**TP 4 : Projet de recherche**

**Application de la neuro-évolution sur des environnements de OpenAi Gym**

**Bernard Meunier, Alexia Reynaud, Olivier Naud-Dulude, Charles-Auguste Marois**

Bernard Meunier : Génie informatique à Polytechnique Montréal, matricule 1878557 bernard.meunier@polymtl.ca

Alexia Reynaud : Génie électrique à Polytechnique Montréal, matricule 12345678 alexia.reynaud@polymtl.ca

Olivier Naud-Dulude : Génie logiciel à Polytechnique Montréal, matricule 1878557 olivier.naud-dulude@polymtl.ca

Charles-Auguste Marois : Génie informatique à Polytechnique Montréal, matricule 1850196 charles.marois@polymtl.ca

**Abstract**

Les réseaux de neurones profonds sont généralement entrainés à partir d’algorithmes d’apprentissage par descente du gradient dont *Q-learning* et Policy Gradient. Toutefois, les stratégies de neuro-évolution basées sur les algorithmes génétiques permettent le même genre d’entraînement (Such et al., 2018). Nous avons donc voulu tester différents environnements pour confirmer qu’il est possible d’utiliser des algorithmes génétiques pour entraîner efficacement des réseaux de neurones profonds.

**1 Introduction**

Lors d’un problème d’apprentissage par renforcement (Reinforcement Learning), le réseau de neurones cherche généralement à maximiser un gain cumulatif (total ou actualisé) sans supervision sur la façon de maximiser ce dit gain. Jusqu’à présent, trois familles d’algorithmes se sont avérées efficaces pour la résolution de problèmes RL, soit les méthodes Q-Learning (DQN) (Mnih et al., 2015), les méthodes de Policy gradient (A3C, TRPO, PPO) (Sehnke et al., 2010) et plus récemment les stratégies neuro-évolutives (Salimans et al., 2017).

Les algorithmes de Q-learning estiment la fonction Q optimale d’un réseau de neurones profond en produisant une politique qui, pour un état quelconque, choisit l’action qui maximise la Q-value. En d’autres mots, le réseau qui joue le rôle de la fonction Q prend à son entrée les états d’un agent et retourne en sortie, via la fonction d’activation softmax, une action discrète.

Les méthodes de Policy gradient apprennent directement les paramètres d'une politique de réseau de neurones profonds qui génère la probabilité de prendre chaque action dans chaque état. Il vise à éviter les intégrales trop laborieuses à calculer en utilisant le gradient d’une politique déterministe au lieu de celui d’une politique stochastique Ainsi, on peut améliorer la performance et l’efficacité dans l’estimation du gradient par rapport au Q-learning.

Enfin, une équipe d’OpenAi a récemment mis au point une version simplifiée de Natural Evolution Strategies qui apprend la moyenne d'une distribution de paramètres, mais non sa variance. Ils ont constaté que cet algorithme est compétitif avec les deux autres familles d’algorithmes sur des problèmes de RL difficiles avec des temps d'entraînement beaucoup plus rapides (lorsque de nombreux processeurs sont disponibles) grâce à une meilleure parallélisation (Salimans et al., 2017).

**2 Les stratégies de neuro-évolution**

L’approche de l’intelligence artificielle du style neuro-évolution est une approche inspirée de l’évolution biologique du système nerveux des êtres vivants. Le but est de faire évoluer un réseau neuronal capable de prendre des décisions selon l’état de la situation dans laquelle il se trouve.

Pour ce faire, il faut avoir un environnement qui est capable d’être observé, soit par des données quantitatives sur différents éléments, soit simplement par des images ou des vidéos. L’environnement doit aussi être capable d’accepter une décision et de déterminer si cette dernière est une bonne décision ou non dans le but d’atteindre un objectif. C’est grâce à ces stimuli qu’il est possible de faire évoluer un réseau neuronal qui permettra d’atteindre l’objectif voulu de la façon la plus efficace possible.

