INF4230 – Intelligence artificielle
Hiver 2016 – Examen final / Partie A

Éric Beaudry
Département d'informatique
Université du Québec à Montréal

Mardi 26 avril 2016 – 9h30 à 12h30 (3 heures) – Local SB-M230

Instructions

1. Aucune documentation permise.
2. Une calculatrice de base (non programmable) et sans dispositif sans-fil est permise. Notez que les calculs sont tout de même faisables mentalement sans calculatrice.
3. Tous les autres appareils électroniques, sont strictement interdits.
4. Répondez directement sur le questionnaire à l'intérieur des endroits appropriés.
5. Aucune question ne sera répondue durant l'examen. Si vous croyez qu'une erreur ou qu'une ambiguïté s'est glissée dans le questionnaire, indiquez clairement la supposition que vous avez retenue pour répondre à la question.
6. L'examen est composé de deux parties : la présente partie A contient des questions à développement et la partie B des questions à choix multiples.
7. Ne détachez pas les feuilles du questionnaire, à l'exception de la dernière feuille (annexes A et B).
8. Le côté verso des annexes peut être utilisé comme brouillon. Des feuilles additionnelles peuvent être demandées au surveillant.
9. Vous devez présenter votre carte étudiante lors de la remise de votre copie.

Solutionnaire

Résultat de la Partie A

Q1 (/5)	Q2 (/2)	Q3 (/4)	Q4 (/5)	Q5 (/4)	TOTAL (/20)

1 Raisonnement logique [5 points]

(a) Traduisez les énoncés suivants en logique du premier ordre. Numérotez vos formules. [2 points]

1. Les fruits sont des aliments périssables.
2. Une pomme est un fruit.
3. L'objet *pom*1 est une pomme.
4. Les aliments périssables doivent être mangés avant d'être expirés ou jettés.
5. *pom*1 est expirée.

1. $\forall x : fruit(x) \rightarrow perissable(x)$
2. $pomme(x) \rightarrow fruit(x)$
3. $pomme(pom1)$
4. $\forall x : perissable(x) \rightarrow (mange(x) \wedge \neg expire(x)) \vee jette(x)$
5. $expire(pom1)$

(b) Convertissez les formules obtenues en (a) sous forme clausale. Rappel : une clause est une disjonction («ou» logique) de littéraux (littéral = affirmation ou négation d'un prédicat). Numérotez les clauses. [1 point]

1. $\neg fruit(x1) \vee perissable(x1)$
2. $\neg pomme(x2) \vee fruit(x2)$
3. $pomme(pom1)$
4. $\neg perissable(x3) \vee mange(x3) \vee jette(x3)$
5. $\neg perissable(x4) \vee \neg expire(x4) \vee jette(x4)$
6. $expire(pom1)$

(c) Prouvez l'affirmation «*pom*1 doit être jetté». Commencez par écrire la négation de l'énoncé à prouver. Numérotez cette nouvelle clause en continuant la numérotation utilisée en (b). Ensuite, utilisez la technique de résolution. À chaque étape, indiquez les numéros des deux clauses utilisées et l'unificateur (substitution de variables) requis. Rappel : le but est d'obtenir une clause vide (une contradiction). [2 points]

7. $\neg jette(pom1)$
   ----
8. $fruit(pom1)$ // Clauses 2 et 3 avec l'unificateur le plus général (UPG) $\{x_2 \mapsto pom1\}$
9. $perissable(pom1)$ // Clauses 1 et 8 avec l'UPG $\{x_1 \mapsto pom1\}$
10. $\neg expire(pom1) \vee jette(pom1))$ // Clauses 5 et 9 avec l'UPG $\{x_4 \mapsto pom1\}$
11. $jette(pom1))$ // Clauses 6 et 10 avec l'UPG $\{\}$
12. $false$ // Clauses 7 et 11 avec l'UPG $\{\}$

Nous avons montré une contradiction. Donc, *pom*1 doit être jetté.

