Manuel Développeur - Projet Méthodes Numériques

Par : Abdallah Abdelsadek et Adam Tawfik

1. Présentation Générale

Ce projet implémente une bibliothèque d'algorithmes de Renforcement (**RL**) en Java, capable de s'appliquer à différents environnements comme un jeu simplifié ou le morpion (**TicTacToe**).

2. Structure du Code

- **fr.polytech.mnia** : contient tous les agents, runners, environnements et interfaces.
- **Evironnement.java** : interface commune à tous les environnements.
- **Agent.java** : interface commune aux agents de RL.
- **TicTacToe / SimpleEnv** : environnements spécifiques.
- **QLearning, PolicyIteration, ValueIteration, etc.** : agents spécifiques.

3. Agents Implémentés

- **QLearning** : algorithme tabulaire avec epsilon-greedy.
- **EpsilonGreedySimple** : pour problème simple sans état.
- **UCBSimple** : UCB classique pour problème multi-bras.
- **PolicyIteration et ValueIteration** : classiques MDP.
- **GradientBanditSimple** : version sans état avec préférence sur les actions.

4. Utilisation de la Bibliothèque

- **Compilation** : `mvn compile`
- **Exécution** : `mvn exec:java`
- Exécution typique : **algo** = **"q"**, **env** = **"tictactoe"** lance un apprentissage Q-Learning sur le morpion.

5. Ajout de Nouveaux Algorithmes

- Implémenter une classe qui hérite de **Agent.java**.
- Implémenter les méthodes : `learn(), chooseAction(...), getQValue(...).`
- Ajouter un cas dans **TicTacToeRunner** ou **SimpleRunner**.

6. Fonctions Utiles

- **printResults** : Afficher les Q-values de l'état initial.
- **playHumanVsAgent** : Permet de tester l'agent face à un joueur humain.
- **Simulation** stratégique : permet de tester l'agent contre un adversaire qui minimise ses Q-values.

7. Exemples de Sortie

- Épisode 1 terminé en **5** coups.
- Résultats depuis l'état initial : **Action [2,2] -> Q = 2.1**
- Partie humaine contre l'agent...