Université Mohamed Premier Oujda
École Nationale de l'Intelligence Artificielle et du Digital Berkane
Année universitaire : 2024 / 2025

Filière : IA
Prof : Mohamed Khalifa BOUTAHIR

MACHINE LEARNING II – TP 3

Objectif de TP - Optimisation des Feux de Circulation avec Apprentissage par Renforcement

L'objectif de ce TP est d'explorer l'optimisation des feux de circulation à l'aide de l'apprentissage par renforcement. Les étudiants vont :

- Découvrir un environnement simulé de gestion du trafic.
- Implémenter Q-Learning et SARSA pour apprendre une politique optimale.
- Comparer les résultats des deux algorithmes à l'aide de graphiques et d'évaluations quantitatives.

Exercice 1 : Découverte de l'Environnement

Instructions :

1. Téléchargez le fichier de l'environnement (*env-traffic.py*) depuis Google Classroom.
2. Installez les dépendances nécessaires si ce n'est pas encore fait.
3. Importez et exécutez l'environnement pour tester son fonctionnement.

```
from env_traffic import TrafficEnvironment

env = TrafficEnvironment()
state = env.reset()

for _ in range(10):
    action = 0  # Garder le feu tel qu'il est
    next_state, reward = env.step(action)
    print(f"État : {next_state}, Récompense : {reward}")
```

Exercice 2 : Implémentation de Q-Learning

Instructions :

1. Initialisez une Q-Table pour stocker les valeurs des actions pour chaque état.
2. Implémentez l'algorithme Q-Learning, en mettant à jour la Q-Table à chaque itération.
3. Utilisez une stratégie ε-greedy pour gérer l'exploration/exploitation.
4. Exécutez l'apprentissage sur plusieurs épisodes et Affichez la Q-Table finale après l'entraînement.

Le Q-Learning suit la mise à jour suivante : $Q(s,a) \leftarrow Q(s,a) + \alpha \left[R + \gamma \max_{a'} Q(s',a') - Q(s,a)\right]$

```
# Initialisation de la Q-Table
q_table = np.zeros((10, 10, 10, 10, 2))

def train_q_learning(env, episodes=1000, alpha=0.1, gamma=0.9, epsilon=1.0,
decay=0.995):
.....
```

Exercice 3 : Implémentation de SARSA

Instructions :

1. Créez une nouvelle Q-Table pour SARSA.
2. Mettez en œuvre l'algorithme SARSA, qui met à jour la Q-Table avec la valeur de l'action effectivement choisie.
3. Utilisez une stratégie ε-greedy.
4. Affichez la Q-Table finale et comparez avec celle de Q-Learning.

SARSA suit la mise à jour suivante : $Q(s,a) \leftarrow Q(s,a) + \alpha\,[R + \gamma Q(s',a') - Q(s,a)]$

```
def train_sarsa(env, episodes=1000, alpha=0.1, gamma=0.9, epsilon=1.0,
decay=0.995):
    for episode in range(episodes):
...
```

Exercice 4 : Analyse et Visualisation des Résultats

Instructions :

1. Générez un graphique montrant l'évolution des récompenses au fil des épisodes.
2. Comparez la rapidité d'apprentissage entre les deux algorithmes.
3. Affichez les meilleures politiques apprises.
4. Ajoutez les scores enregistrés pendant l'apprentissage.
5. Interprétez le graphique : Quel algorithme apprend plus vite ?

```
plt.plot(q_learning_rewards, label="Q-Learning")
plt.plot(sarsa_rewards, label="SARSA")
plt.xlabel("Épisodes")
plt.ylabel("Récompense Cumulative")
plt.legend()
plt.show()
    ...
```