Étape 2: Robot-soccer sur Unity et *Proximal Policy Optimization*

Léo Chartrand 20 119 818
Jeremi Levesque 20 063 066
Marc-Antoine Bélisle 20 042 188

IFT608 - Planification en intelligence artificielle
Département d'informatique
Université de Sherbrooke

Abstract

Nous approfondissons et documentons dans ce rapport notre compréhension théorique et applicative de l'algorithme *Proximal Policy Optimization* ainsi que les algorithmes «intermédiaires» qui ont permis d'arriver à celui-ci. Nous illustrons le gain de performance de l'apprentissage de PPO par rapport à *Advantageous Actor-Critic* ainsi qu'à DQN à l'aide de l'environnement multi-agents *SoccerTwos* de ML Agents Unity.

1 Introduction

Dans l'étape précédente de ce projet, nous avons exploré la possibilité d'implanter de l'apprentissage par renforcement hiérarchique grâce à l'article de référence de Cao & Lin [1] sur un environnement de robot-soccer du *ML Agents Toolkit*. Ce simulateur offre un environnement de simulation idéal pour le développement d'algorithmes dans un scénario adversarial et coopératif. Cependant, par les résultats inconclusifs de cette expérience, nous avons choisi d'aller dans une direction légèrement différente. En considérant certaines recherches qui proposent des algorithmes pour des environnements multi-agents et coopératifs [1][2], on peut voir que l'algorithme de *Proximal Policy Optimization* (PPO) proposé par *DeepMind* [3] est fréquemment utilisé comme mesure de performance comparative aux algorithmes proposés. Puisque PPO est un des algorithmes de pointe depuis plusieurs années, nous avons estimé qu'il serait pertinent d'approfondir notre compréhension de cet algorithme afin de pouvoir évaluer le progrès que cet algorithme apporte en général par rapport aux méthodes *Actor-Critic* de base. Ainsi, nous allons aussi comparer les performances des apprentissages de *DQN* et *A2C*. De plus, en bonus, nous avons effectué des tests afin de mieux comprendre le comportement des algorithmes *MA-POCA* et *PPO*, qui sont des algorithmes qui ont été comparés dans l'étape précédente, sur des environnements avec plus de joueurs.

2 Travaux Reliés

2.1 Méthodes de fonction de valeur

Les méthodes d'apprentissage par différence temporelle (*e.g.* DQN [4]) permettent d'approximer l'utilitée d'une paire état-action afin de pouvoir en dériver la politique à suivre. Cependant ces méthodes évaluent un ensemble discret d'états sur lequel une action résultant en une valeur maximale est sélectionnée, introduisant la possibilité d'une convergence prématurée vers un minimum local. Un compromis à ce dilemme d'exploitation et exploration consiste à utiliser une politique ϵ-greedy, où l'hyperparamètre $\epsilon \in [0, 1]$ définit la probabilité de choisir une action aléatoire.

2.2 Policy Gradients et Actor-Critic

Une alternative aux méthodes action-valeur est la famille des méthodes *Policy Gradient*. Ces méthodes apprennent directement une politique au lieu d'apprendre des valeurs d'état-action. La politique est donc une distribution de probabilités des actions, paramétrée par θ, sur laquelle une ascente de gradient est approximée dans le but de maximiser la performance[5]. Les méthodes *Actor-Critic* sont des Policy Gradients qui réintroduisent une fonction de valeur paramétrée, réduisant la variance durant l'apprentissage. Avec cette architecture formée de deux composantes, la politique (*Actor*) peut mettre à jour ses paramètres θ en suivant la direction de la fonction de valeur (*Critic*) qui elle met aussi apprends ses paramètres w en parallèle grâce à l'équation suivante:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta} \left[\nabla_\theta \log \pi_\theta(s, a) V^w(s) \right].$$

(Mnih *et al.*, 2016)[6] propose une dérivation de cette méthode nommée *Asynchronous Advatange Actor-Critic* (A3C) qui utilise une ascente de gradient asynchrone pour l'apprentissage parallèle. Cette technique apporte une performance accrue, faisant entre autres l'usage de l'estimation généralisée des avantages (GAE) proposés (Schulman *et al.*, 2016)[7] ainsi que de l'entropie de la politique π, en ajoutant cette dernière à la fonction objectif afin d'encourager l'exploration en freinant une convergence précipitée vers un maximum local. Une version synchrone de cette méthode (A2C) fait d'ailleurs un meilleur usage des GPUs[8].