2 Raisonnement probabiliste [2 points]

(a) Expliquez en vos propres mots ce qu'est le raisonnement probabiliste en IA. [1 point]

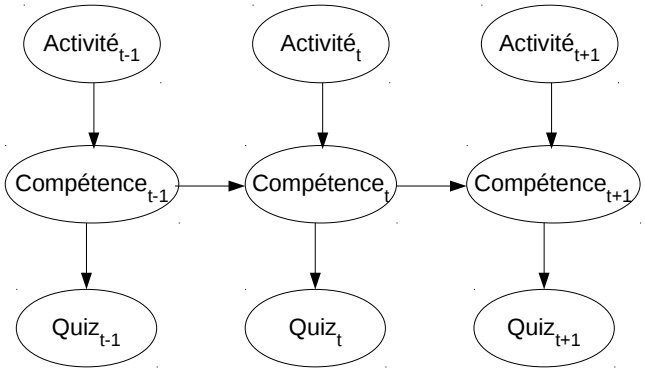
En IA, le raisonnement probabiliste consiste à prendre des décisions sous incertitude basé sur des calculs faisant appel à des probabilités. Typiquement, un agent intellgient évalue la probabilités des conséquences possibles de ses options (actions). En combinant la théorie des probabilités et de l'utilité, un agent intelligent choisi les actions qui maximisent son utilité (niveau de désirabilité ou préférence).

(b) La règle de Bayes : P(A|B)=P(B|A)P(A)/P(B). À quoi sert la règle de Bayes ? [1 point]

En pratique, il est plus généralement facile d'obtenir les probabilités conditionnelles de causes à effets (sens causale). Par exemple, on peut estimer la probabilité d'un sympôme sachant une maladie. C'est ainsi que dans un réseau bayésien, on a généralement des tables de probabilités de type P(effets|causes). La règle de Bayes est utile si on cherche la probabilité d'une ou de causes étant donné les effets observés (sens diagnostique).

3 Réseaux bayésiens dynamiques [4 points]

Un système tutoriel intelligent (STI) est un logiciel qui assiste l'apprenant (l'utilisateur) dans un domaine d'apprentissage donné. Dans un STI, un réseau bayésien dynamique (RBD) peut être utilisé pour estimer et suivre le niveau de compétence de l'apprenant. Voici un réseau bayésien simple pour un petit logiciel STI fictif.



C_{t-1}	A_t	$P(C_t \mid A_t, C_{t-1})$		
		C_t=D	C_t=M	C_t=E
D	X	0.9	0.1	0.0
D	P	0.3	0.6	0.1
D	L	0.7	0.2	0.1
M	X	0.2	0.8	0.0
M	P	0.1	0.7	0.2
M	L	0.1	0.4	0.5
E	X	0.1	0.2	0.7
E	P	0.1	0.1	0.8
E	L	0.0	0.1	0.9

À chaque époque t, le STI propose une activité (A) à l'apprenant. Il y a 3 activités possibles : aucune activité (X), une présentation de type PowerPoint (P) et la lecture d'un texte approfondi (L). Les activités visent à augmenter le niveau de compétence de l'apprenant. Une activité peut parfois mêler l'apprenant, donc réduire sa compétence.
Le niveau de compétence (C) de l'apprenant a 3 valeurs possibles : débutant (D), moyen (M) et expert (E). Le niveau de connaissance de l'apprenant ne peut être mesurée directement. Toutefois, le STI pose une question (Quiz) après chaque activité. Un apprenant débutant, moyen et expert ont respectivement une probabilité de 0.1, 0.5 et 0.9 de répondre correctement à une question quiz.

(a) Une opération typique dans les RBD est le fitrage. Dans le contexte du STI décrit ci-haut, quelles sont les variables ... ?

cachées : aucune (C pourrait être acceptée) d'évidence : A, Q d'interrogation : C

(b) Supposons qu'un apprenant soit initialement débutant. On lui fait suivre deux activités X et P (dans l'ordre). On ignore s'il a répondu correctement aux 2 questions. Quelle est la probabilité qu'il soit finalement rendu expert ? [1 point]

