IFT 615 : Devoir 4
Travail individuel

Remise : 12 avril 2012, 16h20 (**au plus tard**)

1. **[2 points]** Dans le cours, nous avons vu différents types de problèmes d'intelligence artificielle ainsi que plusieurs solutions possibles pour ceux-ci :

Type de problème	Solutions
Recherche du chemin le plus court	recherche heuristique A*
Optimisation d'une fonction générique	recherche locale : *hill climbing*, *simulated annealing*, algorithmes génétiques
Recherche pour jeux à deux adversaires	minimax, élagage alpha-beta
Problème de satisfaction de contraintes	*backtracking search* (avec/sans inférence, ordonnancement des variables, des valeurs), recherche locale *min-conflicts*
Inférence logique	méthode de résolution pour la logique du premier ordre
Raisonnement probabiliste	réseaux bayésiens (dynamiques)
Dérivation du plan optimal dans un environnement stochastique connu	*value iteration*, *policy iteration*
Apprentissage automatique supervisé	classifieur des k plus proches voisins, Perceptron, régression logistique, réseaux de neurones
Apprentissage par renforcement	méthode directe, programmation dynamique adaptative, différence temporelle, *Q-learning*, recherche de plan/politique

Soit les situations (a) à (e) suivantes, (1) identifiez le type de problème décrivant le mieux la situation et (2) justifiez votre choix en choisissant une solution possible pour ce type de problème et en décrivant brièvement comment vous l'appliqueriez à cette situation.

Voici un exemple de réponse à laquelle je m'attends.

Situation : Une compagnie de transport en commun souhaite développer un outil pour ses usagers, que ceux-ci pourraient consulter afin d'obtenir des suggestions d'itinéaires à emprunter dans leurs déplacements.

Réponse : Cette situation correspond à une recherche du chemin le plus court. L'algorithme de recherche heuristique A* pourrait donc être utilisé. Chaque noeud dans le graphe de recherche serait une position possible dans le réseau de transport. Le noeud initial serait l'endroit de départ et le noeud but serait la destination que l'usager souhaite atteindre. Les noeuds successeurs d'un noeud donné serait l'ensemble des positions adjacentes dans le réseau (un noeud sucesseur par ligne de métro ou d'autobus couvrant la position donnée). Le coût entre deux noeuds serait le temps nécessaire pour passer entre deux positions à l'aide du mode de transport associé. Une heuristique similaire à la distance à vol d'oiseau entre les positions pourrait être utilisée par A*.

Situations à résoudre :

(a) Une compagnie de réparation d'ordinateurs souhaite mettre sur pied un outil d'aide au diagnostique pour ses techniciens qui doivent réparer les ordinateurs de ses clients. La compagnie a accès à une base de données de faits concernant le fonctionnement d'un ordinateur (ex. : "si un pilote d'imprimante n'est pas installé, l'imprimante ne fonctionnera pas"). Ces faits sont déterministes.

(b) On souhaite développer un jeu pour téléphone Android. C'est un jeu qui se joue à deux, ainsi on souhaite offrir la possiblité au joueur de jouer contre une intelligence artificielle. Il n'y a pas d'aspect aléatoire dans les règles du jeu : tout est déterministe. Le jeu est très simple et court à jouer, puis l'ensemble des règles du jeu sont faciles à décrire.

(c) Une compagnie d'assurance automobile souhaite implémenter un outil permettant d'évaluer le risque qu'un nouveau client ait un accident et fasse une réclamation. La compagnie a accès à plusieurs statistiques sur d'anciens clients ayant ou n'ayant pas eu d'accidents. Pour chacun de ces anciens clients, on a de l'information sur leur profil : leur âge, la marque de leur voiture, s'ils habitent en ville ou en campagne, la distance quotidienne parcourue en voiture, etc. L'information est partielle, c.-à-d. que certaines des informations du profil ne sont pas connues. Étant données de l'information (aussi partielle) sur un nouveau client, la compagnie aimerait donc avoir un outil pour déterminer le risque que ce client fasse éventuellement une réclamation pour un accident.

(d) Une société de transport en commun a besoin d'un programme qui lui permettrait de gérer son parc d'autobus. Elle aimerait mettre en place plusieurs lignes d'autobus dans la ville, avec un service fréquent. Étant donné un horaire pour toutes ces lignes, la société de transport aimerait que ce programme puisse lui dire si elle a suffisamment d'autobus pour assurer ce service.