2.3 TRPO

En 2015, des chercheurs de l'Université de Berkeley ont proposé une nouvelle méthode de policy iteration, nommée *Trust Region Policy Optimization*[9]. L'algorithme est prouvé de garantir des retours non décroissants, faisant usage d'une pénalité sur la divergence de Kullback–Leibler des politiques π_θ et $\pi_{\theta_{\text{old}}}$. Le résultat est une fonction objectif subrogée semblable à celle des Policy Gradients, où un ratio de probabilités des politiques est guidé par une fonction avantage, respectant une contrainte suivant la divergence KL, c'est à dire

$$\begin{aligned} \underset{\theta}{\text{maximiser}} \quad & \hat{\mathbb{E}}_t \left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)} \hat{A}_t \right] \\ \text{sujet à} \quad & \hat{\mathbb{E}}_t \left[\text{KL} \left[\pi_{\theta_{\text{old}}}(\cdot|s_t), \pi_\theta(\cdot|s_t) \right] \right] \leq \delta, \end{aligned}$$

où l'espérance correspond à la moyenne des échantillons Monte Carlo. Pour résoudre le problème d'optimisation résultant de ces formules, au lieu d'utiliser une matrice de covariance sur les probabilités des politiques, la hessienne de la divergence KL est calculée afin d'obtenir une matrice d'information Fisher [1], ce qui demande donc le calcul de gradients du deuxième ordre. L'algorithme proposé en pratique est très performant et efficace dans son usage d'échantillons, permettant de stabiliser l'apprentissage en empêchant la surestimation des retours. Quoique les résultats offerts par TRPO sont prometteurs, la complexité de la méthode, demandant d'ailleurs une optimisation du deuxième ordre, peut dissuader de son utilisation.

2.4 PPO

La force de TRPO réside dans sa tendance à former une limite pessimiste sur son évaluation des retours à l'aide d'une contrainte. La motivation pour améliorer cet algorithme serait d'émuler l'amélioration monotone de la performance de la politique à l'aide d'un algorithme simplifié, se limitant à une optimisation du premier ordre. La méthode de *Proximal Policy Optimization* proposée par (Schulman *et al.*, 2017) [3] permet d'offrir des performances comparables et parfois même supérieures à TRPO. Afin de faciliter la manipulation des équations suivantes, on dénote le ratio des probabilités entre la politique actuelle et l'ancienne polique $r_t(\theta)$. Ainsi, à partir de la fonction objectif définit ci-haut par TRPO, Schulman *et al.* proposent une nouvelle fonction objectif qui permet de limiter l'envergure d'une mise à jour de façon similaire à TRPO:

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t \left[\min \left(r_t(\theta) \hat{A}_t, \text{clip}(r_t(\theta), 1 - \epsilon, 1 + \epsilon) \hat{A}_t \right) \right] \quad \text{où} \quad r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$$

[1] L'usage d'une matrice d'information de Fisher évoque une certaine similarité avec les *Natural Policy Gradients* [10].

Ici, nous cherchons toujours à maximiser $r_t(\theta)\hat{A}_t$. La fonction de $clip$, qui est la composante fondamentale de PPO, permet de limiter $r_t(\theta)$ à une valeur comprise dans l'intervalle $[1-\epsilon, 1+\epsilon]$, favorisant des mises à jour pessimistes respectant les contraintes suivantes:

$$\begin{aligned}
(1-\epsilon < r_t(\theta) < 1+\epsilon) &\implies L^{CLIP}(\theta) = r_t(\theta)\hat{A}_t \\
(r_t(\theta) > 1+\epsilon) \wedge (\hat{A}_t \geq 0) &\implies L^{CLIP}(\theta) = \boldsymbol{(1+\epsilon)\hat{A}_t} \\
(r_t(\theta) < 1-\epsilon) \wedge (\hat{A}_t \geq 0) &\implies L^{CLIP}(\theta) = r_t(\theta)\hat{A}_t \\
(r_t(\theta) > 1+\epsilon) \wedge (\hat{A}_t < 0) &\implies L^{CLIP}(\theta) = r_t(\theta)\hat{A}_t \\
(r_t(\theta) < 1-\epsilon) \wedge (\hat{A}_t < 0) &\implies L^{CLIP}(\theta) = \boldsymbol{(1-\epsilon)\hat{A}_t}
\end{aligned}$$