C’est l’aspect d’évolution de la neuro-évolution qui est le plus représentatif de l’évolution biologique. Pour ce faire, on commence par générer une population d’agents capable de prendre des décisions par rapport aux observations sur l’environnement. Cette décision est prise par un réseau neuronal qui est généré de façon aléatoire, donc les décisions prises le sont toutes autant. Cette population d’agents est notre première génération d’agents. On teste alors chacun de ces agents par rapport à notre environnement et on attribue à chacun un score selon sa performance à atteindre l’objectif voulu.

Ensuite, il faut faire évoluer la population d’agents. Pour cela, nous commençons par sélectionner certains de ces agents qui serviront de *parent* à la prochaine génération. Le procédé de cette sélection peut être fait de différentes façons et peut avoir différents impacts sur le processus d’évolution. Plus on sélectionne uniquement les meilleurs agents, plus on risque de perdre de la diversité *génétique* et alors plus on prendre le risque d’atteindre un optimum local qui n’est pas le meilleur. Cependant, plus on garde des agents de façon aléatoire, moins rapidement l’algorithme convergera vers un optimum.

Ensuite il faut générer une nouvelle génération d’agents provenant des parents sélectionnés. Cependant, les nouveaux agents créés doivent être une version évolutive de leurs parents pour pouvoir avoir la chance d’être meilleur. C’est pour cela que la création des agents *enfants* se fait par le croisement de parents. Ce croisement peut être fait de plusieurs façons, le but est de construite un agent avec son réseau neuronal fait à partir de celui des parents. Par exemple, on peut sélectionner au hasard la provenance de chacun des poids du nouveau réseau pour en faire un nouveau unique. Par la suite, pour s’assurer de faire apparaitre du matériel génétique nouveau dans la banque de matériel génétique que nous avons, il faut appliquer des mutations sur la nouvelle génération. Tous les nouveaux agents créés par croisement sont passés dans un algorithme de mutation. Par exemple, on peut itérer sur chacun des poids du réseau et selon une probabilité, le remplacer par un nouveau généré aléatoirement.

Une fois cette nouvelle génération créée, on recommence le processus de test sur chacun des agents et ensuite la sélection des *parents* en la création d’une nouvelle génération. Ce cycle peut être réalisé un nombre prédéterminé de fois ou arrêté plus rapidement si certains critères de performance sont atteints. On garde alors le meilleur agent généré et on procède à une évaluation complète de ses performances pour confirmer ses capacités.

**3 Expérimentation**

Pour l’expérimentation de notre projet, nous avons décidé de créer un algorithme de neuro-évolution pour résoudre les différents environnements fournis par la librairie *Gym*[[1]](#footnote-1)de la compagnie *OpenAI*. Cette librairie est une boite à outils permettant de développer différents algorithmes d’apprentissage par renforcement. Elle comprend une collection d’environnements de tests ayant tous une interface d’utilisation semblable pour expérimenter avec différents algorithmes. Le but de l’expérimentation est de développer l’algorithme générique de neuro-évolution le plus efficace possible pour résoudre l’environnement appelé « CartPole-v1 ». Ensuite nous appliquons ce même algorithme à d’autres environnements avec comme seule modification les hyperparamètres de l’algorithme.

**3.1 Construction de l’algorithme.**

**4. Lien vers notre GitHub**

https://github.com/beurnii/INF8225

**5. Références**

Mnih V., Kavukcuoglu K., Silver D., Rusu A. A., Veness J., Bellemare M. G., Graves A., Riedmiller M., Fidjeland A. K., Ostrovski G., and others . Human-level control through deep reinforcement learning. Nature, 2015.

Salimans T., Ho J., Chen X., and Sutskever I. Evolution strategies as a scalable alternative to reinforcement learning. arXiv preprint arXiv:1703.03864, 2017.

Sehnke F., Osendorfer C., R¨uckstieß T., Graves A., Peters J., and Schmidhuber J. Parameter-exploring policy gradients. Neural Networks, 2010.

Such, Madavan, Conti, Lehman, Stanley and Clune. Genetic Algorithms Are a Competitive Alternative for Training Deep Neural Networks for Reinformcement Learning. Uber AI Lab. 2018.

<http://www.scholarpedia.org/article/Neuroevolution>

1. <https://gym.openai.com/> [↑](#footnote-ref-1)