Reinforcement Learning & Advanced Deep

M2 DAC

TME 4. DQN

Implémentation de l'algorithme DQN avec Target network et Prioritized Experience Replay.

L'implémentation doit être réalisée en PyTorch. Sur le site de l'UE vous trouverez une archive contenant:

- randomAgent.py : une nouvelle version de randomAgent sur laquelle vous vous appuierez pour développer vos algorithmes. Cette version ressemble au fichier QLearning_etudiant.py distribué à la session précédente pour fixer le squelette des algorithmes à développer, mais l'adapte au cadre RL profond que l'on va considérer à partir de maintenant. On note notamment que la fonction storeState a disparu: on travaille avec des états continus qu'on ne peut plus répertorier dans une table comme précédemment. Il s'agit de travailler directement sur les observations fournies, plutôt que de raisonner en terme d'index d'état.

- core.py : un fichier contenant diverses fonctions dont vous pourrez vous servir pour l'extraction de features à partir des observations et la construction de réseaux de neurones via pytorch (vous pourrez par exemple utiliser la classe NN donnée dans ce fichier pour implémenter la fonction Q du DQN).

- repertoire configs : un repertoire contenant des fichiers de configuration yaml. Ceux contenus actuellement correspondent aux fichiers de configuration utiles pour lancer randomAgent sur les 3 environnements que l'on considère dans ce TME (et les suivants).

- memory.py : déclare une structure de données Memory pratique pour constituer le replay buffer. Cette structure permet de stocker les transitions (via la fonction store), d'échantillonner des transitions (via la fonction sample) et de modifier leur valeur de priorité (via la fonction update) de manière efficace. Si l'objet est initialisé avec la valeur prior=True, alors les transitions sont échantillonnées en fonction de leur écart de différence temporelle, accompagnées de leur poids d'Importance Sampling utile pour débiaiser l'algorithme DQN (voir cours 3).

Dans un premier temps, vous testerez randomAgent sur chaque fichier de configuration fourni dans le repertoire configs:

- config_random_gridworld.yaml : configuration pour l'environnement gridWorld considéré dans les TMEs précédents
- config_random_cartpole.yaml : configuration pour l'environnement Cartpole de gym. Cartpole est un jeu où l'on cherche à stabiliser une barre verticale sur un chariot roulant, en fonction de ses mouvements. Deux actions possibles à chaque itération: gauche ou droite. Le jeu est perdu si la barre verticale s'incline de plus de 15 % (elle tombe alors inévitablement) ou si le chariot sort de l'écran. Le reward cumulé correspond au nombre de pas de temps sans terminaison du jeu (maximum 500 pas de temps). Les observations sont des vecteurs de 4 réels donnant la position du chariot, sa vitesse, l'inclinaison de la barre et sa vitesse de rotation. C'est a priori l'environnement le plus simple.
- config_random_lunar.yaml : configuration pour l'environnement LunarLander de gym. LunarLander est un jeu où l'objectif est de faire alunir une fusée, qui a une vitesse et une direction initiales aléatoires. Le reward obtenu dépend de la vitesse à laquelle la fusée s'est posée et de la distance de la cible (ainsi que du temps mis à alunir, du fait que les deux pieds touchent le sol, du fuel dépensé, etc.). 4 actions possibles par pas de temps: ne rien faire, allumer le moteur de gauche, le moteur central ou le moteur de droite. Les observations sont des vecteurs de 8 réels donnant diverses informations sur la position, la direction et la vitesse de la fusée.

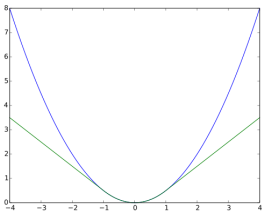
Pour les jeux Cartpole et LunarLander, vous devrez peut-être lancer les commandes suivantes pour installer les librairies requises:

```
pip3 install Box2D --proxy proxy:3128 --user
pip3 install Box2D-kengz --proxy proxy:3128 --user
```

Vous noterez que la classe randomAgent fournie instancie une variable featureExtractor, pas utile pour randomAgent, mais dont vous aurez besoin pour extraire une représentation des observations fournies par l'environnement, en utilisant la fonction getFeatures de la classe correspondante. Selon le type d'extracteur souhaité, cette variable est définie différemment dans le fichier de configuration. Pour Cartpole et LunarLander, c'est une fonction qui ne touche pas au vecteur fourni par l'environnement. Pour GridWorld, le fichier de configuration déclare un transformateur dont la fonction getFeatures prend en argument l'observation de l'environnement et retourne un vecteur de features contenant la position de l'agent, celles des éléments jaunes ainsi que celles des éléments roses (les éléments rouges et verts étant fixes, pas besoin de les représenter dans le vecteur de caractéristiques).

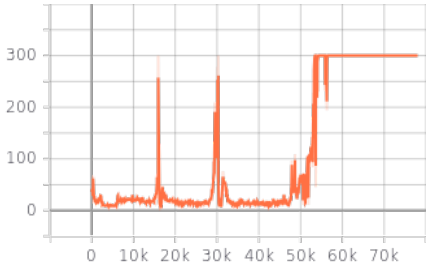
⇒ Produire des courbes d'apprentissage pour les 3 jeux, en comparant des versions avec Experience Replay (Prioritized ou non) et Target Network et des versions sans, selon différents jeux d'hyper-paramètres.

À noter que pour l'apprentissage de la fonction Q, il est souvent préférable d'utiliser un coût Huber (torch.nn.SmoothL1Loss) plutôt qu'un coût moindre carré classique, afin d'éviter l'explosion des gradients:

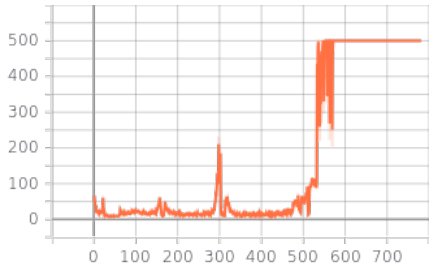


$$L_\delta(a) = \begin{cases} \frac{1}{2}a^2 & \text{for } |a| \leq \delta, \\ \delta(|a| - \frac{1}{2}\delta), & \text{otherwise.} \end{cases}$$

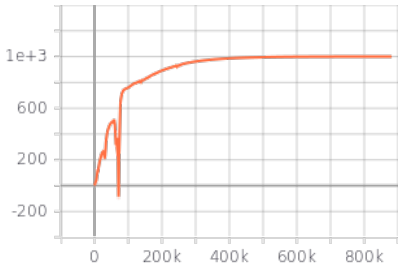
À titre indicatif, voici ce que l'on peut obtenir sur Cartpole, avec DQN utilisant prioritized replay et un target network (mis à jour tous les 1000 évènements), une exploration epsilon-greedy (epsilon initial = 0.1, avec decay de 0.99999 à chaque évènement, un discount de 0.999, un pas d'apprentissage de 0.0003, un batch de 100 transitions, un replay buffer de 10000 transitions, un pas d'optimisation tous les 10 évènements, une taille d'épisode maximale de 300 évènements en apprentissage (500 en test) et selon un réseau de neurones à une couche cachée de 200 neurones (avec activation tanh sur la couche cachée):



(a) Reward en apprentissage



(b) Reward en test (1 test tous les 100 épisodes d'apprentissage)



(c) Valeur Q moyenne (abscisse = nombre d'évènements)

Bonus

- Implémenter la version Double DQN et comparer les résultats
- Implémenter la version Dueling-DQN et comparer les résultats
- Implémenter la version Noisy-DQN et comparer les résultats