En général, ϵ est un hyperparamètre ayant une valeur comprise dans l'intervalle $[0, 1]$ et qui démontre généralement une bonne performance lorsque $\epsilon = 0.1$ ou $\epsilon = 0.2$. L'intervale $[1-\epsilon, 1+\epsilon]$ forme un intervalle de confiance pessimiste par rapport à la mise à jour réelle, permettant d'ajuster les paramètres raisonnablement afin de converger plus rapidement vers une meilleure solution. Par ailleurs, étant donné un modèle de forme Actor-Critic où une politique et une fonction état-valeur sont apprises, il est possible que ces architectures partagent des paramètres. Dans un tel cas, les auteurs suggèrent de calculer également un terme d'erreur sur la fonction de valeur. Un terme d'entropie S est ensuite ajouté pour garantir l'exploration, pour finalement obtenir

$$L_t^{CLIP+VF+S}(\theta) = \hat{\mathbb{E}}_t \left[L_t^{CLIP}(\theta) - c_1 L_t^{VF}(\theta) + c_2 S\left[\pi_\theta\right](s_t) \right],$$

où c_1 et c_2 sont des constantes et L_t^{VF} est une perte de moindres carrés sur $V(s)$. L'algorithme est ainsi exécuté pour T transitions, en utilisant $\pi_{\theta_{\text{old}}}$ pour générer les avantages $\hat{A}_t$ obtenues par l'estimation GAE. La fonction objective subrogée est ensuite optimisée afin de mettre à jour les paramètres θ.

3 Méthodologie

L'évaluation des algorithmes se fait dans le même environnement qu'à l'étape 1, i.e. l'environnement de *SoccerTwos* offert par ML-Agents dans la *Release 20*. De plus, nous fixons le $seed = 5$ pour l'apprentissage de tous les modèles afin d'avoir des résultats qui ne dépendent pas de l'aléatoire et qui sont ainsi comparables avec PPO qui a été entraîné lors de l'étape précédente avec le même seed. Les courbes des graphiques utilisés pour la comparaison des algorithmes sont toutes générées grâce au module Tensorboard qui permet de comparer, visualiser et d'exporter les données des apprentissages. Les données des courbes d'apprentissages ainsi que les configurations des agents sont disponibles sur le GitHub du projet.

3.1 Méthodes de référence

Afin d'évaluer la performance de PPO, nous comparons cette dernière avec 2 autres algorithmes: DQN et A2C. L'évaluation de ces méthodes permettra de comparer la performance entre les méthodes PPO et les approches par différence temporelle ainsi que les policy gradients. ML Agents avait des implémentations de DQN et de A2C qui étaient implémentées plus spécifiquement pour d'autres environnements non adversarials et non multi-agent. Dans notre contexte d'utilisation de l'environnement de simulation, l'entraînement de tous les agents pour l'environnement de *SoccerTwos* a été fait en *self-play*. Puisque *SoccerTwos* est composé de deux équipes, le *self-play* permet d'assigner les politiques d'apprentissage aux membres d'une première équipe et c'est l'équipe qui améliorera sa politique. L'autre équipe, elle, est assignée la politique initiale de l'équipe adverse, mais on la fixe afin que cette équipe n'apprenne pas durant l'épisode. De cette façon, les agents peuvent apprendre en jouant contre une équipe initialement identique à eux. Ça permet d'avoir une mesure de performance relative intéressante qui est détaillée dans la sous-section suivante: l'Elo.