$P(C_0) = \langle 1.0, 0.0, 0.0 \rangle$
$P(C_1|A_1 = X) = \langle 0.9, 0.1, 0.0 \rangle$ $P(C_2|A_1 = X, A_2 = P) = 0.9\langle 0.3, 0.6, 0.1 \rangle + 0.1\langle 0.1, 0.7, 0.2 \rangle = \langle 0.28, 0.61, 0.11 \rangle$
$P(C_2 = E|A_1 = X, A_2 = P) = 0.11$

(c) Supposons qu'un apprenant soit initialement moyen. On lui fait suivre deux activités P et L (dans l'ordre). On ignore s'il a répondu correctement aux 2 questions. Quelle est la probabilité qu'il soit finalement rendu expert ? [1 point]

$P(C_0) = \langle 0.0, 1.0, 0.0 \rangle$
$P(C_1|A_1 = P) = \langle 0.1, 0.7, 0.2 \rangle$ $P(C_2|A_1 = P, A_2 = L) = 0.1\langle 0.7, 0.2, 0.1 \rangle + 0.7\langle 0.1, 0.4, 0.5 \rangle + 0.2\langle 0.0, 0.1, 0.9 \rangle$
$= \langle 0.14, 0.32, 0.54 \rangle$
$P(C_2 = E|A_1 = X, A_2 = P) = 0.54$

(d) Supposons qu'un apprenant soit initialement débutant. On lui fait suivre deux activités P et L (dans l'ordre). Cette fois-ci, on connaît qu'il a répondu incorrectement aux deux questions de quiz. Quelle est la probabilité qu'il soit finalement rendu expert ? [1 point]

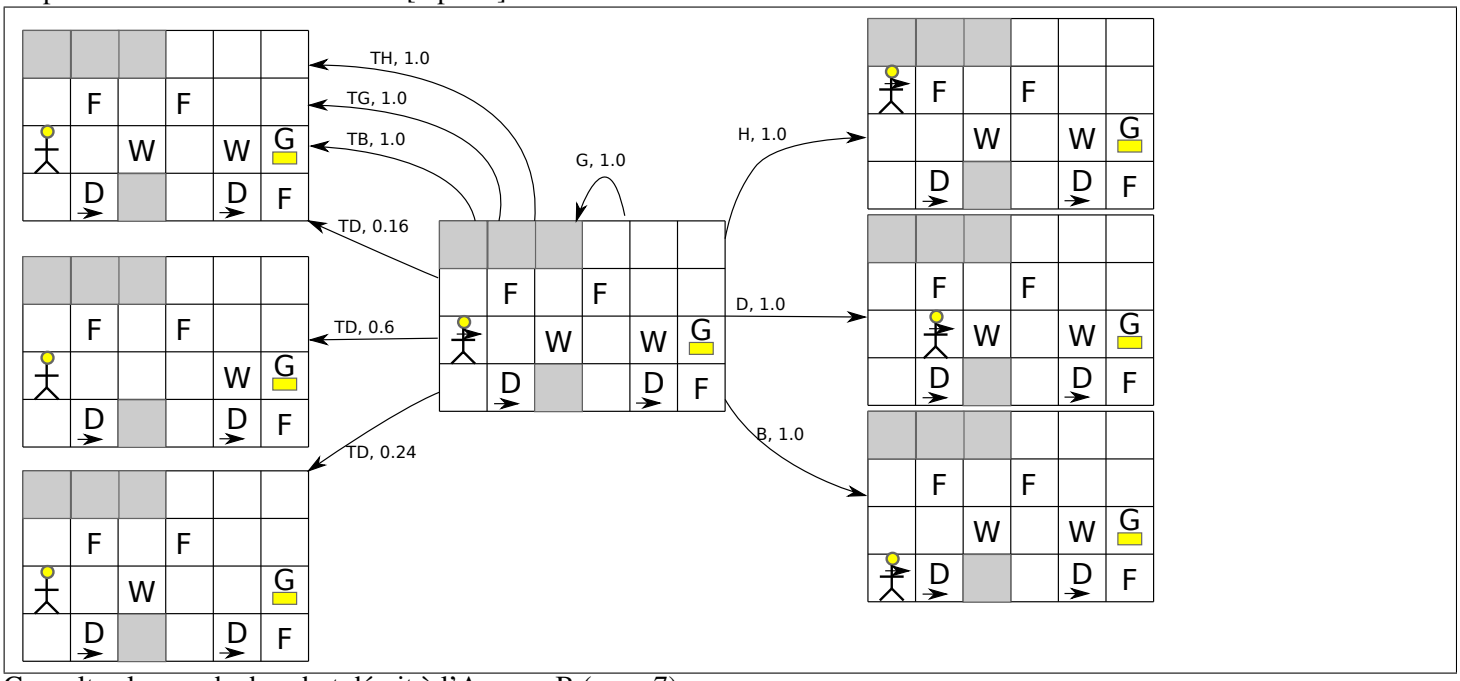
$P(C_0) = \langle 1.0, 0.0, 0.0 \rangle$
$P(C_1|A_1 = P) = \langle 0.3, 0.6, 0.1 \rangle$
$P(C_1|A_1 = P, \neg q_1) = \alpha\langle 0.3 \cdot 0.9, 0.6 \cdot 0.5, 0.1 \cdot 0.1 \rangle = \alpha\langle 0.27, 0.30, 0.01 \rangle = \langle 0.466, 0.517, 0.017 \rangle$
$P(C_2|A_2 = L) = 0.466\langle 0.7, 0.2, 0.1 \rangle + 0.517\langle 0.1, 0.4, 0.5 \rangle + 0.017\langle 0.0, 0.1, 0.9 \rangle = \langle 0.378, 0.302, 0.321 \rangle$
$P(C_2|A_2 = L, \neg q_2) = \alpha\langle 0.378 \cdot 0.9, 0.302 \cdot 0.5, 0.321 \cdot 0.1 \rangle = \alpha\langle 0.340, 0.151, 0.032 \rangle = \langle 0.650, 0.289, 0.061 \rangle$
Il y a 6.1% de chance d'être rendu expert.

