TD3 – Apprentissage par renforcement

L'objectif de ce TD est de mettre en oeuvre l'apprentissage par renforcement. Nous allons utiliser le toolkit Gym qui propose plusieurs types d'environnements de simulation. Nous nous intéresserons en particulier à l'environnement Cartpole et à celui du jeu Breakout d'Atari.

1. Découverte de Gym

La documentation de Gym est accessible à l'adresse suivante : https://gym.openai.com/docs/
Après avoir suivi la procédure d'installation, vous pouvez apprendre à le manipuler en testant les exemples décrits dans cette page.

2. Plateformes d'exécution

Pour ce TD, il sera possible d'exécuter votre code sur votre ordinateur portable ainsi que sur les serveurs GPU de l'école dont les informations pour leur utilisation sont données dans le fichier "*PyTorch_serveur_GPU.pdf*". L'avantage de cette dernière solution est que l'apprentissage de vos modèles sera plus rapide du fait de l'utilisation du GPU. L'inconvénient des serveurs GPU est leur faible disponibilité puisque seules 4 cartes graphiques au total (2 serveurs à 2 cartes) sont disponibles pour l'ensemble de trois groupes de TD. Cependant, vous aurez la possibilité de finaliser vos entraînements en dehors de la séance de TD. Une alternative est d'utiliser le serveur Jupyter de l'Ecole (limité à 50 connexions simultanées) à l'adresse suivante : https://jupyter.mi90.ec-lyon.fr (suivre les indications pour installer des dépendances Python qui ne seraient pas présentes).

3. Premier cas d'application : le CartPole

Pour ce premier cas d'application du CartPole (https://gym.openai.com/envs/CartPole-v1/), nous allons développer un modèle d'apprentissage par renforcement de type Q-learning basé sur un réseau de neurones de type perceptron.

L'objectif est d'apprendre à déplacer le chariot afin de maintenir le pendule en équilibre.

Gym sera utilisé pour obtenir les états (position du chariot, vitesse du chariot, angle du pendule, vitesse du pendule au sommet) suite aux actions de l'agent (déplacement du chariot à gauche ou à droite). A noter qu'ici, l'observation n'est pas une image mais directement l'état. L'apprentissage du modèle sera ainsi simplifié et donc plus rapide ce qui vous permettra de tester plus facilement votre programme.

Un code vous est fourni (fichier "*qlearning_cartpole.py*") que vous devez étudier et compléter pour rendre le programme fonctionnel.
Il s'inspire de l'algorithme DQN suivant, en utilisant un perceptron simple plutôt qu'un réseau de neurones convolutif.

Algorithm 1 Deep Q-learning with Experience Replay

Initialize replay memory $\mathcal{D}$ to capacity N
Initialize action-value function Q with random weights
for episode $= 1, M$ **do**
  Initialise sequence $s_1 = \{x_1\}$ and preprocessed sequenced $\phi_1 = \phi(s_1)$
  for $t = 1, T$ **do**
    With probability ϵ select a random action a_t
    otherwise select $a_t = \max_a Q^*(\phi(s_t), a; \theta)$
    Execute action a_t in emulator and observe reward r_t and image x_{t+1}
    Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
    Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in $\mathcal{D}$
    Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from $\mathcal{D}$
    Set $y_j = \begin{cases} r_j & \text{for terminal } \phi_{j+1} \\ r_j + \gamma \max_{a'} Q(\phi_{j+1}, a'; \theta) & \text{for non-terminal } \phi_{j+1} \end{cases}$
    Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ according to equation 3
  end for
end for

4. Deuxième cas d'application : le jeu Breakout

Nous allons maintenant nous intéresser au célèbre jeu Breakout d'Atari (https://gym.openai.com/envs/Breakout-v0/). Cette fois-ci, l'observation consistera en une séquence de 4 images consécutives afin d'avoir l'information de mouvement. Le modèle sera donc de type DQN basé sur un réseau convolutif afin de pouvoir traiter directement les images.

Comme précédemment, un code vous est fourni (fichier "*dqn_breakout.py*") que vous devez compléter en vous inspirant de votre programme précédent.

5. Travail à rendre

Ce travail peut être réalisé par monôme ou par binôme, et un compte-rendu numérique devra être produit, soit sous forme d'un notebook Python, soit sous forme d'une archive « .zip » contenant le code Python et le rapport au format pdf. Le compte-rendu devra contenir notamment les explications concernant le principe des fonctions que vous avez programmées ainsi que leur code commenté. Les expérimentations que vous aurez réalisées devront être décrites et les résultats également commentés. Ce travail devra être déposé sur le site « Moodle» pour le lundi 17 janvier 2022.