Intelligence artificielle : Applications
Examen de la session principale

Avril-2026

Durée : 2 heures. **Aucun document autorisé**. Calculatrices simples autorisées. Les exercices sont indépendants.

---

1. Optimisation : questions de cours et de réflexion

1. Dans quel cas utiliser une métaheuristique plutôt qu'une méthode exacte pour résoudre un problème d'optimisation ?
2. Dans l'algorithme du recuit simulé quelle est l'influence d'une température trop élevée sur l'exécution de l'algorithme ?
3. Dans l'algorithme du recuit simulé quelle est l'influence d'une température trop basse sur l'exécution de l'algorithme ?
4. Comment estimer une valeur initiale du paramètre de température "convenable" à une exécution "cohérente" de l'algorithme ?

2. Deep Learning I (Convnet) : questions de cours et de réflexion

1. Donner les principales différences qu'il y a entre les réseaux de neurones convolutifs et les réseaux de neurones 'standards' (vus au premier semestre).
2. Décrire brièvement le fonctionnement et l'utilité des couches de convolution et de pooling, et donner la manière dont sont définis leurs nombres de neurones.

3. Quels sont les hyperparamètres et les paramètres d'un convnet (à détailler type de couche par type de couche) ?

3. Deep Learning II (RNN)

On mesure la température dans une ville X le matin, l'après-midi et le soir. Chaque observation (un jour) est donc caractérisée par 3 températures et on lui associe un label journalier parmi les 5 suivants : très chaude, chaude, tempérée, fraîche, très froide.

On souhaite prédire le **label** du 5e jour en se basant sur les **températures** des 4 jours précédents (3 températures par jour, donc). Nous disposons de 10000 observations séquentielles (une observation = un jour).

1. Donnez le nombre d'exemples obtenus à partir de ces 10000 observations.
2. On divise le dataset en training set (7000 observations) et test set (le reste). Donnez les dimensions de *Xtrain* et *ytrain*, ainsi que de *Xtest* et *ytest* (notations du TD).
3. Appliquez le codage one-hot sur les labels pour tenir compte des 5 *classes. Donnez les dimensions* de *ytrain* après encodage.
4. Pour la construction du RNN, nous utilisons le code suivant :

   model = Sequential()
   model.add(SimpleRNN(32, input_shape=(1.?)))
   model.add(Dense(2.?, activation='3.?'))

   Remplacez 1.?, 2.? et 3.? par les valeurs correctes.
5. Dessinez le RNN, puis sa version **dépliée** sur 4 pas de temps.
6. Donnez le nombre de paramètres de chaque couche.

4. Apprentissage par renforcement

On considère une grille 5×5. La case centrale est un **état puits**, et la case en bas à droite est une **cible à atteindre**.

L'agent peut se déplacer vers la gauche, la droite, le haut ou le bas, vers une case adjacente.

On suppose en plus que :

— l'agent n'essaie jamais de sortir de la grille,

— L'agent ne quitte jamais l'état puits ni l'état cible une fois qu'il les atteint.

On suppose enfin que les récompenses sont définies comme suit :

- De n'importe quel état vers l'état puits : -30
- De n'importe quel état vers l'état cible : $+100$
- Vers tout état partageant la même ligne ou la même colonne que l'état puits : -10
- Vers toutes les autres cases : -2

1. Donner le graphe décrivant les états et les arêtes étiquetées par les récompenses. En déduire le processus de décision markovien.
2. Trouver 'à la main' la meilleure stratégie pour l'agent.
3. On souhaite utiliser l'algorithme de *Value iteration* avec un nombre d'itérations égal à 1000. Donner le nombre de fois que chaque valeur $V(s)$, avec s un état, est mise à jour.
4. Donner deux critères d'arrêt possibles pour l'algorithme de *Value iteration*.
5. Entamer l'exécution de l'algorithme *Value iteration* dans notre exemple.
6. Quel est le résultat (l'output) de l'algorithme *Value iteration* ? Comment utiliser ce résultat pour déduire la stratégie optimae ?