Démarche

Mehdi Mounsif

21 février 2018

En observant les tentatives de l'agent, je remarque qu'après un certain nombre d'itérations, une action semble dominer complètement toutes les autres. Par conséquent :

- Il faudrait observer l'évolution des poids → J'ai créé FastPlot, une librairie python pour l'observation des poids et du processus de décision. Disponible sur GitHub
- J'ai rencontré la notion d'entropie.

En faisant des recherches sur l'entropie, j'ai trouvé des implémentations [2] qui utilisent l'entropie comme régularisation. On a :

$$H = \frac{\log(\pi(\star|s_t))}{\pi(\star|s_t)}$$

Si l'entropie est basse, c'est qu'une action domine les autres, ce qui implique, entre autre, une supression de l'exploration. Afin de prévenir cet état, l'entropie est considérée dans la fonction de coût de la manière suivante :

$$L = L_v + L_\pi - \alpha E$$

Où : L est le coût global, composé de L_v, le coût de la fonction valeur (critique), L_π, coût de la politique (acteur) et E, l'entropie, dont l'importance est modulée par α.

L'entropie est une notion très utilisée dans le domaine de **Inverse Reinforcement Learning**. A investiguer. J'ai également découvert : SAC (Soft Actor Critic). [1], apparement plus stable que les approches classiques. Les auteurs utilisent trois approximations de fonctions :

- Value function $V_\psi(s_t)$
- Soft Q-function $Q_\theta(a_t|s_t)$
- Policy π_ϕ

Le papier est plutôt clair concernant les mises à jour de $V_\psi(s_t)$ et $Q_\theta(a_t|s_t)$. En revanche, l'amélioration de la politique est similaire à PPO [4], dont je n'ai pas encore compris comment établir le ratio.

Inspiré par [3], j'ai écrit une première implémentation. Une classe ES reçoit en argument la taille de l'espace d'observation et de l'espace d'action, ainsi que les paramètres de la fonctions d'approximation (taille des couches cachées + biais). Un individu base est généré suivant une distribution normale $\mathcal{N}(0, 0.1)$. En fonction des paramètres de bruits donnés par l'utilisateur, on crée toute une population. Après évaluation, on déplace l'individu base suivant les paramètres des individus pondérés par leurs réussite.

Sur un test trivial (déplacement d'une population sur un plan), l'algorithme se comporte parfaitement. Sur CartPole, certains individus arrivent à la récompense maximale et un calibrage adapté des hyperparamètres permet de maintenir la performance. Sur le Reacher, échec. Penser à utiliser CMA-ES décrite dans [3] et apparement très utilisée. Egalement, l'article insiste sur la normalisation des *virtual batchs* qui serait **fondamentale**.

Algorithm 1 ES 1

Require: Paramètres θ de la fonction d'approximation, nombre d'individus N, Environnement
Création d'une population à partir d'un individu non perturbé Ω. Enregistrer paramètres de bruit μ des individus
Evaluation des individus dans l'environnement. Enregistrer récompense
Evaluer le gradient : $\nabla\theta = \frac{1}{N}\sum_{i=1}^{N} \mu_i R_i$
Mise à jour de l'individu non perturbé Ω : $\theta \leftarrow \theta + \alpha\nabla\theta$

Références

[1] Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. Soft Actor-Critic : Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. *ArXiv e-prints* (January 2018).

[2] Kostrikov, I. Pytorch implementations of reinforcement learning algorithms. https ://github.com/ikostrikov/pytorch-a2c-ppo-acktr, 2018.

[3] Salimans, T., Ho, J., Chen, X., Sidor, S., and Sutskever, I. Evolution Strategies as a Scalable Alternative to Reinforcement Learning. *ArXiv e-prints* (March 2017).

[4] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *CoRR* (2017).