe d'hypothèses H et la classe cible H^* .
- L'apprenant cherche à prédire le comportement de son environnement à partir de stratégies h sélectionnées dans H. La classe cible H^* sert de référence pour mesurer la performance des hypothèses
- L'objectif est de trouver par entraînement une hypothèse $h \in H$ dont la performance est proche de la meilleure fonction cible $h^* \in H^*$.

Performance des hypothèses

- La performance des hypothèses est mesurée par une fonction l : $H \times X \times Y \rightarrow R^{+}$, appelée fonction de perte (loss).
- Elle associe à toute hypothèse h dans H, toute entrée x dans X et toute sortie y dans Y un réel non négatif l(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 $z = (z_1 , \cdots , z_m)$ selon l est définie par la perte cumulée: $\sum_{t=1}^{m} l(h, z_t)$.

Fonctions de perte

- Parmi les fonctions de perte les plus utilisées dans les problèmes unidimensionnels, nous pouvons mentionner la fonction de perte discrète ou zero-un définie par : $l_{DIS}(h, x, y) = I_{h(x)\neq y}$ et la fonction de perte quadratique, définie par : $l_{SQ}(h, x, y) = \frac{1}{2}\|h(x) - y\|^2$
- Dans le cadre plus général des problèmes multidimensionnels ou structurels, les fonctions de perte sont souvent définies par des fonctions convexes sur H qui évaluent la distance entre les structures h(x) et y

Schéma de représentation

- Nous appelons schéma de représentation la donnée d'un ensemble Ω appelé classe de représentation, d'une mesure de complexité $f : \Omega \rightarrow R^+$ 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 H.
- Remarque: En apprentissage statistique, f est souvent appelée fonction de régularisation.
- 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.

Composants d'un problème d'apprentissage

- L'espace des hypothèses engendrées par Ω est dénoté H_Ω , c'est à dire $H_\Omega = \{h_\omega : \omega \in \Omega\}$.
- La longueur de description d'une hypothèse $h \in H_\Omega$ 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 X ,
  - un espace de décisions Y,
  - un schéma de représentation (Ω, f) engendrant l'espace H_Ω mesuré par f ,
  - un espace de fonctions cibles 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

- Reprenons le scénario sur le filtrage anti-spam. Rappelons que l'espace des entrées est $X = \{0, 1\}^n$ et l'espace des sorties est $Y = \{-1, +1\}$.
- 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 R^n et θ un scalaire de R.
- L'hypothèse associée à $\omega = (w, \theta)$ est définie par $h_\omega(x) = sgn(<w, x> - \theta)$ où $<w, x>$ désigne le produit scalaire entre w et x et sgn(z) est la fonction qui retourne le signe 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 $\omega \in R^{n+1}$.

Exemple (continue)

- La performance de l'agent est le nombre d'erreurs de prédiction qu'il fait sur toute séquence de courriels

$$z = ((x_1, y_1), \cdots, (x_m, y_m)).$$

- Ce nombre d'erreurs est m mesuré par la perte discrète cumulée : $\sum_{\{t=1\}}^{m} l_{DIS}(h_{w_t}, x_t, y_t)$ où ω_t est la fonction linéaire à seuil maintenue par l'agent au tour t.

Exercice n° 2

a) Donner avec explication les composants du problème de l'apprentissage supervisé de porte logique XOR

b) TP: Ecrire un programme Python pour montrer ces composants

Classes de concepts

- Afin d'analyser et comparer les modèles d'apprentissage étudiés, 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 $\{x_1 , \cdot \cdot \cdot , x_n\}$, rappelons qu'un littéral est une variable x_i ou sa négation $\neg x_i$, 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.

Exemple

- Une fonction h : $\{0, 1\}^n \rightarrow \{-1, +1\}$ est dite linéaire à seuil si elle est représentable par une paire (w, θ) où $w \in R^n$ et $\theta \in R$, telle que $h(x) = \text{sgn}(<w, x> - \theta)$.
- La fonction linéaire h est dite booléenne à seuil si $w \in \{0, 1\}^n$ et $\theta \in N$.
- 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 $w = 1$ et $\theta = \left\lfloor \frac{n}{2} \right\rfloor$.

