Apprentissage par Renforcement

M2 DAC

TME 2. Programmation Dynamique

Ce TME a pour objectif d'expérimenter les modèles algorithmes de programmation dynamique sur un MDP classique de type GridWorld.
Nous utiliserons au cours des TME de RL la plateforme en python `gym` (de *open-ai*). Pour l'installer, tapez la commande:

```
pip3 install gym --proxy proxy:3128 --user
```

1 Gym et GridWorld

L'environnement `gridworld-v0` pour `gym` est une tâche de RL où un agent (point bleu) doit récolter des éléments jaunes dans un labyrinthe 2D et terminer sur une case verte, tout en évitant les cases roses (non terminales) et rouges (terminales). Ci-dessous quelques fonctions utiles pour les environnements :

- initialiser un environnement : `env = gym.make('gridworld-v0')` puis `env.reset()`
- `env.action_space` permet de connaître l'ensemble des actions possibles;
- `obs,reward,done,info = env.step(action)` permet de jouer l'action passée en argument : `obs` contient le nouvel état, `reward` la récompense associée à l'action, `done` un booléen pour savoir si le jeu est fini;
- `env.render()` permet de visualiser le jeu;

Pour le cas du gridworld la fonction `env.getMDP()` retourne un couple (states,P), où states est un dictionnaire associant les observations à des numéros d'états (states[ob.dumps()] contient le numéro d'état d'une observation ob) et où P correspond à l'automate du MDP de la tâche: chaque chaque clé est un numéro d'état, chaque valeur un dictionnaire; pour ce 2ème dictionnaire, chaque clé est une action et la valeur une liste de tuples correspondant à la transition associée `(proba de la transition,état`

`destination,reward,done)`, où done est vrai si l'état de destination est un état terminal. Ainsi, `P[state][action]` permet de connaître pour un état et une action la liste des états atteignables avec leur probabilité et la récompense associée.

Dans le répertoire /Vrac/lamprier/FDMS/RL, vous trouverez des ressources utiles pour la réalisation de ce TME. Copiez ce répertoire dans votre espace personnel, vous y mettrez les différents agents que vous aurez à programmer au cours des TMEs de RL. Dans ce répertoire vous trouverez notamment un agent aléatoire randomAgent.py dont vous pourrez vous servir pour tester les environnements et vous donner une idée de la manière de lancer des expérimentations avec gym.

Dans l'environnement gridworld il est possible de charger différentes cartes de problème par la fonction setPlan("plan0.txt",{0:-0.001,3:1,4:1,5:-1,6:-1}) prenant en argument le fichier de la carte à charger et une liste des récompenses associées aux différents types de cases du jeu. 0 correspond à une case vide, 1 correspond à un mur (pas de reward associé car impossible de s'y déplacer), 2 correspond au joueur, 3 correspond à une case verte, 4 une case jaune, 5 une case rouge et 6 une case rose. Dans le répertoire RL, vous pouvez trouver dans envs/gridworldPlans différentes cartes de jeu, de difficulté variable, que vous devrez tester au cours du TME.

2 Travail demandé

Codez les algorithmes de policy iteration et de value iteration vus en cours et testez les sur l'environnement. Discutez des résultats obtenus sur les différentes cartes/configurations (vous pouvez aussi en imaginer d'autres). Vous pouvez aussi faire varier le paramètre de discount pour en observer l'impact.