Afin de pouvoir utiliser A2C dans un contexte de *self-play*, nous avons ajouté quelques fonctions qui étaient nécessaires afin de supporter le *self-play*. L'algorithme DQN fourni par ML-agents étant

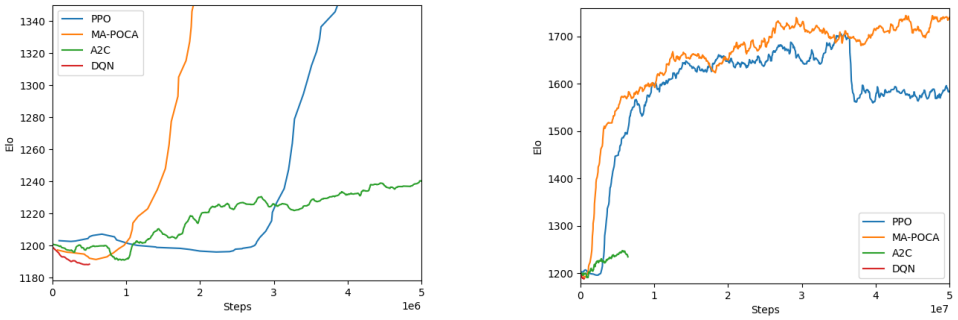
implémenté pour des tâches simples à un seul degré de liberté, une restructuration de l'algorithme a été exécutée pour permettre l'approximation de Q-valeurs pour plusieurs branches d'Actions par état. De plus, l'environnement *SoccerTwos* était spécialement conçu pour l'utilisation de récompenses de groupes et ce genre de récompenses n'est pas supporté par les algorithmes PPO, A2C, et DQN. Donc, afin d'assurer une uniformité des récompenses reçues entre les différents types d'algorithmes, nous avons introduit des récompenses individuelles identiques à celles de groupe lorsqu'un but est marqué. Donc, si chaque membre de l'équipe marquante reçoivent $1 - (nombredeticks/max_steps)$ en récompenses alors que les agents de l'autre équipe reçoivent une récompense de -1.

3.2 Elo

Nous utilisons le système de classement *Elo* afin de mesurer la performance relative des agents. Cette méthode est couramment utilisée pour des jeux antagonistes à somme nulle, représentant la performance des joueurs comme une variable suivant la loi normale[11]. Nous estimons que ce critère d'évaluation est idéal pour nos expérimentations puisqu'il permet d'éviter le biais introduit par les récompenses cumulatives lorsque plus de deux entités sont évaluées dans des jeux à somme nulle. Dans un tel cas, il est impossible de modéliser les récompenses pour émuler la mesure offerte par Elo.

4 Expérimentations

Afin d'offrir de mettre en perspective les performances des agents entraînés avec A2C, DQN et PPO, nous avons aussi inclus les résultats obtenus par l'algorithme de référence, MA-POCA, pour cet environnement qui a été entraîné lors de l'étape précédente. Cela permet de faire ressortir l'échelle de la différence de performance entre les agents.



(a) Échelles restreintes (b) Entraînement complets

Figure 1: Perspectives de l'apprentissage des algorithmes de comparaison.

Pour rappel, il est important de noter que PPO et MA-POCA sont les deux algorithmes de référence pour l'environnement de simulation ce qui explique une convergence beaucoup plus rapide et performante qu'A2C et DQN selon la figure 1a puisqu'ils sont optimisés et spécialement implémentés pour des scénarios multi-agents. Comme nous pouvons le remarquer dans les figures 1a et 1, l'entraînement de A2C et DQN a été beaucoup plus bref en termes d'itérations, mais puisque ces algorithmes sont moins adaptés aux scénarios multi-agents, l'entraînement était beaucoup plus long notamment lors de l'apprentissage des modèles. En guise de comparaison, pour MA-POCA et PPO, nous obtenons un temps d'entraînement pour les 500k premières itérations de 11 et 7 minutes respectivement alors que nous obtenons 45 minutes et 120 minutes pour un même nombre d'itérations. Ainsi, par souci du temps, nous avons limité DQN à 500k itérations afin d'offrir un nombre d'itérations intéressantes. Bien que DQN et A2C aient moins d'étapes d'entraînement, nous estimons que l'elo conservera une tendance similaire; A2C restera bien inférieur à PPO et DQN sera inférieur à A2C. A2C atteindra son apogée avec un elo 1248 alors que DQN ne fait que décroître jusqu'à un elo 1189 qui sont des valeurs bien en dessous de la valeur de convergence moyenne de PPO qui est environ à 1720 elo et celle de MA-POCA qui atteint un Elo de 1752.

