Université de Paris

M1 – Introduction à l'Intelligence Artificielle et la Théorie des Jeux

Examen – 07 Janvier 2020

Durée : 2 heures.
Notes de cours et documents manuscrits autorisés. Toutes les réponses doivent être justifiées.

Exercice 1

Donner pour chaque cas de figure ci-dessous un arbre pour lequel :

1. la recherche A^* est plus efficace que la recherche en largeur d'abord **et** la recherche en profondeur d'abord.
2. la recherche en largeur d'abord **et** la recherche en profondeur d'abord sont plus efficaces que la recherche A^*.
3. la recherche A^* est plus efficace que la recherche gloutonne **et** la recherche à coût uniforme.
4. la recherche gloutonne est plus efficace que la recherche à coût uniforme **et** la recherche A^*.
5. la recherche à coût uniforme est plus efficace que la recherche gloutonne **et** la recherche A^*.

L'efficacité est mesurée en nombre de nœuds développés. Pour chaque cas, donner toute l'information nécessaire : le coût d'un arc, la valeur de l'heuristique pour chaque nœud, le nœud final. Pour chaque cas, l'heuristique utilisée doit être admissible. Il faut choisir les arbres les plus simples possible (le moins de nœuds possible, et le plus faible degré possible).

Exercice 2

Question 1 : Donner un perceptron à trois entrées $x_0 = 1$, x_1, et x_2, qui calcule la fonction booléenne : $B_1(x_1, x_2) = (\neg x_1 \vee x_2)$. (On ne demande pas d'appliquer un algorithme d'apprentissage pour trouver le perceptron. La réponse doit bien entendu être justifiée).

Question 2 : Donner un réseau de neurones à deux couches, à trois entrées $x_0 = 1$, x_1, et x_2, qui calcule la fonction booléenne de l'équivalence : $B_2(x_1, x_2) = ((x_1 \wedge x_2) \vee (\neg x_1 \wedge \neg x_2))$. (Ici non plus, on ne demande pas d'appliquer un algorithme d'apprentissage). Expliquer pourquoi un seul perceptron n'est pas suffisant pour calculer cette fonction.

Exercice 3

Question 1 : L'ensemble d'apprentissage suivant est linéairement séparable :

	1	2	3	4	5	classe
$\vec{x}_1$:	1	1	0	0	1	0
$\vec{x}_2$:	0	0	1	1	1	1
$\vec{x}_3$:	0	0	1	0	1	1
$\vec{x}_4$:	0	1	1	1	1	0
$\vec{x}_5$:	1	0	0	0	1	0
$\vec{x}_6$:	0	0	1	1	0	1

Entraîner un perceptron avec cet ensemble et la procédure de correction d'erreur. Le vecteur de poids du perceptron est un vecteur à six dimensions. Commencer avec $\vec{w} = (0, 0, 0, 0, 0, 0)$. Faire au plus 18 itérations. Il est conseillé de présenter les vecteurs un par un dans l'ordre.

Question 2 : Donner un exemple d'un ensemble d'apprentissage de vecteurs à 3 dimensions dans $\{0,1\}^3$ qui n'est pas linéairement séparable. Est-il possible de décrire cet ensemble par une formule booléenne ?

Exercice 4

On considère un espace de description comprenant trois attributs : forme (rond ou carré), taille (petit ou grand), et couleur (bleu, blanc, ou rouge). L'attribut cible (à déterminer lors du classement) est binaire (il y a deux classes oui et non). On dispose des informations suivantes (le "?" correspond à une valeur manquante) :

T1 T2 T3

forme	taille	couleur	classe
rond	petit	bleu	oui
carré	grand	rouge	non
rond	g ?	blanc	oui
carré	petit	bleu	oui
rond	grand	bleu	oui
carré	grand	blanc	non
carré	g ?	blanc	oui
carré	grand	bleu	non
carré	petit	rouge	oui
rond	grand	blanc	oui

On remplace les valeurs manquantes par la valeur majoritaire prise par cet attribut sur l'échantillon complet. Quelle valeur associe-t-on sur l'échantillon ci-dessus ? Peut-on trouver un arbre de décision parfait ? Appliquer l'algorithme de construction de l'arbre de décision en utilisant la fonction Gini (*) pour le calcul du gain. On décide qu'un nœud est terminal, i.e., de considérer un nœud comme une feuille, lorsqu'il y a au plus un exemple mal classé associé à ce nœud. Détailler les calculs pour le test à choisir en racine de l'arbre.

(*) On rappelle que la fonction Gini à une position p de l'arbre de décision est définie par :

$$Gini(p) = 1 - \sum_{k=1}^{c} P(k/p)^2$$

où $C = \{1, \ldots, c\}$ est l'ensemble des classes, et $P(k/p)$ est la proportion des éléments de l'échantillon qui sont de la classe k à la position p.

On rappelle aussi que le gain pour un test t à une position p est donné par :

$$Gain(p,t) = Gini(p) - \sum_{j=1}^{n} P_j \cdot Gini(pj)$$

où n est le nombre de valeurs possibles pour le test t, et P_j est la proportion des éléments de l'échantillon qui passent de la position p à la position pj, c'est-à-dire qui satisfont la j^{ieme} branche du test t.