Premier rapport : projet de recherche

Environnement logiciel pour l'apprentissage de l'exploration visuelle d'une image

Équipe :

Pablo Donato

Louis Cuni

Professeur encadrant :

Olivier Sigaud

Période : février 2017

Environnement logiciel pour l'apprentissage de l'exploration visuelle d'une image

Introduction

Il existe de nombreux procédés pour identifier un chiffre manuscrit. L'intérêt de ce projet est à terme de concevoir un agent qui est capable de mémoriser une représentation partielle d'un chiffre et de l'identifier grâce à une accumulation d'évidence. Ce projet pourra ensuite servir de base pour d'autres travaux de recherches qui exploreraient d'autres méthodes d'identification d'objets dans un espace.

Notre premier objectif est de développer un environnement qui permet l'usage d'un agent pour le parcours d'image.
La finalité de ce projet est de parcourir une image composée de chiffres manuscrits et de pouvoir identifier chaque chiffre, sachant que l'on ne peut voir qu'une fraction de chiffre.

Exemple : l'agent travaille sur ce genre d'image :

Grille de chiffres manuscrits

La seconde partie du projet est plus centrée sur l'intelligence artificielle. Il n'y a pas de contraintes définies si ce n'est que l'on ne peut voir qu'une fraction de l'image. Il n'y a donc pas de contraintes de taux de réussite d'identification ; ni de complexité mémoire ou temporelle. Ceci étant dit, il serait idéal de rester très générique afin de ne pas se cantonner à l'identification des chiffres.

Environnement

La première partie de ce projet consiste donc à développer un environnement permettant l'interaction avec un agent, en l'occurrence représenté par un réseau de neurones. Cet environnement doit comporter plusieurs fonctions :

1. Une fonction pour récupérer les images dans la bibliothèque MNIST sur lesquelles nous allons travailler. La fonction les associe afin de créer une grille sur laquelle sont juxtaposées lesdites images.
2. Une fonction pour retrouver une image sur la grille à partir de coordonnées. Cette fonction nous permet de savoir sur quelle image l'agent travaille.
3. Une fonction qui donne les coordonnées de l'image sur laquelle travaille l'agent afin que l'environnement puisse savoir s'il doit le récompenser ou le punir.
4. Une fonction de récompense.
5. Une méthode step.

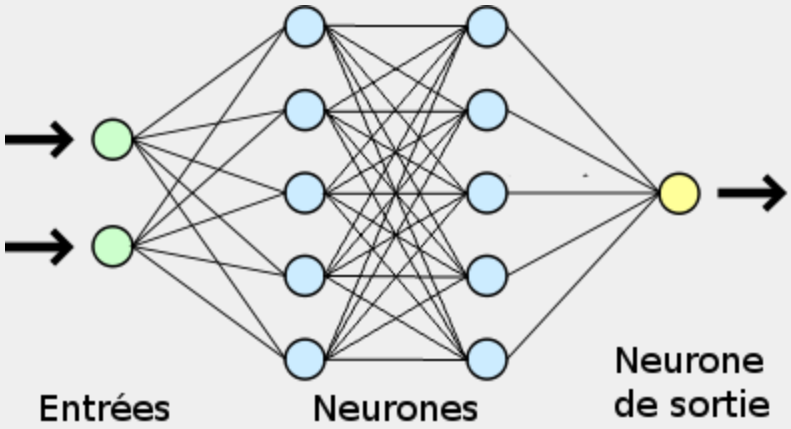
L'environnement est soumis à plusieurs contraintes :

- doit être codé en Python.
- doit être compatible avec l'API imposée par OpenAI Gym.

OpenAI Gym est à la fois une bibliothèque Python offrant un ensemble d'outils facilitant le développement d'environnements pour l'apprentissage par renforcement, et une plateforme communautaire permettant de les partager afin de tester, comparer et améliorer les algorithmes d'apprentissage par renforcement.
Dans un premier temps, l'objectif va donc être de se conformer à l'API proposée par la classe Env d'OpenAI Gym, tout en s'inspirant éventuellement du code source d'autres environnements déjà présents sur la plateforme, afin de faciliter le développement de notre environnement.
Si le code final s'avère suffisamment abouti, on pourra alors procéder à une pull request sur le dépôt GitHub d'OpenAI Gym, qui si elle est acceptée, sera garante de la qualité de notre travail, et permettra au reste de la communauté de proposer des algorithmes pour résoudre notre environnement.

Outils de résolution

Nous utilisons des réseaux de neurones et des algorithmes d'apprentissage par renforcement. On peut résumer un réseau de neurones comme un graphe avec plusieurs couches de sommets, chaque sommet d'une couche étant relié à tous les autres sommets d'une couche adjacente. La particularité d'un neurone est qu'il peut s'activer ou se désactiver selon l'information qui lui est donnée.



Exemple de réseau de neurones. Celui-ci possède quatre couches :
une couche d'entrée, une de sortie et deux couches cachées.

Un réseau doit ensuite s'entraîner. C'est une phase durant laquelle on "apprend" à notre agent à réaliser sa mission. Dans ce projet, on utilise un apprentissage par renforcement.
Le principe de ce dernier est de punir l'agent lorsqu'il se trompe et de le récompenser quand il a raison.
Sachant tout cela, nous allons coder en Python notre agent. Celui-ci utilisera la bibliothèque Tensorflow qui contient des fonctions orientées sur les réseaux de neurones qui seront donc très utiles.

Fonctionnement de l'agent

D'un point de vue algorithmique, il y a deux choses à traiter : le déplacement du curseur et l'identification de ce que l'agent perçoit.

Pour ce qui est du déplacement, nous allons essayer de permettre à l'agent de faire des déplacements à complexité croissante du curseur. Ce qui correspond, dans un premier temps, à déplacer le curseur pixel par pixel dans quatre directions possibles, puis, après un apprentissage, à faire des sauts dans l'image. L'idée étant que l'agent sache déplacer le curseur sans suivre un circuit mais de façon autonome.

Par la suite, nous allons essayer d'entraîner notre réseau en utilisant des auto-encodeurs afin qu'il puisse "mémoriser" les fractions d'image qui déterminent un chiffre. Une fois l'entraînement terminé, le réseau doit pouvoir prévoir ce qu'il verra si on déplace le curseur dans une direction donnée. Ainsi, soit il se trompe et il est puni, soit il a raison et il est récompensé. Ce procédé est appliqué sur 10 réseaux de neurones différents. Chaque réseau détermine l'identification d'un chiffre ; à la fin de l'algorithme, les neurones de sortie des différents réseaux auront des valeurs différentes. Le réseau associé au neurone de sortie ayant la plus grande valeur est considéré comme le plus performant, c'est donc lui qui renvoie le chiffre lu.