IA-Applications
Examen de la session principale

09/04/2025

Durée : 2 heures. **Aucun documents autorisé**. Les exercices sont indépendants. Le barême est donné à titre indicatif.

---

1 Jeux (5 points)

L'objet de cet exercice est d'étudier l'extension de l'algorithme minimax aux **jeux non-déterministes** (e.g. le backgammon). Nous supposons que le déroulement du jeu est le suivant :

- Lorsque c'est son tour, le joueur (Min ou Max) commence par 'générer un événement aléatoire' (e.g. lancer une pièce ou un dé).
- Cet événement aléatoire (e.g. la pièce montre Pile) détermine les coups possibles que le joueur peut jouer.
- Le joueur joue l'un de ces coups.

L'algorithme **Expectiminimax**, qu'on peut appliquer dans ce cas, généralise l'algoritme **Minimax** en construisant un arbre de jeu comprenant un nouveau type de niveau (en plus des niveaux Min et Max) : c'est le niveau **Chance**. Chaque noeud d'un niveau Chance a autant de fils qu'il y a d'événements possibles dans le processus de génération, chaque branche d'un tel noeud est étiquetée par la probabilité de l'événement correspondant, et la valeur du noeud est la moyenne pondérée de ses fils.

1. Complétez l'arbre de l'exemple de la figure 1 ci-dessous.
2. Déduire le meilleur coup de Max.
3. Rappelez la signification de la valeur de la racine dans le cas de l'algorithme *Minimax* 'standard' et donnez celle dans le cas de l'algorithme *Expectiminimax*.

4. Représentez un noeud Chance lorsque l'événement aléatoire est généré (a) par le lancer d'une pièce, et (b) par le lancer deux fois d'un dé (e.g. le backgammon) sans prendre en compte l'ordre (autrement dit l'événement $\{4, 5\}$ par exemple correspond à la sortie d'un 4 puis d'un 5 ou l'inverse.

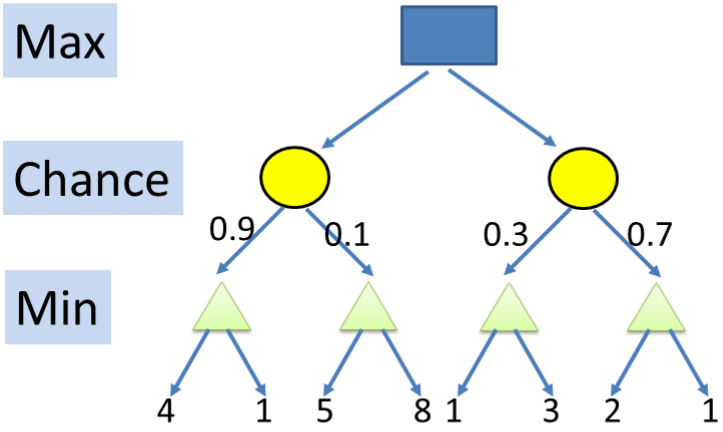

FIGURE 1 – Expectiminimax-Exemple

2 Optimisation (5 points)

Nous souhaitons appliquer le **recuit simulé** (RS) au problème du clustering défini comme suit :

— Étant donné n points de l'espace $\mathbb{R}^p$ et un entier k, répartir les n points sur k groupes (clusters) tels que deux points appartenant à un même cluster soient aussi proches que possible, et deux points appartenant à deux clusters différents soient aussi éloignés que possible.

1. Donnez une représentation des solutions possibles de ce problème adaptée à l'application du RS.

2. Quel est le nombre de ces solutions possibles (la taille de l'espace de recherche) ? En déduire que le RS peut être utile pour résoudre ce problème.
3. Comment définiriez-vous la relation de voisinage et l'énergie que le RS utilisera pour cette résolution.

3 Apprentissage par renforcement (6 points)

On considère un agent qui se déplace sur le terrain de la figure 2 dans laquelle la case verte (2ème case de la ligne 1) représente l'objectif et les 3 cases rouges des obstacles.

1. Proposez une **modélisation** (détaillée et argumentée) de ce problème.
2. Rappelez la définition d'une stratégie et donnez-en un exemple.
3. Rappelez l'utilité de l'algorithme Value Iteration (rappelé en annexe).
4. Déroulez les deux premières itérations de cet algorithme sur ce problème.

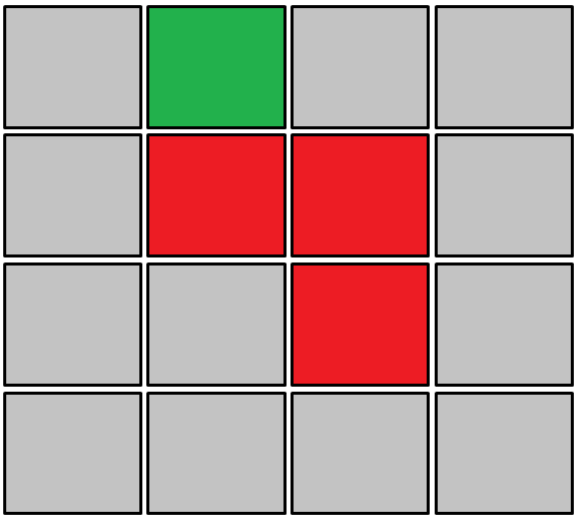

FIGURE 2 – Apprentissage par renforcement

4 Deep learning (6 points)

Dans cet exercice, nous nous intéressons au jeu de données *Fashion Mnist* composé de 60000 images de taille 28×28 en niveau de gris. Les étiquettes des données sont les suivantes (Figure 3) :

Label	Description
0	T-shirt/top
1	Trouser
2	Pullover
3	Dress
4	Coat
5	Sandal
6	Shirt
7	Sneaker
8	Bag
9	Ankle boot

FIGURE 3 – Classes du jeu de données Fashion Mnist

La classification des images est faite par le réseau de neurones créé par le code de la figure 4.

```
model = Sequential()

# ajout des couches au réseau de neurones
model.add(Conv2D(32, kernel_size=(3, 3), activation='relu', input_shape=input_shape))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(Conv2D(128, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(num_classes, activation=Activation(tf.nn.softmax)))
```

FIGURE 4 – Création d'un convNet

```
model.fit(X_train, y_train,
          batch_size=batch_size,
          epochs=epochs,
          verbose=1,
          validation_data=(X_test, y_test))
```

FIGURE 5 – Entrainement du modèle

Il vous demandé de répondre aux questions suivantes :

1. Quels sont les avantages d'un réseau de neurones convolutif (Convnet) sur un réseau entièrement connecté pour la classification d'images ?
2. Quel est l'intérêt de la fonction d'activation *Softmax* à la dernière couche ?
3. Décrire de manière synthétique chacune des couches.
4. Quelles sont les couches qui n'ont pas de paramètres ? Donnez pour chacune d'elles la liste des hyperparamètres.
5. Après avoir compilé le modèle, on lance l'entraînement (figure 5). À quoi correspondent les paramètres de *batch size* et *epochs* ?

A L'algorithme Value Iteration

1. Initialiser $V(s)$, $s \in S$ à des valeurs quelconques.
2. Répéter jusqu'à (critère d'arrêt atteint)
   (a) Pour chaque état s
      i. Pour chaque action a
      $Q(s,a) = R(s,a) + \gamma \sum_{s'} T(s,a,s')V(s')$
      ii. $V(s) = max_a Q(s,a)$