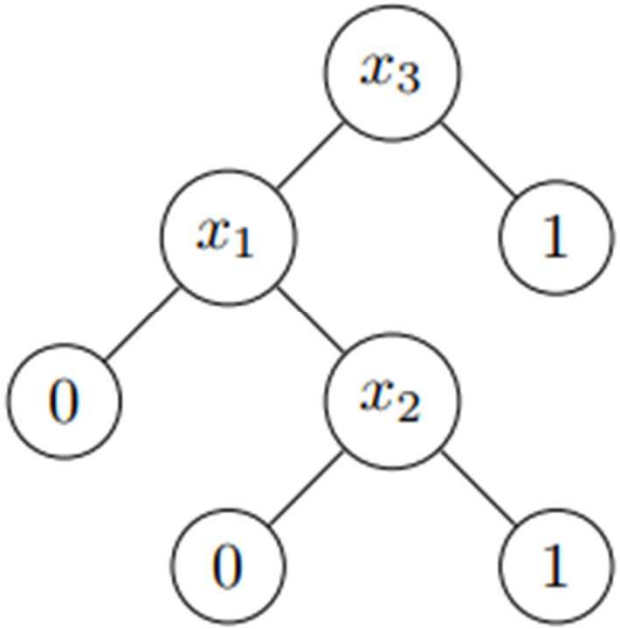
Arbre de décision

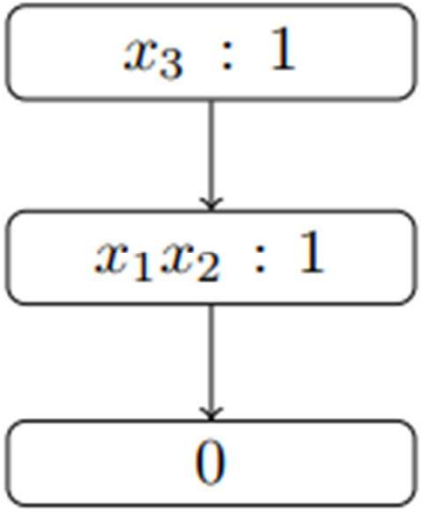
- Un arbre de décision (booléen) est un arbre binaire dont chaque feuille est étiquetée par 0 ou 1 et chaque nœud interne est étiqueté par un index $i \in \{1, \cdots, n\}$ pointant sur deux fils.
- La classification d'une instance x selon l'hypothèse associée à l'arbre de décision ω est déterminée en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

Liste de décision

- Une liste de décision est une séquence de règles $<t, v>_d = ((t_1 , v_1), \cdot \cdot \cdot , (t_d , v_d))$ 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 .

Un arbre de décision et une liste de décision pour la fonction $x_1 \wedge x_2 \vee x_3$





Exercice n° 3

- Donner l’arbre de décision et son correspondante liste de décision de la formule:

$$x_1 \wedge x_2 \vee x_3 \wedge x_4$$

Apprentissage exact

- Appelé aussi apprentissage avec requêtes, 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 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.
- L'apprentissage avec requêtes sera expliqué sur des problèmes de classification binaire (Y = {0, 1}), mais gardons à l'esprit que ce modèle peut se généraliser à des tâches multi-classes ou multidimensionnelles.

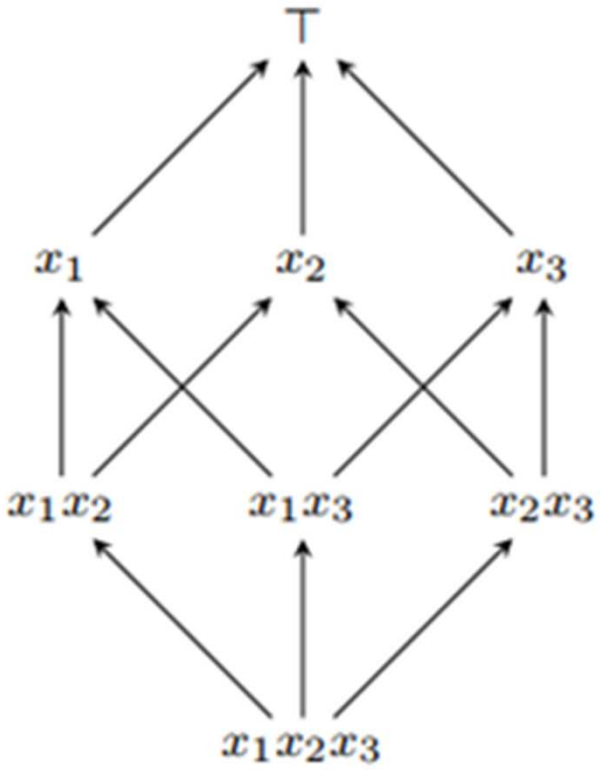
Définitions

- Un problème d'apprentissage de concepts consiste en un espace d'instances X, un espace d'hypothèses H_Ω engendré à partir d'une classe de représentation Ω, et un espace de concepts cibles 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 $x \in X$ 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.

