Démarche

Mehdi Mounsif

27 février 2018

1 Positionnement des algorithmes RL

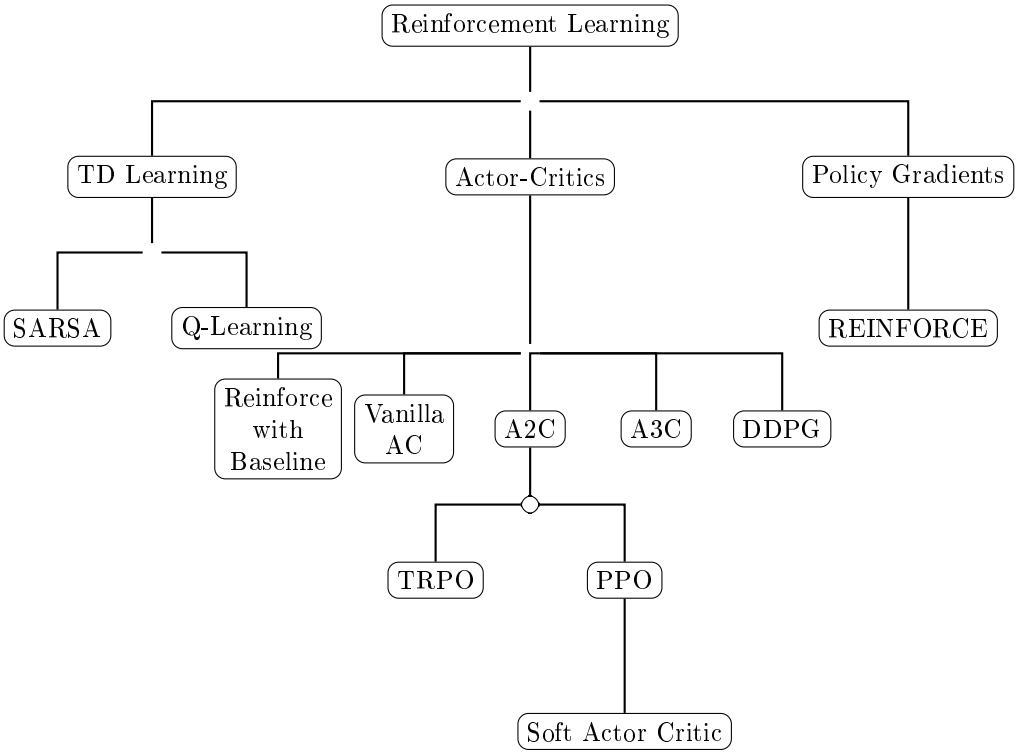

FIGURE 1 – Cartographie des algorithmes RL

2 Utilisation de PPO dans le cadre des notice networks

Proximal Policy Optimization [6] doit, si l'on en croit ses créateurs grandement solutionner le problème des chutes de performances. En effet, PPO découle de TRPO [5] dont le principe est de s'assurer que les mises à jour n'emmènent pas fonction hors de son approximation locale, dites *trust region*. Partant de cette hypothèse, je vais tester incessament sous peu l'algorithme dans Reacher. En cas d'échec :

- Il est possible que j'ai raté l'implémentation (mais j'en doute)
- Vérifier les hyperparamètres. OpenAI préconise $\epsilon = 0.2, \gamma = 0.99, \lambda = 0.95, Adam = 3.10^{-4}$
- La représentation de l'état est-elle à mettre en cause ?

C'est cette dernière question qui soulève la nécessité de commencer à considérer des représentations visuelles. En effet, comment représenter fidèlement le point de prise d'un objet avec des primitives bas-niveau ?

3 Idées RL

Her : Hindsight Experience Replay [1]

4 GANs - Generative Adversarial Networks

4.1 Intérets

D'après [4], les GANs sont une catégorie de modèle générateurs, qui, à partir d'un training set représentant une distribution p_{data} détermine une approximation de cette probabilité de distribution p_{model}. On peut alors soit explicitement représenter cette distribution (low-dimensionality) où plus généralement, en extraire des échantillons (génération d'images). Il est possible (dans notre cas précis) d'appliquer ces modèles générateurs à l'apprentissage par renforcement, plus précisément au **model-based RL**. Utilisation en plannification, et compatibles avec IRL [3], [2]

4.2 Fonctionnement

On se focalise sur les GANs à base de **maximum likelihood**. Tous les modèles ne fonctionnement cependant pas de cette manière. L'idée directrice est de définir un modèle qui retourne une estimation de la distribution de probabilités, paramétrisée par θ. On appelle alors la vraisemblance la probabilité que le modèle attribue au jeu de données d'entrainement contenant m exemples $x^{(i)}$:

$$\prod_{i=1}^{m} p_{model}(x^{(i)}|\theta)$$

Le principe de **maximum likelihood** préconise de choisir les paramètres θ qui maximisent la probabilité du training set. Ceci est plus aisé dans un espace logarithmique afin de considérer une somme plutôt qu'un produit :

$$\theta^* = \text{argmax}_\theta \prod_{i=1}^{m} p_{model}(x^{(i)}|\theta)$$

$$\theta^* = \text{argmax}_\theta \log \prod_{i=1}^{m} p_{model}(x^{(i)}|\theta)$$

$$\theta^* = \text{argmax}_\theta \sum_{i=1}^{m} \log p_{model}(x^{(i)}|\theta)$$

Comme le log est une fonction est monotone $\text{argmax}_v f(v) = \text{argmax}_v \log f(v)$. On note également que l'estimation du **maximum likelihood** revient à minimiser la divergence KL entre le générateur et la vraie distribution.

