Résumé de l'article *Playing Atari with Deep Reinforcement Learning*

> Résumé biblio : PDF 2-3 pages sur l'article qu'on a choisi : dequoi il parle ? Il faut un avis critique --> en quoi c'est pratique, en quoi c'est intéressant, qu'est ce que ça coûte ? difficile à manipuler ? difficile à apprendre ?

Introduction

L'article Playing Atari with Deep Reinforcement Learning traite de l'apprentissage par renforcement sur des jeux Atari. Il a été publié en 2013 par des chercheurs de l'entreprise DeepMind Technologies :

- Volodymyr Mnih
- Koray Kavukcuoglu
- David Silver
- Alex Graves
- Ioannis Antonoglou
- Daan Wierstra
- Martin Riedmiller

Cet article présente donc les résultats d'un apprentissage par renforcement et plus particulièrement d'une variante de Q-Learning sur des jeux Atari 2600. Ces expériences sont basées sur 7 jeux de l'environnement Arcade Learning Environment qui est environnement simple permettant de développer des agents pour de l'intelligence artificielle ou du machine learning sur des jeux de l'Atari 2600.

Le principe de l'apprentissage par renfocement et du Q-learning sont décrits plus détails ici.

Présentation de la méthode

L'objectif est de développer un seul agent capable de jouer à différents jeux Atari. Les entrées du réseau sont les mêmes que celles qu'un joueur humain recevrait, c'est-à-dire l'entrée vidéo, les signaux de récompense et de victoire et l'ensemble des actions possibles. Ainsi, l'architecture du réseau ainsi que les hyper paramètres ne diffèrent pas d'un jeu à un autre.

Le réseau est donc conçu pour être au plus proche de l'être humain.

Deep Reinforcement Learning

Ce projet a donc pour but d'utiliser de l'apprentissage par renforcement. Pour ce faire, les chercheurs ont connecté un algorithme d'apprentissage par renforcement à un réseau neuronal profond qui opère directement sur les images RVB en entrée et qui traite les données d'entraînement par la méthode du gradient stochastique.

La méthode présentée dans cet article est basé sur l'architecture TD-Gammon qui est connue pour être très performante au jeu de Gammon. Cependant, l'approche est légèrement différente. En effet, il est utilisé le principe d'*experience replay* (qui est très utilisé dans le Deep Q-Learning). Ainsi, plutôt que d'utiliser les N dernières actions pour prédire la prochaine action, on stocke les expériences de l'agent dans un ensemble de données (que l'on va noter D) dans lequel on tire au hasard les N échantillons qui vont servir à prendre la décision.

L'agent suit une politique ϵ-greedy, c'est à dire qu'il a une probabilité de $1-\epsilon$ de choisir la meilleure action (celle que maximise le gain) et une probabilité ϵ de choisir une action aléatoire parmis les actions possibles. Une action aléatoire est considérée comme une action *d'explration* tandis qu'une action qui vise à maximiser le gain est une action *d'exploitation*.

Avantages

Cette méthode, selon ses auteurs, présente de nombreux avantages. Elle présente notamment des avantages par rapport aux autres méthodes de Deep Learning, mais aussi par rapport aux approches classiques du Q-Learning.

Avantages vis-à-vis du Deep Learning

Dans un premier temps, la plupart des méthodes de Machine Learning performantes à ce jour ont besoin d'une grande quantité de données étiquetées pour leur apprentissage. Ce n'est pas le cas des algorithmes de renforcement. En effet, ces algorithmes apprennent seulement à partir de leur environnement et des récompenses qui y sont associées.

De plus, de nombreux algorithmes de Deep Learning sont basées sur la supposition que les échantillons sont iid (*independent and identically distributed*). Lors d'un apprentissage par renforcement, il n'est pas rare de rencontrer des séquences de données fortement corrélées, ils le deviennent d'ailleurs de plus en plus, au fil de l'apprentissage de l'algorithme. La distribution n'est pas non plus identique puisque le comportement de l'agent sont aussi amenés à changer vers un comportement "parfait".

Avantages vis-à-vis du Q-Learning

Nous allons présenter ici les avantages d'utiliser le principe d'*experience replay*.

Comme chaque action est définie à partir d'actions précédentes choisies aléatoirement, les données sont potentiellement utilisées plusieurs fois, ce qui augmente donc leur efficacité.

De plus, l'apprentissage à partir d'échantillons consécutifs peut s'avérer inefficace, car ces échantillons sont fortement corrélés. Le fait de sélectionner aléatoirement réduit considérablement cette corrélation.

Enfin, l'état actuel de l'environnement peut être trop influent sur l'agent, pouvant le coincer dans un minimum local non optimal ou bien encore le faire diverger. En utilisant l'*experience replay*, la distribution est moyennée sur plusieurs états précédents et est donc plus "lisse".

Limites

Dans cette partie, nous allons essayer d'avoir un avis critique sur la méthode présentée dans l'article.

Premièrement, il peut être difficile dans certains cas de fixer les gains. Certains jeux sont clairement plus propices à la définition des gains, si on prend l'exemple de *Space Invader*, il est facile de définir un gain positif quand on arrive à toucher un ennemi et un gain négatif quand un ennemi nous touche. Mais cela peut être plus abstrait pour d'autres jeux.

Dans un second temps, ces algorithmes peuvent être longs à apprendre. En effet, puisqu'ils "apprennent de leurs erreurs", il faut leur laisser le temps nécessaire pour commettre les-dites erreurs. De plus, ces algorithmes sont très gourmands en données. C'est pour cela qu'ils sont très utilisés sur les jeux vidéos, car on peut relancer une partie indéfiniment et ainsi avoir un nombre conséquent de données d'apprentissage.

Troisièmement, le choix des hyper paramètres peut être compliqué. En effet, il faut par exemple bien choisir la valeur de ϵ pour la politique ϵ-greedy, sinon l'algorithme va soit trop explorer, soit trop exploiter.

Enfin, étant donné que la distribution des données change au fur et à mesure de l'apprentissage, on ne peut pas envisager de coupler le renforcement par une autre méthode de Deep Learning, car ces méthodes supposent que la distribution des données est fixe.

Résultats des expériences

L'agent développé par les chercheurs de DeepMind a donc été entraîné sur 7 jeux Atari différents : *Beam Rider*, *Breakout*, *Enduro*, *Pong*, *Q*bert*, *Seaquest* et *Space Invaders*. Ses performances ont étées comparées à deux autres algorithmes développés aussi pour les jeux Atari : SARSA et Contengency ainsi qu'avec un joueur humain professionnel. Le tableau suivant résume les différents valeurs de gain moyen total des différents algorithmes et du joueur humain en fonction des 7 jeux :

Player	**B. Rider**	**Breakout**	**Enduro**	**Pong**	**Q*bert**	**Seaquest**	**S. Invaders**
Random	354	1.2	0	−20.4	157	110	179
Sarsa	996	5.2	129	−19	614	665	271
Contingency	1743	6	159	−17	960	723	268
DQN	**4092**	**168**	**470**	**20**	**1952**	**1705**	**581**
Human	7456	31	368	−3	18900	28010	3690

On remarque que la méthode DQN surpasse clairement les deux autres algorithmes sur les 7 jeux. Elle arrive même à surpasser un joueur humain sur 3 de ces jeux.

La méthode décrite dans cet article surpasse ses concurrents en termes de performance sur les 7 jeux Atari sélectionnés, il serait intéressant de voir ses performances sur les autres jeux Atari, voire sur d'autres types de jeux.