Généralisation

- étant donnés deux concepts h_1 et h_2 dans H_Ω, 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 $x \in X$.
- Par exemple, la figure suivante 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 $h_{\omega 1} \leq h_{\omega 2}$.

Apprentissage de termes monotones



Requêtes d’apprentissage

- 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 $h(x) = 1$.

Requêtes d’apprentissage (continue)

- 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.

Définition (Apprentissage exact).

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

Apprentissage exact (continue)

- Nous disons que la classe cible H^* est identifiable si la classe de représentation Ω utilisée par l'apprenant coïncide avec H^* c'est à dire $H^* = H_\Omega$.
- 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.

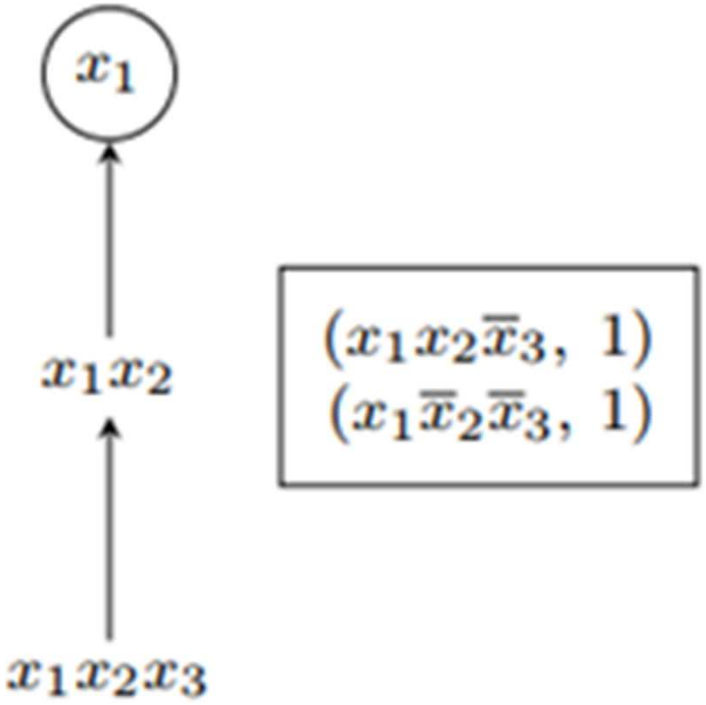
Apprentissage exact (continue)

- D'autre part, nous disons que 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 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 (Apprendre des termes).

- Soit $H_{MM} = \bigcup_{n \in N} H_{MM}^{n}$, où H_{MM}^{n} 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 $\in H_{MM}^{n}$ 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 suivante, où le concept cible est x_1 et l'apprenant reçoit deux contre-exemples.

Apprentissage de termes monotones



Requêtes d’équivalence et d’appartenance

- 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 et les fonctions linéaires à seuil, sont efficacement identifiables.

Exercice n° 4

a) Quelle est la différence entre une requête d'appartenance et une requête d'équivalence?

b) Si pour toute instance (x_1, x_2, x_3), nous avons $h_1((x_1,x_2,x_3)) = x_1$ et $h_2((x_1,x_2,x_3)) = x_1 \wedge x_2$, quelle est l'hypothèse (le concept) le plus spécifique,

c) Si $h(x_1,x_2,x_3) = x_1 \wedge x_2$ et la requête est $h((0,1,1)) = 0$ quel est le type de cette requête? et si $h^*((0,1,1)) = 1$, quelle serai la réponse à cette requête? Si le contre exemple est (0,1,0) mettre à jour h.

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 l'algorithme suivant, ce cadre peut-être vu comme un jeu répétitif 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.

Algorithme : Apprentissage enligne

- Paramètres : espace d'entrées X , espace de sorties Y, classe d'hypothèses H, fonction de perte l
- Initialisation : L'apprenant choisit une hypothèse initiale $h_1 \in H$
- Tours : pour chaque tour de jeu t = 1, 2, . . .
  1) l'environnement présente une instance $x_t \in X$
  2) l'apprenant effectue la prédiction $\hat{y}_t = h_t(x_t)$
  3) l'environnement révèle la réponse $y_t \in Y$ et l'apprenant subit $l(h_t, x_t, y_t)$
  4) l'apprenant choisit une nouvelle hypothèse h_{t+1}

Apprentissage enligne

- 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 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.

