



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

Introduction	4
1 Le Perceptron	5
1.1 Théorie du Perceptron	5
1.2 Implémentation	5
1.3 Résultats	5

Table des figures

Introduction

Le projet long READY PLAYER ONE a pour but d'étudier le fonctionnement d'algorithmes d'apprentissage automatique. Cette étude, orientée recherche et développement, cherche à appliquer une branche du machine learning, le Q-Learning, à l'intelligence artificielle (IA) du jeu vidéo PONG. Cette IA se formera par elle-même sur ce jeu.

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 deux groupes sont encadrés par Joanna TOMASIK et Arpad RIMMEL, qui nous guident et nous donnent des pistes pour assurer l'avancée du projet, et à qui nous rendent compte chaque semaine du travail réalisé.

L'étude du projet se fait en plusieurs parties. Comme la tâche à réaliser est importante, et que le projet a pour but de nous apprendre les mécanismes du machine learning, nous étudierons plusieurs algorithmes différents au cours de l'année, avec lesquels nous expérimenterons. Les différents codes utilisés lors de ce projet sont consultables sur [Github](#).

Dans un premier temps, nous allons étudier le fonctionnement du perceptron, un réseau de neurones basique, que nous allons entraîner 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.

Puis nous étudierons les réseaux neuronaux à convolution, version améliorée du perceptron. Ces réseaux sont particulièrement adaptés à l'analyse de certaines données comme les images en couleurs.

Nous allons ensuite rentrer dans le vif du sujet : Le Q-learning, appliqué au jeu vidéo PONG. Nous allons pour cela réaliser une interface grâce à laquelle notre algorithme pourra interagir 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 $X = [X_0 ; X_1 ; \dots ; X_n]$ en entrée, le perceptron va fournir ce vecteur X à la première couche, qui va calculer le produit matriciel entre X et une 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 coefficient 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 une version de celui-ci 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 première 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