4.1 Modification de l'environnement

Dans cette section, nous avons repris les algorithmes, PPO et MA-POCA, entrainés précédemment et avons cherché à observer leurs performances dans des environnements plus grands et avec plus d'agents. Nous avons adapté l'environnement SoccerTwos 2v2 pour créer des versions en 4v4 et 8v8. L'environnement de 4v4 est 20% plus grands que celui de 2v2, tandis que l'environnement de 8v8 est 50% plus grands que celui de 2v2. Ces modifications de taille ont pour conséquence d'influencer le comportement des agents. Chaque agent dispose de 11 *ray-cast* vers l'avant couvrant un angle de 120 degrés et de 3 *ray-cast* vers l'arrière couvrant un angle de 90 degrés, pour une distance d'environ 20 unités, soit la moitié d'un terrain de 2v2. Dans un environnement plus grand, ces *ray-cast* deviennent moins efficaces pour couvrir l'ensemble du terrain, ce qui affecte les stratégies développées par les agents. Dans l'environnement 2v2, les agents PPO et MA-POCA ont développé une stratégie avec des rôles de "striker" et "goalie", où un joueur se positionne dans le but pour le défendre et l'autre se concentre sur l'attaque. Cependant, dans les environnements 4v4 et 8v8, cette stratégie évolue différemment. Dans ces environnements plus grands, un agent adopte généralement un comportement plus offensif, tandis que la pluspart des autres agents restent à proximité du but pour assurer sa protection.

5 Conclusion

Pour conclure, l'efficacité et la performance de PPO pour l'entraînement d'agents sont dûment mises en avant par l'entraînement de A2C et DQN dans un même environnement notamment par la différence significative du Elo ainsi que par le temps relatif d'entraînement des agents. DQN et A2C pourraient bénéficier d'une amélioration de la gestion de la mémoire et de l'apprentissage des agents pour offrir un temps d'entraînement minimal et ainsi pouvoir les entraîner pour avoir une comparaison juste de tous les algorithmes, même si la tendance suggère que PPO demeure supérieur à DQN et A2C dans plusieurs domaines.

References

[1] Zehong Cao and Chin-Teng Lin. "Reinforcement learning from hierarchical critics". In: *IEEE Transactions on Neural Networks and Learning Systems* (2021).

[2] Arthur Juliani et al. "Unity: A general platform for intelligent agents". In: *arXiv preprint arXiv:1809.02627* (2020).

[3] John Schulman et al. "Proximal policy optimization algorithms". In: *arXiv preprint arXiv:1707.06347* (2017).

[4] Volodymyr Mnih et al. "Human-level control through deep reinforcement learning". In: *Nature* 518 (2015), pp. 529–533.

[5] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[6] Volodymyr Mnih et al. *Asynchronous Methods for Deep Reinforcement Learning*. 2016. arXiv: 1602.01783 [cs.LG].

[7] John Schulman et al. *High-Dimensional Continuous Control Using Generalized Advantage Estimation*. 2018. arXiv: 1506.02438 [cs.LG].

[8] *OpenAI OpenAI Baselines: ACKTR & A2C*. https://openai.com/research/openai-baselines-acktr-a2c. Accessed: 2023-03-26.

[9] John Schulman et al. *Trust Region Policy Optimization*. 2017. arXiv: 1502.05477 [cs.LG].

[10] Sham M Kakade. "A Natural Policy Gradient". In: *Advances in Neural Information Processing Systems*. Ed. by T. Dietterich, S. Becker, and Z. Ghahramani. Vol. 14. MIT Press, 2001. URL: https://proceedings.neurips.cc/paper_files/paper/2001/file/4b86abe48d358ecf194c56c69108433e-Paper.pdf.

[11] *HuggingFace Self-Play: a classic technique to train competitive agents in adversarial games*. https://huggingface.co/deep-rl-course/unit7/self-play?fw=pt. Accessed: 2023-03-26.