Février 2026

TP3

Jeu Easy21

Apprentissage par Renforcement

Présentation

Le but de ce TP est d'appliquer des méthodes d'**apprentissage par renforcement** à un jeu de cartes simplifié appelé **Easy21**. Cet exercice s'inspire de l'exemple du Blackjack présenté dans *Sutton et Barto* (section 5.3) – notez cependant que les règles du jeu sont différentes et non standard.

Règles du jeu Easy21 :

- Le jeu utilise un jeu de cartes **infini** (les cartes sont tirées avec remise).
- Chaque tirage produit une valeur entre 1 et 10 (distribution uniforme) avec une couleur : **rouge** (probabilité $1/3$) ou **noire** (probabilité $2/3$).
- Il n'y a ni as ni figures dans ce jeu.
- En début de partie, le joueur et le croupier tirent chacun **une carte noire** (entièrement observable).
- À chaque tour, le joueur peut **tirer** (*hit*) ou **rester** (*stick*).
- Les cartes noires sont **ajoutées** à la somme du joueur ; les cartes rouges sont **soustraites**.
- Si la somme du joueur dépasse 21 ou passe en dessous de 1, le joueur est **éliminé** et perd (récompense -1).
- Si le joueur reste, le croupier joue : il reste toujours sur une somme ≥ 17 et tire sinon. Si le croupier est éliminé, le joueur gagne ; sinon, la victoire (récompense $+1$), la défaite (-1) ou l'égalité (0) est déterminée par celui qui a la plus grande somme.

Contraintes et Remise du TP

- Le code sera écrit en **Python**. Organisez vos fichiers de manière lisible en les nommant en cohérence avec les numéros de questions (ex. : `question2.py`, `question3.py`, etc.).
- Documentez vos fonctions et classes avec des *docstrings*.
- Les TP sont à réaliser en **binômes** et à rendre dans le dossier dédié sur le Moodle InfoRob. Vous avez **8h** pour réaliser le TP ; le rendu pris en compte sera le dernier déposé avant minuit à la fin du prochain TP.
- Vous déposerez **une archive unique** contenant tout votre code source, ainsi qu'un **document PDF** contenant vos graphiques et votre discussion.

Exercice 1: Implémentation de l'environnement Easy21 (10 points)

Écrire un environnement qui implémente le jeu Easy21. Plus précisément, écrire une fonction nommée `step` qui prend en entrée un état s (première carte du croupier 1–10, et somme du joueur 1–21) et une action a (*hit* ou *stick*), et qui retourne un échantillon de l'état suivant s' (potentiellement terminal si la partie est terminée) ainsi que la récompense r.

1. 6 points Cet environnement sera utilisé pour de l'apprentissage **sans modèle** : vous ne devez pas représenter explicitement la matrice de transition du MDP. Il n'y a pas d'actualisation ($\gamma = 1$). Les actions du croupier font partie de l'environnement : appeler `step` avec l'action *stick* doit simuler l'intégralité du jeu du croupier et retourner la récompense finale ainsi que l'état terminal.

```python
# Exemple d'interface attendue
state = {'dealer': 5, 'player': 14}

# Action 'hit' : le joueur tire une carte
next_state, reward, terminal = step(state, action='hit')

# Action 'stick' : le croupier joue en entier, renvoie le resultat
next_state, reward, terminal = step(state, action='stick')
```

2. [4 points] Écrire également une fonction `draw_card()` simulant un tirage (valeur entre 1 et 10 avec couleur rouge ou noire), et une fonction `init_game()` initialisant l'état de départ en faisant tirer une carte noire à chaque joueur.

Exercice 2: Contrôle Monte-Carlo dans Easy21 (15 points)

Appliquer la méthode de **contrôle Monte-Carlo** au jeu Easy21. Initialiser la fonction de valeur à zéro.

1. [8 points] Utiliser un pas d'apprentissage scalaire variable $\alpha_t = 1/N(s_t, a_t)$ et une stratégie d'exploration ε-greedy avec :

$$\varepsilon_t = \frac{N_0}{N_0 + N(s_t)}$$

où $N_0 = 100$ est une constante, $N(s)$ est le nombre de fois que l'état s a été visité, et $N(s, a)$ est le nombre de fois que l'action a a été sélectionnée depuis l'état s. Vous êtes libres de choisir une valeur différente pour N_0 si cela permet d'obtenir de meilleurs résultats.

2. [7 points] Tracer la fonction de valeur optimale $V^*(s) = \max_a Q^*(s, a)$ sous forme de surface 3D, avec en axes la carte du croupier (1–10) et la somme du joueur (1–21), similairement à la figure suivante tirée de l'exemple Blackjack de Sutton et Barto.

Exercice 3: Apprentissage TD dans Easy21 – Sarsa(λ) (15 points)

Implémenter l'algorithme **Sarsa(λ)** dans Easy21. Initialiser la fonction de valeur à zéro. Utiliser le même pas d'apprentissage et la même stratégie d'exploration que dans la section précédente.

1. [4 points] Exécuter l'algorithme avec les valeurs de paramètre $\lambda \in \{0;\ 0{,}1;\ 0{,}2;\ \ldots;\ 1\}$. Arrêter chaque exécution après **1 000 épisodes**.
2. [6 points] Calculer l'erreur quadratique moyenne (MSE) sur tous les états s et les actions a :

$$\text{MSE} = \sum_{s,\,a} \big(Q(s, a) - Q^*(s, a)\big)^2$$

en comparant les valeurs vraies $Q^*(s, a)$ calculées dans la section précédente aux valeurs estimées $Q(s, a)$ par Sarsa. Tracer la MSE en fonction de λ.

3. [5 points] Pour $\lambda = 0$ et $\lambda = 1$ **uniquement**, tracer la courbe d'apprentissage de la MSE en fonction du numéro d'épisode.

Exercice 4: Discussion (10 points)

Rédiger une discussion dans votre rapport PDF en répondant aux questions suivantes.

1. [1 point] Quels sont les **avantages et les inconvénients du bootstrapping** dans Easy21 ?
2. [1 point] Le bootstrapping serait-il plus utile dans le Blackjack classique ou dans Easy21 ? Justifier.
3. [2 points] Quels sont les avantages et les inconvénients de l'**approximation de la fonction de valeur** dans Easy21 ?
4. [1 point] Comment modifieriez-vous l'approximateur de fonction proposé dans la section précédente pour obtenir de **meilleurs résultats** ?

Total : 60 points. *Ce TP est à réaliser en binôme. Bon courage !*