Li Tingting
XU Arthur

Rapport de projet : Exploration / Exploitation

STATISTIQUE ET PROBABILITÉ 3I005

Sommaire :

I) Introduction

L'exploration et l'exploitation est un problème fondamental qu'on retrouve dans de nombreux domaines. L'exploitation consiste à faire la meilleure décision à partir de toute l'information collectée et l'exploration consiste à obtenir plus d'information.

On peut se demander vaut-il mieux exploiter la connaissance acquise et choisir l'action estimée la plus rentable ou vaut-il mieux continuer à explorer d'autres actions afin d'acquérir plus d'information.

Dans la suite, on va étudier différents algorithmes dans 3 cas different : en premier on va voir des algorithmes classiques d'exploration et exploitation dans le cadre d'un jeu de bandits-manchots. Puis on va utiliser l'algorithme de Monte-Carlo dans le jeu du Morpion. Enfin, on va voir l'algorithme UCB adapté aux arbres de jeu (UCT) aussi pour le jeu du Morpion.

II) Bandits-manchots

Dans cette première partie, on va coder un jeu de bandits-manchots en utilisant 4 algorithmes différents :

-l'algorithme aléatoire : on va jouer aléatoirement sans prendre en compte des informations qu'on peut collecter en fonction des jeu qu'on réalise.

-l'algorithme greedy : c'est un algorithme purement de l'exploitation. On jouer les premiers coup aléatoirement pour l'exploration et puis on va jouer tel que le rendement estimé est maximal.

-l'algorithme ε-greedy : on va appliquer l'algorithme greedy pour l'exploitation avec un probabilité qu'on choisi et dans les autres cas on va explorer en utilisant l'algorithme aléatoire.

-l'algorithme UCB : c'est un algorithme qui permet l'exploration et l'exploitation en même temps.

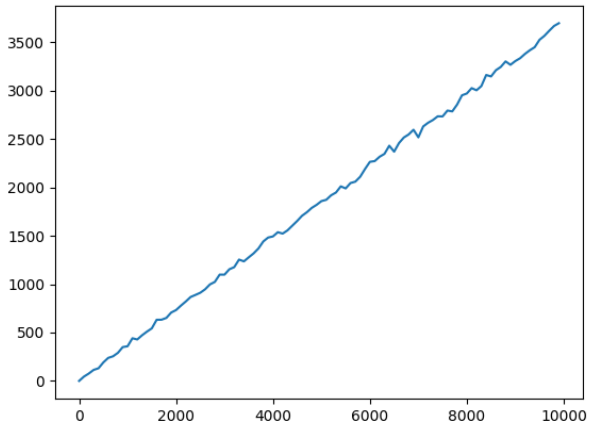
Dans la suite, on va observer le regret en fonction du temps des chaque algorithmes pour les étudier.

Observation du regret en fonction du temps des quatre algorithmes :

Nous allons fixé le nombre de fois qu'on va joueur au bandits-manchot à T = 10000. Pour réaliser ces expériences, nous allons fixé different nombre de levier et different probabilité de récompense lié à chaque levier afin d'observer leur courbe de regret en fonction du temps T.

Le regret ici est défini par la différence entre le gain maximal espéré G* et le gain du joueur GJ. Nous calculons G* en prenant la probabilité de récompense la plus haute multiplié par T, et GJ le nombre de fois le joueur à gagner au jeu.

a) Avec l'algorithme aléatoire

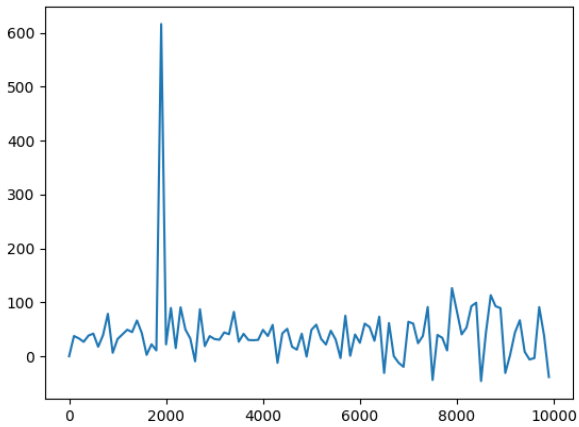


La liste de probabilité de récompense associé :
[0.11 ,0.21, 0.33, 0.43, 0.212, 0.666, 0.1234]