(e) L'aéroport de Montréal aimerait développer un outil de détection de matériel interdit dans les valises de ses voyageurs. Elle a à sa disposition une base de données d'images de baggages obtenues à l'aide de radiographie à rayon X. Pour chacune de ces images, elle a également la décision d'un de ses employés, déterminant si le baggage contient un objet dangeureux ou pas.

2. **[2 points]** Soit un MPD avec $S = \{s_0, s_1, s_2, s_3\}$ où s_2 est terminal, l'ensemble d'actions $\{a_1, a_2, a_3\}$ et le facteur d'escompte $\gamma = 0.5$. On suppose que toutes les actions sont possibles à partir de chaque état. Également, soit les essais suivants :

$(s_0)_1 \xrightarrow{a_1} (s_1)_1 \xrightarrow{a_2} (s_1)_1 \xrightarrow{a_2} (s_2)_{10}$
$(s_0)_1 \xrightarrow{a_3} (s_0)_1 \xrightarrow{a_3} (s_1)_1 \xrightarrow{a_1} (s_1)_1 \xrightarrow{a_1} (s_0)_1 \xrightarrow{a_3} (s_2)_{10}$
$(s_0)_1 \xrightarrow{a_2} (s_3)_2 \xrightarrow{a_1} (s_2)_{10}$
$(s_0)_1 \xrightarrow{a_1} (s_0)_1 \xrightarrow{a_1} (s_3)_2 \xrightarrow{a_1} (s_1)_1 \xrightarrow{a_2} (s_2)_{10}$

Supposez que ces essais aient été générés par un agent faisant de l'apprentissage par renforcement à l'aide du *Q-learning*, avec un taux d'apprentissage $\alpha = 1$.

(a) Donnez la liste des mises à jour de la fonction action-valeur. Supposez une initialisation de $Q(s, a)$ à 0.

(b) Quelle serait la politique apprise à la fin ? Donnez l'action choisie par cette politique pour chaque état, excepté l'état terminal.

3. **[2 points]** Soit les deux descriptions de cours suivantes, une du programme d'informatique et l'autre du programme d'études littéraires et culturelles, à l'Université de Sherbrooke :

Informatique	Études littéraires et culturelles
Formaliser les structures de données, comparer et choisir les meilleures mises en oeuvre des structures en fonction du problème à traiter.	Étudier, d'un point de vue épistémologique, les notions de littérature et de culture. Étudier les rapports qu'elles entretiennent entre elles.

On a donc deux corpus, un pour la catégorie informatique, l'autre pour les études littéraires et culturelles. Soit la nouvelle description de cours suivante :

> Étudier les notions de base en théorie des graphes. Étudier les structures de données externes.

Supposez l'utilisation du vocabulaire suivant : {"les", "structures", "de", "données", "graphes", "Étudier", "en", "problème", "culture", "littérature", "notions", "."}.

(a) Quelle est la distribution unigramme associée à la catégorie informatique ? Utiliser une constante de lissage $\delta = 0.1$.

(b) Quelle est la distribution unigramme associée à la catégorie études littéraires et culturelles ? Utiliser aussi une constante de lissage $\delta = 0.1$.

(c) À l'aide des distributions unigrammes calculées en (a) et (b), et en supposant une distribution a priori uniforme sur les catégories (c.-à-d. $P(C = \text{informatique}) = P(C = \text{études littéraires et culturelles}) = 0.5$.

Déterminez dans quelle catégorie la nouvelle description serait classifiée, par un classifieur bayésien naïf multinomial.

4. **[4 points]** Implémentez une classe Python `ReseauDeNeurones` correspondant à un réseau de neurones à une couche cachée, entraîné à l'aide de l'algorithme de rétropropagation des gradients, pour un problème à 2 classes ($y = 0$ et $y = 1$). Pour ce faire, complétez l'implémentation des méthodes suivantes :

```
class ReseauDeNeurones:

    def __init__(self, alpha, T):
        #Mettre code ici
        pass

    def initialisation(self, W, w):
        # Mettre code ici
        pass

    def parametres(self):
        # Mettre code ici
        pass

    def prediction(self, x):
        # Mettre code ici
        pass

    def mise_a_jour(self, x, y):
```

```
        # Mettre code ici
        pass

    def entrainement(self, X, Y):
        # Mettre code ici
        pass
```

où :