4 Processus décisionnels de Markov (MDP) [5 points]

Consultez le monde du Wumpus modifié et décrit à l'Annexe A (page 7).
(a) Dans l'annexe, on dit qu'il y a une récompense dans la ou les cases contenant de l'or. À votre avis, serait-il requis d'ajouter des pénalités (récompenses négatives) dans les cases qui contiennent des puits ? Justifiez votre réponse. [1 point]

Non, l'ajout de pénalités n'est pas **requis**. Ces états seront natuellement évités, car ils ne permettent pas d'accumuler des récompenses présentes et futures.

(b) Dessinez l'état initial et les états successeurs (il y en entre 5 et 9). Il est important d'indiquer clairement les actions et les probabilités sur les transitions. [1 point]



Consultez le monde du robot décrit à l'Annexe B (page 7).
(c) Quelle est la taille de l'espace d'états accessibles (résultants de 0, 1 ou plusieurs actions) ? [1 point]

$8 \times 4 = 32$ états

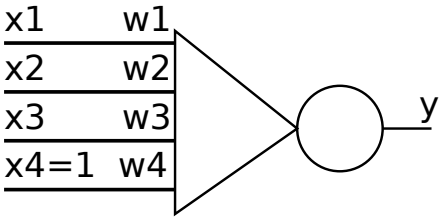
(d) Simulez deux itérations de l'algorithme d'itération par valeurs sur les états (c_1, E) et (c_9, N). Les valeurs de tous les états sont initialisées à 2. Un facteur d'atténuation (*discount factor*) de 0.8 est utilisé. Les cellules c_2, c_6 et c_9 contiennent des récompenses de -50, $+10$ et $+20$. [2 points]

Itération 1 :
$V[(c_1,E)] = 0.0 + 0.8 \times \max(1.0 \cdot 2; 1.0 \cdot 2; 0.05 \cdot 2 + 0.85 \cdot 2 + 0.1 \cdot 2) = 1.6$
$V[(c_9,N)] = 20 + 0.8 \times \max(1.0 \cdot 2; 1.0 \cdot 2; 0.05 \cdot 2 + 0.85 \cdot 2 + 0.1 \cdot 2) = 21.6$

