IFT 615 : Devoir 3
Travail individuel

Remise : 29 mars 2012, 16h20 (**au plus tard**)

1. [**2 points**] Soit un modèle de Markov caché d'ordre 1 dont les variables cachées H_t et les variables observées S_t ont toutes comme domaine les symboles a, b, c. Soit les distributions de transition et d'émission suivantes :

	$H_t = a$	$H_t = b$	$H_t = c$
$P(S_t = a \mid H_t)$	0.8	0.4	0.1
$P(S_t = b \mid H_t)$	0.1	0.4	0.3
$P(S_t = c \mid H_t)$	0.1	0.2	0.6

	$H_{t-1} = a$	$H_{t-1} = b$	$H_{t-1} = c$
$P(H_t = a \mid H_{t-1})$	0.2	0.1	0.6
$P(H_t = b \mid H_{t-1})$	0.7	0.1	0.2
$P(H_t = c \mid H_{t-1})$	0.1	0.8	0.2

Soit également les probabilités initiales $P(H_1 = a) = 0.4$, $P(H_1 = b) = 0.4$ et $P(H_1 = c) = 0.2$ de la variable cachée au temps $t = 1$.

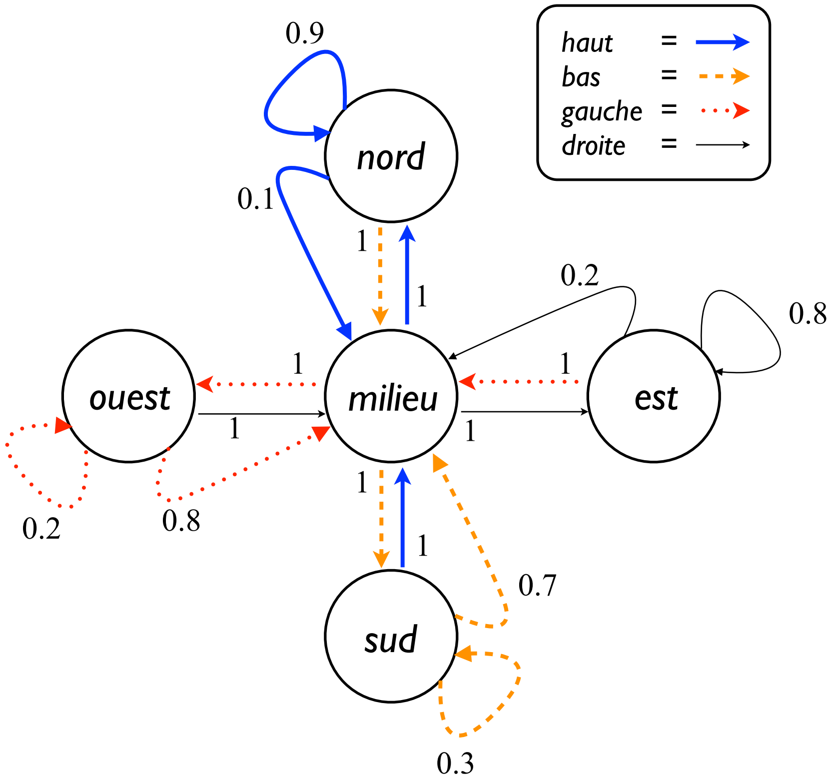
(a) Calculez la distribution de filtrage $\mathbf{P}(H_3 \mid S_1 = b, S_2 = b, S_3 = c)$.
Indice : utilisez le programme dynamique de $\alpha(i,t) = P(S_{1:t} = s_{1:t}, H_t = i)$.

(b) Calculez la distribution de lissage $\mathbf{P}(H_2 \mid S_1 = b, S_2 = b, S_3 = c)$.
Indice : utilisez le programme dynamique de $\beta(i,t) = P(S_{t+1:T} = s_{t+1:T} \mid H_t = i)$ où $T = 3$, ainsi que le tableau $\alpha(i,t)$ calculé en (a).

(c) Calculez la distribution de prédiction $\mathbf{P}(H_4 \mid S_1 = b, S_2 = b, S_3 = c)$.
Indice : utilisez le programme dynamique $\pi(i,k) = P(H_{t+k} = i \mid S_{1:t} = s_{1:t})$, initialisé à $\pi(i,0) = \alpha(i,3) / \sum_{j \in \{a,b,c\}} \alpha(j,3)$ à partir du tableau $\alpha(i,t)$ calculé en (a).

(d) Trouvez l'explication la plus plausible, c'est-à-dire la valeur la plus vraisemblable de H_1 H_2 et H_3 étant donnée la séquence observée $S_1 = b$, $S_2 = b$ et $S_3 = c$.
Indice : utilisez le programme dynamique $\alpha^*(i,t) = P(S_{1:t} = s_{1:t}, H_{1:t-1} = h^*_{1:t-1}, H_t = i)$.

2. [**2 points**] Soit le processus de décision markovien suivant :



où la fonction de récompense est telle que $R(nord) = -1$, $R(ouest) = 1$, $R(milieu) = 0$, $R(est) = 2$ et $R(sud) = 3$ et le facteur d'escompte est $\gamma = 0.5$. L'ensemble des états est ainsi $S = \{milieu, nord, sud, est, ouest\}$ et l'ensemble complet des actions est $gauche, droite, haut, bas$.