Apprentissage enligne (continue)

- De manière formelle, un algorithme d'apprentissage enligne peut-être identifié par une fonction A qui associe à toute séquence d'exemples $z = (z_1, \cdots, z_t)$ une hypothèse $h_t \in H$. Le regret à l'horizon m de A par rapport à une hypothèse $h^* \in H^*$ est défini par:

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

- Intuitivement, un algorithme d'apprentissage A atteint une performance non-triviale pour une classe cible H^* si, pour toute $h^* \in H^*$, 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.

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

Définition (Apprentissage enligne).

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

Remarque

- Comme pour l'apprentissage exact, H^* est identifiable enligne si $H^* = H_\Omega$.
- Nous disons que H^* est efficacement apprenable enligne si, durant chaque tour t, le temps calculé par A pour prédire (étape 2) et réviser son hypothèse (étape 4) est polynômial en n, d et $f(h^*)$.
- 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 H^* est particulièrement difficile à apprendre, il est utile de contraindre les séquences d'exemples.
- Nous disons que 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 (x_t, y_t) fournie par l'environnement est cohérente avec h^*.

Remarque

- 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 $h^* = (h^*_1, h^*_2, \ldots)$ une séquence infinie de fonctions dans H^* et h^*_m la sous-séquence obtenue à l'horizon m.
- Un changement dans 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 h^* est défini par

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

Définition (Tracking enligne).

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

Remarques

- La plupart des algorithmes enligne “apprennent très bien des choses simples”.
- En effet, lorsque l’espace cible 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.
- L’idée générale est de considérer chaque fonction cible $h \in H^*$ comme un expert et de prédire en se basant sur la performance des experts.

Exemple (Weighted Majority).

- L'algorithme d'apprentissage avec conseil d'experts le plus connu est "Weighted Majority", qui est généralisé en suite sous le nom de "Hedge".
- Considérons un espace H^* de fonctions cibles de X dans Y, appelées experts. Nous supposons que 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

$$\hat{y}_t = \frac{\sum_i^N w_{t,i} \; h *_i (x_t)}{\sum_j^N w_{t,j}}$$

Exemple (continue)

- L'algorithme calcule la moyenne pondérée des prédications de chaque expert.
- Dans le cas où 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 ∈ {1, · · · , N} est mis à jour par la règle

$$w_{t+1,i} = w_{t,i}\, e^{-\eta l(h*_i,\, x_t,\, y_t)}$$

- où $l(h*_i, x_t, y_t)$ 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 cette règle, l'algorithme de majorité pondérée apprend à rechercher le meilleur expert dans H∗ en maintenant des hypothèses dans l'enveloppe convexe de H∗.

Exemple (Perceptron)

- 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 $w \in R^n$.
- Initialement, le vecteur est $w_1 = 0$. Durant chaque tour t, le perceptron prédit avec la règle $\hat{y}_t = \text{sgn}\langle w_t, x_t \rangle$
- 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 $w_{t+1} = w_t + \eta y_t x_t$, où η est un paramètre d'apprentissage.

Exemple (Winnow)

- L'algorithme Winnow ressemble au Perceptron par sa simplicité, mais utilise une mise à jour multiplicative, plutôt qu'additive, des poids.
- Comme pour le Perceptron, Winnow maintient un vecteur de poids $w \in R^n$.
- Initialement, le vecteur est $w_1 = 1$. Durant chaque tour t, Winnow prédit avec la règle $\hat{y}_t = \text{sgn}\langle w_t, x_t \rangle$
- Si la prédiction est correcte alors $w_{t+1} = w_t$. Sinon l'algorithme utilise, pour chaque $i \in \{1, \cdot\cdot\cdot, n\}$, la mise à jour $w_{t+1,i} = w_{t,i}\, e^{\eta y_t x_{t,i}}$ où η est un paramètre d'apprentissage.
- Comme pour le Perceptron, la convergence de Winnow a fait l'objet de nombreuses investigations.