Itération 2 :
$V[(c_1,E)] = 0.0 + 0.8 \times \max(1.0 \cdot 1.6; 1.0 \cdot 1.6; 0.05 \cdot 1.6 + 0.85 \cdot V[(c_2,E)] + 0.1 \cdot V[(c_2,E)]) = 1.28$
À noter qu'il n'est pas requis de calculer la valeur exacte de $V[(c_2,E)]$, car il est c'est assez évident que $V[(c_2,E)] < 1.6$.
$V[(c_9,N)] = 20 + 0.8 \times \max(1.0 \cdot 21.6; 1.0 \cdot 21.6; 0.05 \cdot 21.6 + 0.85 \cdot V[(c_6,N)] + 0.1 \cdot V[(c_6,N)]) = 37.28$
À noter qu'il n'est pas requis de calculer $V[(c_6,E)]$, car il est c'est assez évident que $V[(c_6,E)] < 21.6$.

5 Réseaux de neurones artificiels [4 points]

Soit le réseau de neurones suivant et les données d'entraînement suivantes.



	Échantillons			
x_1	1	1	1	1
x_2	2	4	2	8
x_3	5	4	2	1
y	+1	+1	-1	-1

Fonction d'activation : $y = sign(\sum_{i=1}^{4}(w_i \cdot x_i))$

(a) Simulez une itération complète de l'algorithme d'apprentissage du perceptron en utilisant les données d'entraînement ci-haut. Le taux d'apprentissage est fixé à 0.3 et les poids initiaux sont : $w_1 = 0$, $w_2 = 0$, $w_3 = 0$ et $w_4 = 0$. [2 points]

Ici, on suppose que $sign(0) = 0$. Les supposions comme $sign(0) = +1$ ou $sign(0) = -1$ sont également acceptées.

Itération #1 / Échantillon #1 $(1,2,5) \rightarrow +1$:
- Sortie : $sign(0 \cdot 1 + 0 \cdot 0 + 0 \cdot 4 + 0 \cdot 1) = 0$
- Erreur : $(+1) - (0) = +1$
- Correction : $\Delta w = 0.3 \cdot +1 \cdot (1,2,5,1) = (0.3, 0.6, 1.5, 0.3)$
- Poids : $(0.3, 0.6, 1.5, 0.3)$

Itération #1 / Échantillon #2 $(1,4,4) \rightarrow +1$:
- Sortie : $sign(0.3 \cdot 1 + 0.6 \cdot 4 + 1.5 \cdot 4 + 0.3 \cdot 1) = 1$
- Erreur : $(+1) - (+1) = 0$
- Correction : $\Delta w = 0.3 \cdot 0 \cdot (1,4,4,1) = (0,0,0,0)$
- Poids : $(0.3, 0.6, 1.5, 0.3)$

Itération #1 / Échantillon #3 $(1,2,2) \rightarrow -1$:
- Sortie : $sign(0.3 \cdot 1 + 0.6 \cdot 1.5 + 0.3 \cdot 2 + 0.3 \cdot 1) = 1$
- Erreur : $(-1) - (+1) = -2$
- Correction : $\Delta w = 0.3 \cdot -2 \cdot (1,2,2,1) = (-0.6, -1.2, -1.2, -0.6)$
- Poids : $(-0.3, -0.6, 0.3, -0.3)$

Itération #1 / Échantillon #4 $(1,8,1) \rightarrow -1$:
- Sortie : $sign(-0.3 \cdot 1 + -0.6 \cdot 8 + 0.3 \cdot 1 + -0.3 \cdot 1) = -1$
- Erreur : $(-1) - (-1) = 0$
- Correction : $\Delta w = 0.2 \cdot 0 \cdot (1,0,2,1) = (0,0,0,0)$
- Poids : $(-0.3, -0.6, 0.3, -0.3)$

(b) Le neurone artificiel ci-haut sera-t-il capable d'apprendre correctement le jeu de données d'entraînement fourni ? Justifiez votre réponse. [1 point]

Oui. Les données sont linéairement séparables.
[Mettre un graphique ici]

(c) Un réseau de neurones artificiels pourrait-il être **utile** pour résoudre le TP2 (*Connect*5) ? Pourquoi ? [1 points]

