CentraleSupélec

Rapport de projet

Ready Player One : Deep Reinforcement Learning

Étude des réseaux neuronaux, puis application au développement d'une intelligence artificielle pour des jeux Atari

2018 – 2019

Nathan Cassereau
Paul Le Grand Des Cloizeaux
Thomas Estiez
Raphaël Bolut

Professeurs encadrants :
Joanna Tomasik
Arpad Rimmel

Établissement :
CentraleSupélec cursus Supélec (promotion 2020)

Table des matières

Table des figures

Introduction

Le projet long Ready Player One a pour but d'étudier le fonctionnement d'algorihtmes de Machine Learning. Cette étude, orientée recherche et dévellopement, cherche à appliquer une branche du machine learning, le Q-Learning, à l'intelligence artificielle d'un jeu video, Pong. Cette IA devra pouvoir jouer au jeu le mieux possible, en se formant au fur et à mesure.

Le projet est mené par deux groupes de quatre élèves, afin de pouvoir comparer les performances des deux produits finaux. Notre groupe, le groupe « Eponge », est composé de Raphaël Bolut, Nathan Cassereau, Thomas Estiez, et Paul Le Grand Des Cloizeaux.

Les 2 groupes sont encadrés par Mme Tomasik et M. Rimmel, qui nous donnent les instructions et les pistes à suivre pour l'aboutissement du projet, et à qui nous rendont des comptes chaque semaines sur le travail réalisé.

L'étude du projet se fait en plusieurs parties : en effet, comme la tâche à réaliser est complexe et dense, et que le projet à pour but de nous faire comprendre les mécanismes du machine learning, nous étudirons plusieurs algorithmes différents au fur et à mesure de l'année, avec lesquels nous expérimenterons :

- dans un premier temps, nous allons étudier le fonctionnement du perceptron, un réseau de neurones basique, que nous allons entrainer à la reconnaissance de chiffres manuscrits de la base de données MNIST de Yann LeCun. Cette première étude a pour but de nous faire comprendre le fonctionnement global du machine learning, et les différents mécanismes d'optimisations utilisés

- Nous allons ensuite rentrer dans le vif du sujet : Le Q-learning, appliqué au jeu vidéo Pong. Nous allons pour cela réaliser une interface permettant à notre algorithme d'intéragir avec le jeu, pour lui permettre d'apprendre. Afin de nous faciliter la tâche, nous utiliserons l'outil TensorFlow (bibliothèque Python), qui permet de faire des calculs de machine learning de façon optimisée

1 Le Perceptron

1.1 Théorie du Perceptron

Le perceptron est un des algorithmes de base du machine learning. Son invention remonte aux années 70, mais a été abandonné alors, son exécution étant trop coûteuse pour les performances des ordinateurs de l'époque. Ce n'est que récemment qu'il a pu resurgir, grâce à l'amélioration des processeurs et des cartes graphiques, particulièrement adaptées aux calculs matriciels. Son principe est simple :

[SCHEMA]

Le perceptron est composé de plusieurs couches, qui réalisent des calculs, que les couches se transmettent successivement. Pour une information X = [X0 ; X1 ; ... ; Xn] en entrée, le perceptron va fournir ce vecteur X à la première couche, qui va calculer le produit matriciel entre X et un matrice W de coefficients appelés « poids ». Le vecteur obtenu est alors transmis à la fonction d'activation, qui applique une fonction non-linéaire à chaque coefficients du vecteur. Le vecteur ainsi obtenu en sortie est fourni à la seconde couche, qui réalise les mêmes opérations, et ainsi de suite.

... // décrire le fonctionnement du perceptron

1.2 Implémentation

Afin de pouvoir comprendre en détail le fonctionnement du perceptron, nous avons commencé dans un premier temps à implémenter un version de celui-çi chacun de notre côté. Cela nous a permis de commencer à réfléchir à l'architecture du code que nous voulions, et de pouvoir comparer les performances des différentes implémentations.

Nous avons testé dans un premier temps les résultats de nos perceptrons sur la fonction XOR. Cette fonction est un bon départ pour pouvoir avoir un code fonctionnel, car il s'agit d'une fonction ne pouvant pas être répliquée par une fonction linéaire : il faut au moins une couche cachée afin de pouvoir l'implémenter grâce à un perceptron. Cette prémière étape nous a permis de comparer les résultats et les performances de nos algorithmes, et de pouvoir choisir l'implémentation du perceptron que nous avons utilisé par la suite.

1.3 Résultats