Exercice n° 5 (TP)

- Comme travail pratique, essayer de reprogrammer l'apprentissage de la porte logique XOR en utilisant l'algorithme d'apprentissage avec conseil d'experts "Weighted Majority" (la majorité pondérée). Utiliser trois experts:
  1. $h_1((0,0,1),(0,1,1),(1,0,1),(1,1,0))$
  2. $h_2((0,0,0),(0,1,1),(1,0,1),(1,1,0))$
  3. $h_3((0,0,0),(0,1,1),(1,0,1),(1,1,1))$
- Expliquer les résultats de l'exécution

Apprentissage statistique

- Le dernier modèle d'apprentissage supervisé présenté a été introduit en statistique, puis en informatique théorique. 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 D sur $X \times Y$, où X est un espace d'instances et Y un espace de décisions.
- Nous appelons échantillon une séquence $z = (z_1 , \cdots , z_m)$ d'exemples tirés selon la distribution jointe D^m.

Apprentissage statistique (continue)

- Etant donné une classe de représentation Ω, un algorithme d'apprentissage statistique pour Ω est une fonction A qui prend en entrée un échantillon z tiré aléatoirement selon D^m et retourne en sortie une hypothèse A(z) dans la classe H_Ω .
- En général, nous souhaiterions que A puisse produire, avec une forte probabilité sur 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 $risk_D(h)$; il est défini par la perte espérée de h sur un exemple z tiré aléatoirement selon D,
- Soit H une classe d'hypothèses cibles. Le but de l'apprenant A est de minimiser son regret probabiliste par rapport à toute hypothèse $h^* \in H^*$.
- En se basant sur ces notions, le modèle d'apprentissage "agnostique" fournit un cadre général à l'apprentissage statistique.

Modèle “agnostique”

- Dans le modèle “agnostique”, la distribution 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 D.
- En revanche, dans le modèle d’apprentissage probablement approximativement correct (PAC), nous supposons qu’il existe une dépendance fonctionnelle gouvernée par une fonction cible $h^* \in H^*$.
- Dans ce cadre réalisable, D est une distribution sur l’ensemble X; chaque exemple fourni par l’environnement est une paire $z = (x, h^*(x))$ où x est tiré aléatoirement selon D.

Apprentissage non-supervisé 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, le monitoring partiel et, bien entendu, l'apprentissage de processus de décision séquentielle.
- 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.
- Pour un ensemble E, nous notons 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 $M = (S, A, T, R, \gamma)$ où S est un espace d'états, A est un espace d'actions, $T : S \times A \rightarrow P_S$ est la fonction de transition, $R : S \times A \rightarrow P_{[0,1]}$ est la fonction de récompense (bornée), et γ est le facteur de dévaluation compris dans l'intervalle [0, 1].
- Un processus de décision Markovien est fini si les espaces S et A sont tous deux finis.

Apprentissage par renforcement

- 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 associée à l'état $s \in S$ et l'action $a \in A$. 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 $\pi : S \rightarrow A$.
- 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 ;
- 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, Notons qu'une politique ne peut pas avoir une valeur au delà de $1/(1 - \gamma)$ puisque la récompense maximale vaut 1.

Algorithme : Apprentissage par Renforcement

- Paramètres : espace d'états S, espace d'actions A, fonction (cachée) de transition T , fonction (cachée) de récompense R, facteur de dévaluation γ
- Initialisation : L'environnement occupe un état $s_1 \in S$ et le communique à l'agent
- Tours : pour chaque tour de jeu t = 1, 2, . . .
  1) l'agent perçoit l'état s_t et choisit une action $a_t \in A$
  2) l'environnement retourne à l'agent la récompense r_t choisie aléatoirement selon $R(s_t , a_t)$ et occupe un nouvel état $s_{t+1} \in S$ choisi aléatoirement selon $T(\cdot \mid s_t, a_t)$

Remarques

- 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.
- 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 l'algorithme, 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 récompense et de transition. Un chemin est une séquence de la forme $c_t = (s_1 , a_1 , r_1 , s_2 , \cdot \cdot \cdot , s_t)$ 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 $A_t : \{S \times A \times [0, 1]\}^* \times S \rightarrow A$. Les fonctions de valeur sont étendues de manière naturelle aux politiques non stationnaires.

Exemple (R-MAX)

- 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 = (S, A, T, R, γ), et utilisent leur modèle approximatif $\hat{M} = (S, A, \hat{T}, \hat{R}, \gamma)$ 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], · · · , 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 :

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

Exercice n° 6

Expliquer la différence entre le modèle agnostique et le modèle PAC

Référence

Modèles d'Apprentissage Artificiel, 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