Oui.
La technique traditionnelle (ou naturelle) pour décider d'une action dans les jeux comme Connect5 est l'algorithme minimax (avec ou sans élagage alpha-beta). Pour beaucoup de jeux, dont Connect5, l'arbre de recherche du jeu a une taille beaucoup trop grande pour être exploré de façon exhaustive. C'est ainsi qu'il faut prendre des décision imparfaite en limitant la profondeur de recherche dans l'arbre et en utilisant une fonction d'évaluation pour calculer une approximation de la valeur minimax des noeuds intermédiaires dans l'arbre. Écrire une bonne fonction d'évaluation est un défi.
Un réseau de neurones artificiel peut être **utile** pour implémenter une telle fonction d'évaluation. L'algorithme AlphaGo de Google utilise d'ailleurs cette stratégie pour jouer au jeu de Go. La même technique pourrait être utile pour Connect5.

Annexe A pour la Question 4

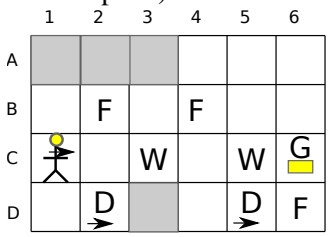
Version modifiée du monde des Wumpus.

- Il peut y avoir plusieurs monstres.
- Le joueur peut garder avec lui qu'une seule flèche. Le joueur peut se ravitailler, d'une flèche à la fois, en allant sur un dépôt de flèches (D). Les dépôts ont un nombre illimité de flèches.
- Le monde est totalement observable. Le joueur connaît la position de lui-même, de l'or (G), des dépôts de flèches (D), des puits/fosses (F) et des montres Wumpus (W).
- Les cases grisées sont des obstacles.
- Le monde est non déterministe. **Les probabilités sont connues.**
- Le lancement d'une flèche a un résultat incertain. Quand la flèche traverse une case occupée par un Wumpus, ce dernier peut être touché (mourir) **avec une probabilité de 0.6** ou l'éviter **avec une probabilité de 0.4**. Si le Wumpus est touché, la flèche arrête. Sinon, la flèche suit son parcours sur les cases suivantes, et ainsi elle peut tuer un autre Wumpus. Une flèche peut tuer au plus un seul Wumpus.
- Lorsque le joueur avance sur un puits/fosse (F), il essaie de sauter par dessus. Il y a 2 résultats possibles : (1) il peut réussir le saut et atterrir sur la case suivante avec une probabilité de 0.9 ; (2) il peut échouer le saut, tomber et mourir avec une probabilité de 0.1.
- **Le but de l'agent est d'accumuler des récompenses. Le joueur accumule 10 unités de récompenses lorsqu'il est dans une case qui contient de l'or**.

Le joueur peut exécuter 8 actions :

- 4 actions de déplacement : H, B, G et D pour haut, bas, gauche et droite ;
- 4 actions pour tirer dans les 4 directions : TH, TB, TG et TD.

Il n'y a pas d'action spécifique pour prendre une flèche dans le dépôt de flèches (cela se fait automatiquement) ;
Le jeu se termine dès que le joueur atteint la case contenant l'or, ou quand il meurt (en touchant un Wumpus ou en tombant dans un puits). Considérez l'état initial suivant (le joueur est à la positon C1 et a une flèche en main).



Annexe B pour la Question 4

Un robot doit naviguer dans un environnement décomposé en une grille de neuf cellules étiquetées de c_1 à c_9. La cellule c_3 est occupée par un obstacle et ne peut-être atteinte. Un état du robot est représenté par une paire (p, o) où p et o représentent respectivement sa position (cellule) et son orientation (Nord, Sud, W pour ouest, Est). La figure ci-droite montre un robot dans l'état (c_1, E).

Le robot peut exécuter trois actions :

- a_1 effectue une rotation de 90° dans le sens horaire ;
- a_2 effectue une rotation de 90° dans le sens anti-horaire ;
- a_3 avance d'une case.

Les actions de rotation sont déterministes. Par contre, l'action d'avancer d'une case est incertaine. Suite à l'exécution de l'action a_3, le robot peut ne pas avoir bougé (P=5%), peut avoir avancé d'une seule case (P=85%), ou peut avoir avancé de deux cases (P=10%).