Assignment 1 - Reinforcement Learning

Nicolas THOU

11 Février 2020

1 Introduction

Dans cette partie nous avons étudié deux façon d'approximer la Q-fonction. La section 5 consistait à approximer le reward et la probabilité de passer d'un état x à un état x' via une action. Dans la section suivante, nous utilisons l'algorithme du Q-Learning qui est de mettre à jour une Q-table itérativement. Nous avons tenté d'observé l'influence des paramètres d'apprentissage et du *discount factor*. Ainsi, tout au long de ces sections nous devions comparer les state-functions J pour chaque politique obtenue.

2 Section 5

Dans cette section 5, nous devons utiliser le cours pour implémenter l'estimation de r(x,u), et de p(x2—x1, u). Nous ne pouvons pas afficher la vitesse de convergence de p et r car nous ne pouvons pas utiliser la librairie pyplot. Cependant, nous avons afficher une liste tel que la figure 1. Elle représente pour différentes simulations possible, le pourcentage d'égalité entre la valeur (de r ou de p) obtenue à la section 4 et la valeur obtenue à la section 5. Nous essayons d'obtenir des 100 pourcent à chaque fois mais parfois le pourcentage est supérieur à 100 car l'estimation est trop élevé par rapport à la vraie valeur.

Nous avons de plus, afficher les différentes valeurs de la state value function pour les différents états x pour deux expériences différentes (voir figure 2). Une première expérience à 1000 pas, et une deuxième expérience à 10 000 pas. On observe alors une différence de convergence du premier tableau dont les valeurs sont toutes négatives et du deuxième tableau où les valeurs deviennent positive mais toujours inférieur au tableau des différente valeur de la value-function optimal pour chaque état x obtenue dans le section 4.

Nous observons que nous sommes presque près des valeurs obtenues avec la policy optimale mais nous y sommes encore assez loin alors que nous avons simulé un épisode à 10 000 pas. D'où le problème de stockage des informations que l'algorithme du Q-learning n'a pas.

```
[200.          0.         200.          0.          0.
 200.          0.           0.          0.        100.
   0.          0.           0.        200.        100.
 100.          0.         133.33333333 100.        133.33333333
   0.        160.           0.        100.        133.33333333
 100.        133.33333333 133.33333333   0.        100.
  80.          0.           0.        150.        100.
   0.        200.         133.33333333   0.        100.
 100.        100.         133.33333333 100.          0.
  80.        200.           0.        100.          0.
 125.        100.         100.        100.        100.
  50.        100.         150.        133.33333333 120.
 100.         40.           0.        150.         66.66666667
   0.        100.         160.        150.         80.
 133.33333333  66.66666667 150.        100.        200.
 100.        125.          80.        160.        100.
 133.33333333 100.         100.          0.        100.
  90.90909091 120.         100.        150.         50.
 114.28571429 114.28571429  50.         60.         66.66666667
 100.        133.33333333  40.        100.       ]
```

Figure 1: similitude en pourcentage pour différentes simulation

```
 ====================== Display J_N_μ*_^ for each state x ==================
[[-48172.99480918 -30194.76630362 -31166.53383084 -43910.91264803
  -37688.90152414]
 [-55771.29780128 -53220.21793955 -55025.22742277 -45596.34318573
  -39366.06009111]
 [-33617.29318281 -40006.10706381 -54300.44047577 -51834.14204898
  -46626.96674726]
 [-56729.53938039 -87667.21671672 -58752.80838578 -36336.94143995
  -50952.82661627]
 [-44036.7238265  -39748.92997609 -27452.60745917 -44806.66995502
  -41765.5159703 ]]

here is the display for a huge trajectory, i-e 10000 step
[[    83.50283014     18.8395047      53.98912734  -4867.91955322
  -19276.33013447]
 [    66.16202701     49.96000818     19.19300772     73.25366326
     28.09664485]
 [    26.9688045      18.88247059     64.58036168     48.15890025
     69.25077528]
 [    48.96167332     45.00810795     61.83259379     59.26763625
     30.63944258]
 [    79.74682943     21.96259869     23.53012863     40.01641854
     59.97382061]]
 ====================== Display J_N_μ* for each state x ==================
[[174.65940775 102.39600614 111.87958197 174.97958675  82.95299653]
 [107.20321532 154.4208029  116.45643221 173.54952367 176.99026177]
 [127.8263329  164.66243498 160.34358465 158.07863945 181.76723359]
 [144.07072279 151.66656414 110.98320916 132.61431351 157.49331585]
 [121.15865372 142.45293655  92.57477483 133.51450806 127.12293471]]
```

Figure 2: Affichage des valeurs des value function pour chaque état, le premier tableau représente la première expérience (1000 pas), le deuxième tableau la deuxième expérience à 10 000 pas et enfin le dernier tableau représente la policy optimal.