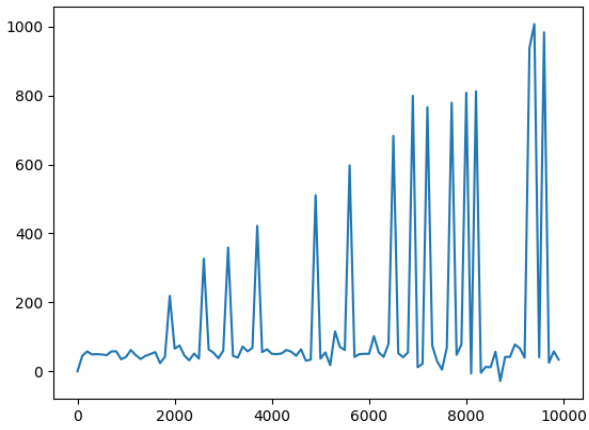
En réalisant plusieurs expériences en variant la liste de récompense, nous remarquons, comme on le voit sur le deuxième graphe, le regret augmente en fonction du nombre de fois qu'on a jouer. Cela veut dire que plus on joue de partie, plus on perd de l'argent.

b) Avec l'algorithme greedy

On fixe le nombre de jeu consacrés au début à l'exploration maxGreedy = 100. La formule utilisée ici est : arg = $argmax_{i\in\{1,\dots,N\}}\hat{u}_t^i$



La liste de probabilité de récompense associé :
[0.11 ,0.21, 0.33, 0.43, 0.212, 0.666, 0.1234]

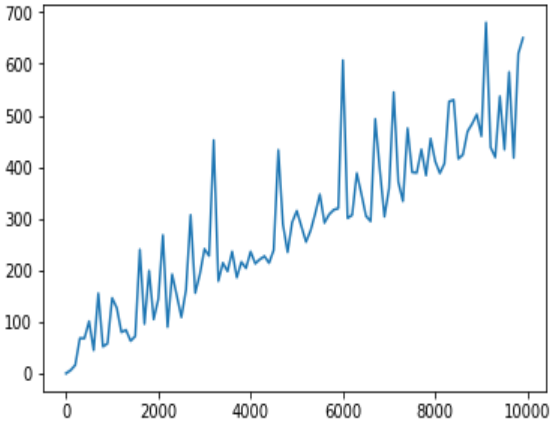


La liste de probabilité de récompense associé :
[0.11 ,0.21, 0.33, 0.43, 0.212, 0.666, 0.1234,0.8,0.9]

En réalisant plusieurs expérience en variant la liste de récompense, nous remarquons que le regret est en général autour de 0. Mais quand il y a un grand écart entre les paramètres de Bernoulli (comme on le voit sur les 2 graphes), on remarque des pics d'augmentation du regret.

c) Avec l'algorithme ε-greedy

On fixe ε = 0.1. On utilise donc l'algorithme greedy avec une probabilité de 1-ε et l'algorithme aléatoire avec une probabilité de ε.

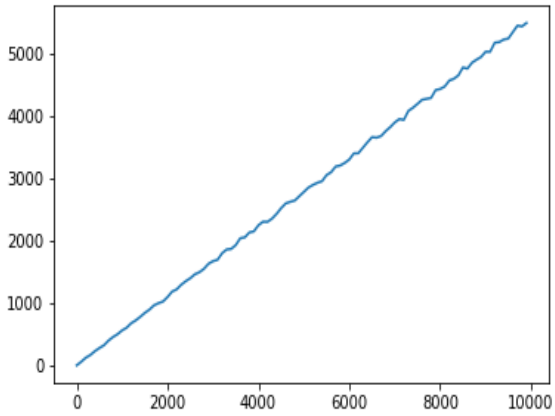


La liste de probabilité de récompense associé :
[0.11 ,0.21, 0.33, 0.43, 0.212, 0.666, 0.1234,0.8,0.9]

En réalisant plusieurs expérience en variant la liste de récompense, nous remarquons que le regret est en augmentation. On fixe le paramètre ε à 0.1 qui est la probabilité d'effectuer des opérations d'exploration et une probabilité de 0.9 pour effectuer les opérations d'exploitation (donc de jouer le levier avec le meilleur récompense actuellement). L'algorithme ε-greedy produit un regret augmentation au cours du temps.