4.3 Taxonomie des modèles

- Explicit density : Ce sont les modèles qui définissent explicitement une densité de probabilité $p(x|\theta)$. Tout le challenge consiste alors à bien approcher la distribution. Pour cela deux options : construction minutieuse du modèle ou approximations.
  - — Tractable density (calcul explicit de la densité de probas)
    - — Non linear independant component analysis
    - — Fully visible belief nets : Parmi le top (WaveNet par exemple), mais très lent.
  - — Approximate density (densité approximée)
    - — Variational (autoencoder) = déterministe.
    - — Markov Chain (Boltzmann Machine) = stockastique.
- Implicit density : L'analyse d'échantillons issus de p_{model} permet de cerner progressivement la distribution p_{data}
  - — Direct (GAN)
  - — Markov Chain (GSN)

Globalement, les GANs ont été conçus pour palier aux problèmes des autres types de modèles générateurs (temps de génération, peu de restrictions, pas de chaines de Markov, pas de borne variationnelle). Voir [4]

4.4 Framework

Il s'agit grossièrement d'un jeu entre deux fonctions : une fonction discriminatrice et une génératrice. La génératrice crée des échantillons issus d'une distribution théoriquement proche de la vraie distribution. La discriminatrice examine ces échantillons et décide s'ils sont des *vrais* échantillons où des échantillons générés (classification suivant deux classes). La génératrice à pour but de tromper la discriminatrice. Plus formellement, on note la discriminatrice D, paramétrée par θ_D et la génératrice G, paramétrée par θ_G. Chacune des fonctions dispose d'une fonction de coût. La discriminatrice doit minimiser $J^D(\theta_D, \theta_G)$ en ne pouvant agir que sur θ_D. Inversement, la génératrice doit minimiser $J^G(\theta_D, \theta_G)$ en n'agissant que sur θ_G. La possibilité d'une fonction de n'agir que sur ses paramètres rappelle donc la structure du jeu et la solution est un équilibre de Nash, i.e : le tuple (θ_D, θ_G) qui minimise à la fois J_D et J_G

4.5 Entrainement

Descentes de gradient. A chaque étape, deux minibatch sont tirés : un minibatch x du dataset et un minibatch z de valeurs issues du prior du modèle. Les deux descentes de gradients sont ensuite effectuées simultanément (dans la littérature, certains préconisent de faire plusieurs optimisation sur un modèle avant de passer sur l'autre, mais pas Goodfellow). Pour la fonction discriminatrice, on peut utiliser la fonction de coût *cross-entropy* :

$$J_D(\theta_D, \theta_G) = -\frac{1}{2}\mathbb{E}_{x\sim p_data} \log D(x) - \frac{1}{2}\mathbb{E}_z \log(1 - D(G(z)))$$

Les labels sont 1 pour les exemples du dataset et 0 pour les exemples générés.

5 HTM

HTM pour Hierarchical Temporal Memory. Approche alternative au paradigme IA actuel. En substance, l'idée est de répliquer le fonctionnement du cerveau humain. Cette démarche est motivée

par le fait que la structure du cerveau est (quasi-)similaire en tout point. Ainsi, il n'existe pas un algorithme spécifique de traitement de l'image, un autre dédié au son, un troisième au toucher et ainsi de suite. Par conséquent, ce qui fait qu'une zone du cerveau est considérée comme la zone de traitement visuelle est uniquement le fait que cette zone reçoit des signaux en provenance des yeux.

Les évangélistes de cette approches stipulent que pour développer des machines intelligentes, il est nécessaire de comprendre le fonctionnement du cerveau. Avec ces connaissances, il sera possible de deviser des principes pour répliquer l'algorithme cortical.

Les trois principes fondamentaux de cette approche sont :

- La mémoire
- Une dimension temporelle
- La hiérarchie

5.1 Mémoire

Le néocortex, la couche supérieure du cerveau, uniquement présente chez les mamifères et particulièrement développée chez l'homme est (?). Le néo-cortex doit deviser de la structure du monde *via* les patterns qu'il reçoit en provenance des 5 sens. Et puisque l'algorithme cortical est le même quelque soit le sens en question, alors une machine équipée de cet algorithme pourrait reposer sur des sens complètement différents (GPS, LiDAR...).

Références

[1] Andrychowicz, M., Wolski, F., Ray, A., Schneider, J., Fong, R., Welinder, P., McGrew, B., Tobin, J., Abbeel, P., and Zaremba, W. Hindsight Experience Replay. *ArXiv e-prints* (jul 2017).

[2] Finn, C., Christiano, P., Abbeel, P., and Levine, S. A Connection between Generative Adversarial Networks, Inverse Reinforcement Learning, and Energy-Based Models. *ArXiv e-prints* (Nov. 2016).

[3] Finn, C., Goodfellow, I., and Levine, S. Unsupervised Learning for Physical Interaction through Video Prediction. *ArXiv e-prints* (May 2016).

[4] Goodfellow, I. NIPS 2016 Tutorial : Generative Adversarial Networks. *ArXiv e-prints* (Dec. 2017).

[5] Schulman, J., Moritz, P., Levine, S., Jordan, M., and Abbeel, P. Trust region policy optimization. *arXiv* (April 2017).

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