Reinforcement learning - TP4

Lilian SCHALL<lilian.schall@epita.fr>, Guillaume LALIRE <guillaume.lalire@epita.fr>

Novembre 2024

Table des matières

1 Introduction

Dans cette étude, nous examinons un algorithme d'apprentissage par renforcement mêlé à du Deep Learning : le Deep-Q Learning. Nous utilisons comme environnement d'évolution le jeu ATARI Pong, afin d'entraîner un agent capable de jouer à Pong à un niveau humain voire surhumain. Nous vous présentons notre architecture de projet ainsi que les résultats d'entraînement. Nous nous basons sur le papier "Playing Atari with Deep Reinforcement Learning" de DeepMind pour l'approche théorique du projet.

2 Choix d'implémentation

Le projet se découpe en trois grands modules : l'environnement, l'estimateur de récompense et l'agent de Deep-Q Learning.

Nous utilisons les bibliothèques Gymnasium de OpenAI pour la gestion de l'environnement et PyTorch pour l'élaboration de l'estimateur de récompense et l'algorithme Deep-Q Learning.

2.1 L'environnement

L'environnement est défini par une classe **PongEnvironment** qui est en vérité un wrapper sur l'environnement de la bibliothèque Gymnasium. L'objectif de ce wrapper est d'inclure les phases de prétraitement sur les images renvoyés par l'environnement de Gymnasium. L'environnement Gym choisi est donc *"ALE/Pong-v5"*. Deux méthodes principales sont définies dans **PongEnvironment** :

- *reset* : L'environnement Gymnasium est réinitialisé et un nouvel état est construit.
- *step* : L'environnement Gymnasium reçoit une action à effectuer et un nouvel état est renvoyé avec la récompense associée.

Sur chaque nouvelle image renvoyée par l'environnement Gymnasium, un prétraitement est appliqué. L'image est redimensionnée à une dimension de 110×84 pixels, suivie d'un recadrage central et d'une binarisation des pixels, ce qui donne une image binaire de 84×84 pixels. Chaque état est produit avec les 4 dernières images de l'environnement.

2.2 L'estimateur de récompense

L'estimateur de récompense est un **Deep-Q Network** tel que spécifié dans le papier de 2015 : c'est un modèle de classification à trois couches de convolution et deux couches denses qui reçoit en entrée un état construit par l'environnement et retourne pour chaque action une estimation de la récompense associée :

- La première couche de convolution applique 32 filtres de taille 8×8 avec un stride de 4.
- La seconde couche de convolution applique 64 filtres de taille 4×4 avec un stride de 2.
- La troisième couche de convolution applique 64 filtres de taille 3×3 avec un stride de 1.
- Deux couches denses respectivement de taille 256 et le nombre d'actions légales sont enfin appliquées.

Entre chaque couche, une fonction d'activation *ReLU* est utilisée.

2.3 L'agent de Deep-Q Learning

L'agent de Deep-Q Learning est représenté sous la forme d'une classe nommée **QLearningAgent**. Cette classe utilise notre estimateur de récompense afin de sélectionner l'action la plus préférable selon une politique ϵ-greedy avec un scheduling de décroissance de ϵ le faisant varier de 1.0 jusqu'à 0.1 durant le premier million de frames traitées.

Deux méthodes principales sont définies. La méthode *forward* permet de prédire l'action à appliquer à l'environnement selon la politique ϵ-greedy. La méthode *backward* permet à l'agent

de faire évoluer l'estimateur de récompense en appliquant une descente de gradient selon une mémoire d'actions passées, tel que présenté dans l'algorithme 1 du papier "Playing Atari with Deep Reinforcement Learning".

2.4 Les Sessions

Enfin, pour entraîner ou utiliser l'agent, il nous faut une session. On en distingue deux :

— La session d'entraînement permet d'entraîner l'estimateur de récompense au travers d'une mémoire de replay, selon une politique ϵ-greedy.

— La session d'utilisation permet de charger un estimateur de récompense pré-entraîné et de faire jouer l'agent à Pong.

3 Résultats

Grace à une implémentation ergonomique et optimisée, notre projet nous permet de traiter 380 états par seconde sur une NVIDIA GeForce RTX 4060 Ti, atteignant ainsi 1000 époques de chacune 1000 steps en moyenne en l'espace de 40 minutes.

Cependant, malgré une expérimentation avec différentes architectures de DQN et une exploration de différentes valeurs pour les hyperparamètres, nous ne sommes pas parvenus à des résultats satisfaisants : à la suite de l'entraînement, l'agent ne parvient pas à remporter la partie, car il reste coincé aux extrémités du terrain.

Pourtant, durant l'entraînement, notamment lorsque ϵ est encore élevé, il réussi parfois à renvoyer la balle et même à marquer des points, mais cela survient uniquement lorsque l'action est choisie aléatoirement à la place du DQN.

La partie enregistrée à la suite de l'entraînement peut être observée sur la vidéo ajoutée au dépôt.

4 Bibliographie

— Papier de recherche associé :
https://arxiv.org/pdf/1312.5602

— Notre dépôt GitHub :
https://github.com/LilianSchall/reinforcement-learning-tp4