d) Avec l'algorithme UCB

L'algorithme d'UCB consiste à choisir le levier: $I_t = \arg\max_{i\in(1,\dots,N)} UCB_{i,t-1}$avec $UCB_{i,t} = \hat{u}_{i,t} + \sqrt{\frac{2\log(t)}{N_t(i)}}$ où $\hat{u}_{i,t}$ est la moyenne empirique des récompenses reçues en ayant tiré le bras t (i.e., Xk,i est la i-ème récompense reçue en ayant tiré le bras k)



La liste de probabilité de récompense associé :
[0.11 ,0.21, 0.33, 0.43, 0.212, 0.666, 0.1234]

En réalisant plusieurs expérience en variant la liste de récompense,nous remarquons que le regret est en argumentation.

III) Morpion et Monte-Carlo

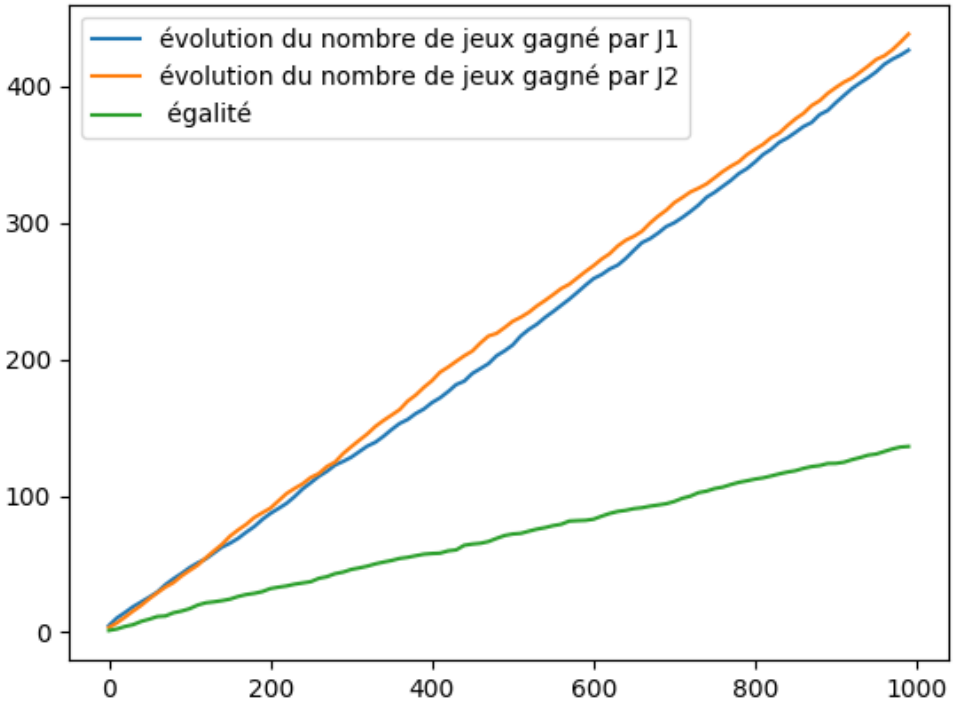
Dans cette partie, on va étudier l'algorithme de Monte-Carlo sur le jeu du Morpion. Le jeu du Morpion se joue à 2 joueurs sur un plateau de 3 × 3 cases. Chaque joueur à son tour marque une case avec son symbole (croix ou cercle) parmi celles qui sont libres. Le but est d'aligner pour un joueur 3 symboles identiques en ligne, en colonne ou en diagonale.

a) Joueur aléatoire

On lance dix jeu de T = 1000 parties. Les deux joueur jouent de façon aléatoirement, c'est à dire à chaque coup, le joueur va choisir un case aléatoirement parmi les cases possibles.

Remarque : Si on commence par le joueur 1 (respectivement joueur 2), la probabilité que le joueur 1 (respectivement joueur 2) gagne est beaucoup plus élevé. Donc pour chaque parties, on choisit aléatoirement le joueur qui commence à jouer.

