Intelligence artificielle : Applications
Examen de la session principale

Avril-2026

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

---

1. Jeux à deux joueurs

On considère l'arbre d'un jeu (à deux joueurs) défini comme suit :

- Il comporte 5 niveaux (racine=niveau 1 → feuilles=niveau 5).
- La racine correspond au joueur Max.
- Chaque noeud non terminal possède exactement 2 fils.
- Les feuilles sont étiquetés par les valeurs suivantes (de gauche à droite) : 3, −5, −6, 7, 7, 8, 8, −3, −4, 6, 8, −3, 3, 2, −1, 5.

Travail demandé :

1. Construire l'arbre.
2. Expliquer la signification des valeurs des feuilles et la façon dont elles ont pu être obtenues.
3. Appliquer l'algorithme *Minimax* et donner la signification de la valeur de la racine ainsi obtenue.
4. Rappeler brièvement le principe et l'intérêt de l'élagage alpha-bêta (rappelé en annexe).
5. Entamer l'application de l'algorithme Minimax avec élagage alpha-bêta (s'arrêter au premier retour à la racine).
6. Expliquer comment on peut mesurer l'amélioration apportée par l'élagage alpha-bêta.

2. Deep Learning I (Convnet)

On considère le réseau de neurones créé par le code de la figure 1 ci-dessous.

1. Expliquez le sens et l'utilité des deux instructions situées juste après la ligne '# Convert class vector to binary ...'.
2. Décrire brièvement le fonctionnement et l'utilité des couches de convolution et de pooling.
3. Quels sont les **hyperparamètres** et les **paramètres** d'un tel réseau?
4. Expliquez brièvement ($3-4$ phrases) comment sont déterminés les **paramètres** du réseau.
5. Calculez le nombre de paramètres de chaque couche.
6. Justifiez le choix du nombre de neurones et de la fonction d'activation de la couche de sortie.

3. Deep Learning II (RNN)

Dans cet exercice, on considère un réseau de neurones récurrent conçu pour faire de l'**analyse de sentiment**. Autrement dit, ce RNN reçoit en entrée un texte et donne en sortie l'opinion/le sentiment exprimé par ce texte. On supposera qu'il y a **trois types de sentiments : positif, négatif et neutre**. On supposera aussi, pour simplifier, que tous les textes ont la même longueur : 40 mots. On supposera enfin que chaque mot est représenté par un vecteur de taille 50.

1. Donnez le nombre de neurones des couches d'entrée et de sortie (vous justifierez, bien entendu, votre réponse).
2. Quelle est la fonction d'activation de la couche de sortie (vous justifierez, bien entendu, votre réponse).
3. On suppose que le nombre de neurones de la couche cachée est de 64. Donnez le nombre de paramètres de chaque couche du réseau.

4. Apprentissage par renforcement

On considère un agent qui se déplace sur le terrain de la figure 2 dans lequel la case verte représente un objectif et la case rouge un 'piège'.

1. Proposer une **modélisation** (détaillée et argumentée) de ce problème.
2. On souhaite appliquer l'algorithme Q-learning à notre problème.

```
from keras.datasets import mnist
# MNIST data, shuffled and split between train and test sets
(X_train, y_train), (X_test, y_test) = mnist.load_data()
# Some pre-processing
X_train = X_train.reshape(60000, 784)
X_test = X_test.reshape(10000, 784)
X_train = X_train.astype('float32')
X_test = X_test.astype('float32')
X_train /= 255
X_test /= 255
# convert class vectors to binary class matrices
Y_train = np_utils.to_categorical(y_train, nb_classes)
Y_test = np_utils.to_categorical(y_test, nb_classes)
```

```
s=(5,5)
ish=(28,28,1)
model = Sequential()
model.add(Conv2D(32,kernel_size=s,activation='sigmoid',input_shape=ish,padding='same'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(64, (5, 5), activation='sigmoid', padding='same'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Flatten())
model.add(Dense(100, activation='sigmoid'))
model.add(Dense(nb_classes, activation='softmax'))
```

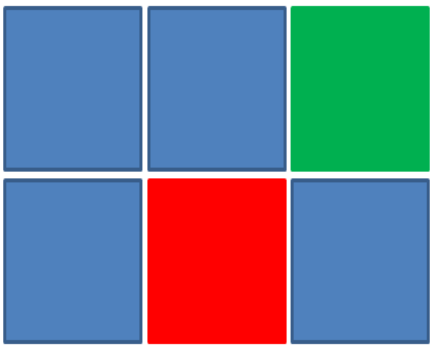

FIGURE 2 – Apprentissage par renforcement

(a) Quelle sera la sortie de cet algorithme ?
(b) Comment utiliser cette sortie pour trouver une stratégie optimale.
(c) Construire 'à la main' une telle stratégie.
(d) Comment cet algorithme prend-il en compte la dichotomie Exploration/Exploitation.

A Élagage alpha-bêta

— Minimax avec élagage alpha-bêta fonctionne comme suit :

1. Pour calculer la valeur v d'un noeud, l'algorithme utilise deux valeurs α et β telles que $\alpha \leq v \leq \beta$.

2. Le calcul de v se fait de manière récursive : pour calculer la valeur d'un noeud l'algorithme s'appelle lui-même pour chacun des fils.
3. α et β sont initialisées à $-\infty$ et ∞, respectivement.
4. Sur un noeud Max, soit r la valeur retournée pour un fils (Min donc).
   - Si $r \geq \beta$, l'algorithme s'arrête car par définition de β v ne peut pas être plus grand que β. Cela s'appelle une **coupe** β. L'algorithme retourne $r = \beta$.
   - Si $\alpha < r < \beta$, l'algorithme met à jour la valeur de $\alpha = r$.
   - Si $r \leq \alpha$, l'algorithme continue et prend en compte les autres fils.
   - Si tous les fils ont été explorés, l'algorithme retourne $r = \alpha$.
5. Sur un noeud Min, soit r la valeur retournée pour un fils (Max donc).
   - Si $r \leq \alpha$, l'algorithme s'arrête car par définition de α, v ne peut pas être plus petit que α. Cela s'appelle une **coupe** α. L'algorithme retourne $r = \alpha$.
   - Si $\alpha < r < \beta$, l'algorithme met à jour la valeur de $\beta = r$.
   - Si $r \geq \beta$, l'algorithme continue et prend en compte les autres fils.
   - Si tous les fils ont été explorés, l'algorithme retourne $r = \beta$.