(a) Calculez le tableau de valeur $V(\pi, s)$ pour la politique $\pi = \{nord \rightarrow haut, ouest \rightarrow droite, milieu \rightarrow bas, est \rightarrow droite, sud \rightarrow haut\}$. Vous pouvez utiliser Python pour calculer la solution du système d'équations linéaires à résoudre, en utilisant la fonction `numpy.linalg.inv`.

(b) Donnez toutes les étapes de l'algorithme *policy iteration* appliqué à ce MDP, en utilisant la politique en (a) comme politique initiale.

(c) Donnez l'exécution de deux itérations de l'algorithme *value iteration* appliqué à ce MDP, en utilisant comme tableau de valeurs $V(s)$ initiales : $V(nord) = -1$, $V(ouest) = 1$, $V(milieu) = 0$, $V(est) = 2$ et $V(sud) = 3$.

3. **[2 points]** Soit l'ensemble d'entraînement suivant :

$\mathbf{x}_t$	y_t
$[4, 4, 0]$	0
$[1, 2, 4]$	0
$[2, 2, 2]$	0
$[8, 0, 0]$	0
$[1, 1, 1]$	0
$[2, 5, 5]$	1
$[3, 3, 3]$	1
$[0, 0, 9]$	1
$[1, 3, 5]$	1
$[5, 5, 3]$	1

Soit une entrée de test $\mathbf{x} = [4.2, 2.1, 3.7]$.

(a) Donnez la classe de $\mathbf{x}$ qui serait prédite par l'algorithme des k plus proches voisins basé sur la distance Euclidienne $d_1(\mathbf{x}, \mathbf{x}') = \sqrt{\sum_i (x_i - x'_i)^2}$, **et ce pour** $k = 1$**,** $k = 3$ **et** $k = 5$.

(b) Donnez également les prédictions pour $k = 1$, $k = 3$ et $k = 5$, mais pour la distance de Manhattan $d_2(\mathbf{x}, \mathbf{x}') = \sum_i |x_i - x'_i|$.

4. **[2 points]** Soit la fonction :

$$g(\mathbf{x}) = \frac{x_1 + x_2^2 - \log(x_3)}{\exp(x_2) + x_4}$$

Calculez toutes les dérivées partielles, c'est-à-dire :

(a) $\frac{\partial g(\mathbf{x})}{\partial x_1}$

(b) $\frac{\partial g(\mathbf{x})}{\partial x_2}$

(c) $\frac{\partial g(\mathbf{x})}{\partial x_3}$

(d) $\frac{\partial g(\mathbf{x})}{\partial x_4}$

5. **[2 points]** Implémentez une classe Python `Perceptron` correspondant à l'algorithme du Perceptron à 2 classes ($y = 0$ et $y = 1$). Pour ce faire, complétez l'implémentation des méthodes suivantes :

```
class Perceptron:

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

    def initialisation(self, w, b):
        # 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, b)` initialise le vecteur de poids $\mathbf{w}$ et le biais b du Perceptron (c'est-à-dire ses paramètres) aux valeurs contenues dans le vecteur Numpy `w` et le nombre à virgule flottante `b`, respectivement.
- `parametres(self)` retourne la paire `(w,b)` du vecteur de poids $\mathbf{w}$ (c'est-à-dire un vecteur Numpy `w`) et du biais b (c'est-à-dire un nombre à virgule flottante `b`) du Perceptron.
- `prediction(self, x)` retourne la prédiction par le Perceptron 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 Perceptron à l'aide de sa règle d'apprentissage, à partir de d'une entrée `x` (vecteur Numpy) et de sa classe cible `y` (0 ou 1).
- `entrainement(self, X, Y)` entraîne le Perceptron 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_3.py` contient un exemple d'exécution de votre code. Vous pouvez l'appeler comme un script. Ce script nécessite également que les fichiers `train.pkl`, `test.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 10%. `devoir_3.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`.

6. **[BONUS]** Pour **2 points** boni, soit le réseau bayésien ayant les tables de probabilités conditionnelles suivantes :

A	*C*	*F=vrai*
faux	*faux*	0.1
faux	*vrai*	0.2
vrai	*faux*	0.8
vrai	*vrai*	0.7

E	*C=vrai*
faux	0.2
vrai	0.4

D=vrai
0.2

B	*D*	*E*	*A=vrai*
faux	*faux*	*faux*	0.7
faux	*faux*	*vrai*	0.2
faux	*vrai*	*faux*	0.5
faux	*vrai*	*vrai*	0.1
vrai	*faux*	*faux*	0.2
vrai	*faux*	*vrai*	0.9
vrai	*vrai*	*faux*	0.8
vrai	*vrai*	*vrai*	0.6

E=*vrai*
0.9

B=vrai
0.7

(a) Calculez la distribution $\mathbf{P}(A|E = vrai)$

(b) Calculez la distribution $\mathbf{P}(C|F = vrai, E = vrai)$

(c) Est-ce que F et E sont indépendantes conditionnellement, sachant A ? Justifiez votre réponse.

(d) Est-ce que A et C sont indépendantes sachant E ? Justifiez votre réponse.