En réalisant la moyenne des dix jeux, on obtient le graphe d'évolution des parties gagnées par le joueur 1 et le joueur 2 (et les parties où il y a égalité) :



On considère X la variable aléatoire le nombre de victoire du premier joueur. Le joueur peut soit gagner soit perdre (y compris les égalités) la partie, donc X suit une loi binomial de paramètre n = T (dans notre cas T = 1000) et dans notre expérience p = 426/1000 =0.43. La variance est V(X) = np(1-p) = 245.

De plus, on récupère :

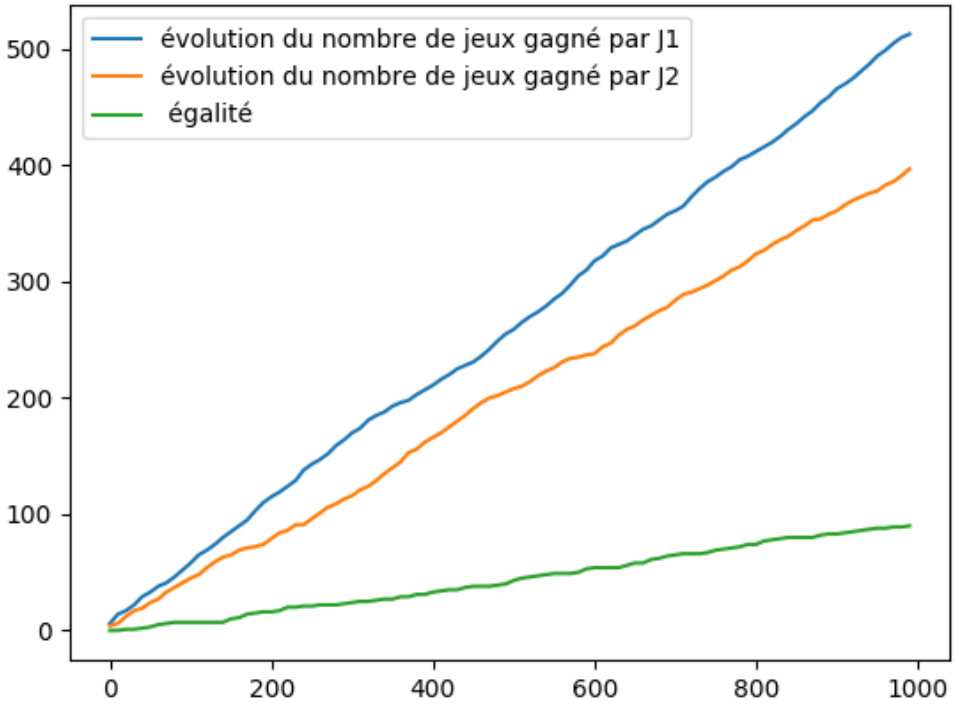
-le nombre de parties gagné par le joueur 1 est 426
-le nombre de parties gagné par le joueur 2 est 438
-le nombre de parties nulles est 136

En excluant les partie nulles, on obtient donc la probabilité pour le joueur 1 de gagner est de 0.493 et celle pour le joueur 2 est de 0.507
On conclut que si les deux joueurs joue de façon aléatoirement, ils ont la même probabilité de gagner et de perdre les parties de jeux.

b) Joueur Monte-Carlo

On lance un jeux de T = 1000 parties, à chaque partie on choisit aléatoirement le joueur qui commence à jouer. On fait jouer un joueur aléatoire et un joueur Monte-Carlo. Le joueur Monte-Carlo joue en fonction d'un algorithme qui s'agit d'échantillonner aléatoirement et de manière uniforme l'espace des possibilités et de rendre comme résultat la moyenne des expériences. Dans notre cas, c'est-à-dire le jeu du Morpion, le joueur va jouer pour chaque action possible un certain nombre de parties au hasard et d'en moyenner le résultat. Ici on fixe ce nombre de partie au hazard à 100.

On obtient le graphe d'évolution des parties gagnées par le joueur 1(qui joue en Monte-Carlo) et le joueur 2 :



De plus, on récupère :

-le nombre de parties gagné par le joueur 1 (Monte-Carlo) est 513
-le nombre de parties gagné par le joueur 1 (aléatoire) est 397
-le nombre de parties nulles est 90

En excluant les partie nulles, on obtient donc la probabilité pour le joueur Monte-Carlo de gagner est de 0.564 et celle pour le joueur aléatoire est de 0.436.
Donc on peut conclure que le joueur Monte-Carlo a une meilleur performance que le joueur aléatoire.