- `__init__(self, alpha, T)` est le constructeur et a comme arguments le taux d'apprentissage `alpha` et le nombre d'itérations `T` à utiliser pour l'entraînement.
- `initialisation(self, W, w)` initialise la matrice de poids $\mathbf{W}$ entre la couche d'entrée et la couche cachée, puis le vector de connexions $\mathbf{w}$ entre la couche cachée et le neurone de sortie. `W` est donc un tableau Numpy à deux dimensions (matrice) et `w` est un tableau Numpy à une dimension (vector). Plus spécifiquement, la valeur de la connexion entre le i^{e} neurone caché et la j^{e} entrée x_j correspond à `W[i,j]`. De plus, la connexion entre le i^{e} neurone caché et le neurone de sortie correspond à `w[i]`.
- `parametres(self)` retourne la paire `(W,w)` de la matrice de connexions $\mathbf{W}$ (c'est-à-dire une matrice Numpy `W`) et du vector de connexions $\mathbf{w}$ (c'est-à-dire un vector Numpy `w`) du réseau de neurones.
- `prediction(self, x)` retourne la prédiction par le réseau de neurones de la classe d'une entrée représentée par un vecteur Numpy `x`. Cette prédiction doit donc être 0 ou 1.
- `mise_a_jour(self, x, y)` met à jour les paramètres du réseau de neurones à l'aide de sa règle d'apprentissage, à partir d'une entrée `x` (vecteur Numpy) et de sa classe cible `y` (0 ou 1).
- `entrainement(self, X, Y)` entraîne le réseau de neurones durant T itérations sur l'ensemble d'entraînement formé des entrées `X` (une matrice Numpy, où la t^{e} rangée correspond à l'entrée $\mathbf{x}_t$) et des classes cibles `Y` (un vecteur Numpy où le t^{e} élément correspond à la cible y_t). Il est recommandé d'appeler votre méthode `mise_a_jour(self, x, y)` à l'intérieur de `entrainement(self, X, Y)`.

Votre implémentation de cette classe doit être placée dans un fichier `solution.py`, qui sera importé lors de la procédure automatique de correction. Le fichier `devoir_4.py` contient un exemple d'exécution de votre code. Vous pouvez l'appeler comme un script. Ce script nécessite également que les fichiers `train2.pkl`, `test2.pkl` et `parametres_attendus.pkl` soient présents dans le même répertoire. Une implémentation correcte obtiendra une erreur d'entraînement de 0% et une erreur de test de 0%. `devoir_4.py` compare également les valeurs de paramètres trouvées par votre implémentation et celles trouvées par une implémentation correcte.

La correction se fera de façon automatique. Veuillez soumettre votre fichier `solution.py` via l'outil **turnin**, avant la date de remise. À noter que **tout autre fichier soumis sera ignoré** : votre implémentation doit être entièrement comprise dans le fichier `solution.py`.

5. **[BONUS]** Une compagnie fabriquant des voitures a trois lignes de production : la ligne A, la ligne B et la ligne C. La compagnie fabrique trois sortes de voitures : une voiture sportive, une voiture familiale et une voiture tout-terrain. Chaque ligne de production fonctionne durant 4 périodes de 2 heures dans une journée : de 8h à 10, de 10 à 12h, de 12h à 14h et de 14h à 16h. La construction d'une voiture dans une ligne nécessite une de ces périodes de 2 heures complète. De plus :

   – La ligne A peut fabriquer la voiture sportive et la voiture familiale seulement.
   – La ligne B peut fabriquer la voiture tout-terrain seulement.
   – La ligne C peut fabriquer la voiture familiale et la voiture tout-terrain seulement.

   Supposons que la compagnie reçoive une commande d'un client pour 2 voitures sportives, 3 voitures familiales et 5 voitures tout-terrain. De plus, on exige que toutes les voitures familiales et toutes les voitures tout-terrain aient été assemblées avant 14h.

   Pour **2 points** boni :

   (a) Formulez ce problème comme un problème de satisfaction de contraintes. Identifiez (1) toutes les variables du problème, (2) le domaine de chacune des variables et (3) l'ensemble des contraintes à satisfaire.

   (b) À partir de votre formulation en (a), montrez qu'il est possible pour la compagnie de satisfaire la demande de son client, à l'aide de l'algorithme *backtracking search*. **Spécifiez d'abord un ordre des variables et des valeurs de domaine**. Donnez ensuite la trace de l'exécution de *backtracking search*, en énumérant la liste des ajouts et retraits d'assignation de variable à une valeur. Il est important de respecter les ordres de variables et de valeurs que vous avez choisis dans votre simulation de *backtracking search*. N'utilisez pas d'inférence.

   Indice : pour simplifier votre solution, faites un choix judicieux de l'ordre des variables et des valeurs, qui vous évitera de faire souvent du *backtracking*.