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 2

Objectif :

L'objectif de ce TP est de mettre en pratique les concepts fondamentaux de l'apprentissage par renforcement en explorant l'algorithme Q-Learning. À travers une série d'exercices progressifs, les étudiants vont apprendre à implémenter cet algorithme, comprendre l'impact des stratégies d'exploration et d'exploitation, et analyser la convergence des valeurs Q. L'environnement FrozenLake de OpenAI Gym servira de terrain d'expérimentation, permettant d'illustrer concrètement comment un agent apprend à optimiser ses décisions grâce aux mises à jour successives de sa Q-table.

Exercice 1 : Exploration de l'Environnement FrozenLake

Instructions :

1. Charger l'environnement **FrozenLake-v1** de OpenAI Gym.
2. Afficher les informations de l'espace d'états et d'actions.
3. Exécuter une boucle où l'agent prend des actions aléatoires pendant plusieurs épisodes.
4. Observer les observations et les récompenses obtenues.

```
import gymnasium as gym
import numpy as np

# Charger l'environnement FrozenLake
env = gym.make("FrozenLake-v1", is_slippery=True)
...
```

Exercice 2 : Implémentation de la Q-Table et Initialisation

Instructions :

1. Créer une **Q-Table** de dimension `(nombre d'états x nombre d'actions)`, initialisée à 0.
2. Afficher la Q-Table avant l'apprentissage.
3. Vérifier que chaque état a une liste de valeurs associées aux actions possibles.

```
# Initialisation de la Q-Table
q_table = ...
# Affichage de la Q-Table initiale
print("Q-Table initialisée :")
print(q_table)
```

Exercice 3 : Implémentation du Q-Learning avec Mise à Jour

Instructions :

1. Définir les **hyperparamètres** : taux d'apprentissage (alpha), facteur de discount (gamma), epsilon pour l'exploration.
2. Mettre à jour la Q-Table en appliquant la règle de mise à jour du Q-Learning :

$$Q(s,a) = Q(s,a) + \alpha\,[R + \gamma \max Q(s',a') - Q(s,a)]$$

3. Exécuter plusieurs épisodes et observer l'évolution de la table.

```
# Paramètres
alpha = 0.1 # Taux d'apprentissage
gamma = 0.99 # Facteur de discount
epsilon = 1.0 # Exploration initiale
epsilon_decay = 0.995 # Décroissance d'epsilon
num_episodes = 5000 # Nombre d'épisodes

# Boucle d'apprentissage
for episode in range(num_episodes):
...
```

Exercice 4 : Évaluation des Performances de l'Agent

Instructions :

1. Lancer 100 épisodes en utilisant uniquement l'action optimale (argmax(Q[s, a])).
2. Mesurer le taux de réussite de l'agent (nombre de fois où il atteint l'objectif).
3. Comparer les performances avec celles obtenues dans l'Exercice 1 (actions aléatoires).

```
num_test_episodes = 100
successes = 0
for _ in range(num_test_episodes):
    ...
```