3 Section 6

La première partie de cette section consistait uniquement d'appliquer le Q-learning Algortithme et d'en déduire une politique. Nous obtenons des valeurs plus grandes que pour la policy optimale obtenue en section 4 (voir figure 3). Cependant, on remarque que pour la trajectoire utilisé, la Q-table n'est pas remplie. On rappelle que la Q-table, les colonnes correspondant aux actions : down, up, right, left. L'estimation des reward ici fait de telle sorte que seul

les rewards égale à -3 (reward de la case (0,0) et de la case (4, 3)) ne peuvent être ajouté à la r-table que sous certaine condition à cause de l'environnement stochastique.

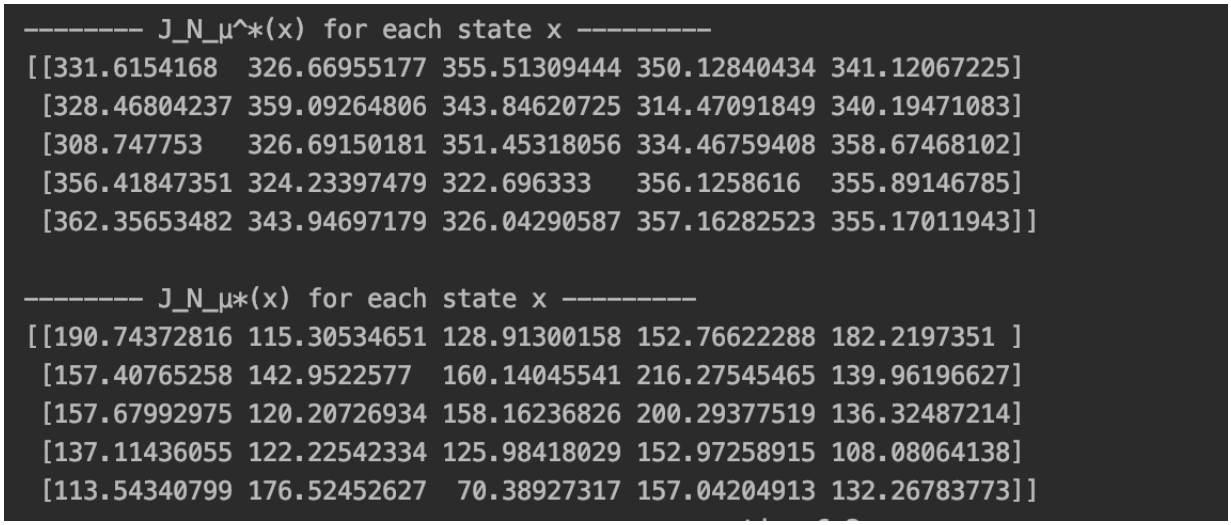

Figure 3: Comparaison des différents valeurs obtenues avec la policy optimal et obtenu avec la policy provenant du Q-learning algorithme.

Dans les partie 6-2, nous avons permit à l'agent de réaliser plus de 100 épisode à 1000 pas. On observe alors que dans ce cas, la Q-table est remplie. Cela lui permet de découvrir plus d'état et ainsi d'améliorer son expérience et de mieux connaître l'environnement. Nous avons vu aussi l'influence du Discount Factor et l'influence du Learning Rate. Le learning rate est le coefficient permettant de représenter le taux d'acceptation de la nouvelle valeur rencontré par rapport à l'ancienne valeur dans la Q-table. Alors que le Discount factor est le coefficient permettant de se concentrer sur les reward actuel au plus proche de l'agent, ou de porter de l'importance aux rewards futurs loin de l'agent. La fonction $offline_{ql}earning$ est la fonction permettant de mettre à jour la q-table à chaque transition.

4 Conclusion

Nous avons été un peu perturbé pour cet assignment car nous avons vu au premier cours de l'année, comment approximer r(x, u) et p(x2 — x1, u). Nous avons donc fait pour la section 4 ce que nous devions faire pour la section 5. Nous avons dû revoir la section 4. Nous avons donc perdu un peu de temps. Le problème pour cet assignment est que l'environnement est stochastique, ainsi les probabilités ne nous donnent pas toujours les mêmes résultats que ce soit pour les valeurs de la value function J ou les valeurs de la Q-table. Ainsi pour

l'exploration et l'exploitation de la Q-table avec la greedy policy, nous avons des résultats qui différent. Nous ne pouvions pas comparer nos résultats avec Q-function de la politique optimale de la section 4 car nous n'avons pas de q-table pour la section 4. Or il suffisait de faire la norme avec *linalg.norm* des Q-table pour obtenir les resultats des convergences. Cependant, la section 4 a été biaisé pour notre cas car le cours était en avance avec la pratique.

Nous avons pu observer le trade-off entre exploration et exploitation du Q-learning Algorithme et la practicité en terme de compléxité avec la Q-table par rapport à la section 5 avec l'approximation des reward et des probabilité de transition.