Deep Reinforcement Learning

Les meilleures méthodes

Alexandre Piché

INF 8225

Ressources

- https://sites.google.com/view/deep-rl-bootcamp/lectures
- http://rll.berkeley.edu/deeprlcourse/
- http://www0.cs.ucl.ac.uk/staff/d.silver/web/Teaching.html
- http://web.stanford.edu/class/cs234/index.html
- http://psthomas.com/courses/CMPSCI_687_Fall2017.html
- https://katefvision.github.io/

- Les résultats impressionants des réseaux de neurones suggèrent qu'ils peuvent être utilisés en apprentisage par renforcement pour résoudre des problèmes de controle.
- La difficulté est de trouver la bonne manière de les utiliser pour résoudre des problèmes non stationnaires (la distribution change lorsqu'on change nos paramètres).
- Dans ce cours, nous allons résumer certains des résultats les plus prometteurs pour résoudre l'apprentisage par renforcement.

Les Composants d'un Agent

- Politique: une politique peut soit être déterministique: $a = \pi(s)$ ou stochastique: $\pi(a|s) = P(a|s)$.
- Valeur: une valeur détermine la somme totale discountée des récompenses: $V(s)$.
- Modèle: la représentation de l'environnement.

Outline

Outline

Atari

https://www.youtube.com/watch?v=V1eYniJ0Rnk

Revue de la Fonction de Valeur

- La fonction de valeur V^π prédit le nombre de récompenses que l'agent va recevoir en suivant la politique π.
- La fonction Q prédit le nombre de récompenses que l'agent va recevoir s'il sélectionne l'action a dans la situation s et ensuite suit la politique π.

$$Q^\pi(s, a) = \mathbb{E}[r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \ldots | s, a]$$

- γ est le facteur d'actualisation.

Itération de la Politique

- Il est possible de définir une politique seulement en sélectionant la meilleure action possible dans chaque situation s.

$$\pi(s_t) = \underset{a \in \mathcal{A}}{\text{argmax}}\, Q(s_t, a)$$
$$Q(s, a) = r(s_t, a_t) + \gamma E_{s_{t+1} \sim p(s_{t+1}|s_t, a_t)}[V^{\pi}(s_{t+1})]$$

- Nous allons approximer la valeur $Q(s, a)$ avec un réseau de neurones $Q_\theta(s, a)$.

Fonction Q

- Il est possible d'approximer la valeur d'une situation avec la Q-function.

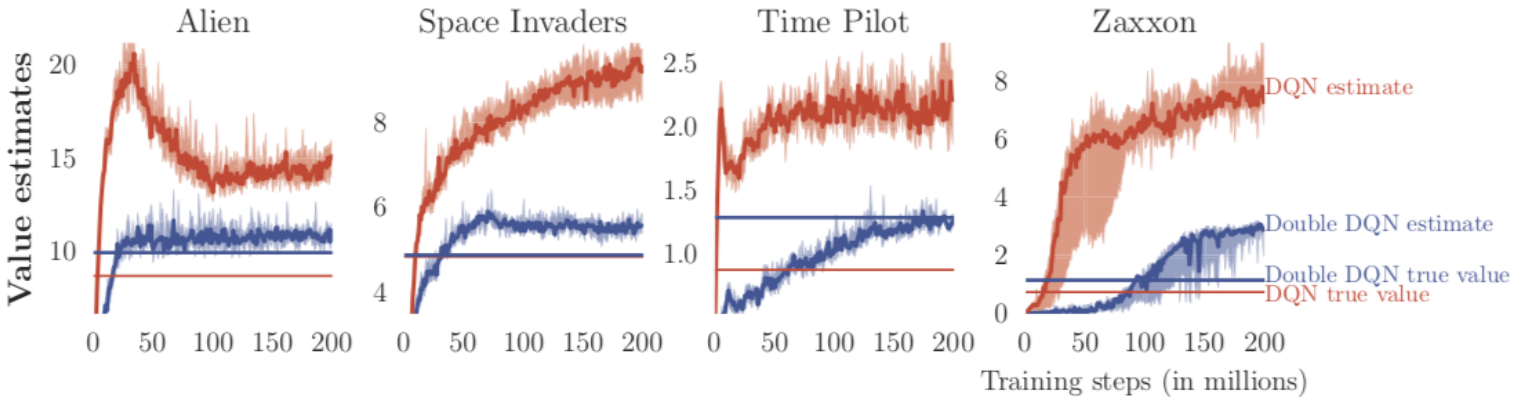
$$E[V^*(s_{t+1})] \approx \max_{a_{t+1}} Q(s_{t+1}, a_{t+1})$$

- Il est possible d'apprendre la valeur $Q_\theta(s, a)$

$$\delta = R(s_t, a_t, s_{t+1}) + \gamma max_{a_{t+1}} Q_\theta(s_{t+1}, a_{t+1})$$
$$\min_\theta (Q_\theta(s, a) - \delta)^2$$

- Similaire à la programmation dynamique.
- δ est non stationnaire puisqu'il est optimisé en même temps que Q_θ.
- Pour stabiliser l'apprentissage, on doit transformer Q learning pour que le problème soit approximativement stationnaire (régression).

Sur-Évaluation des valeurs de Q



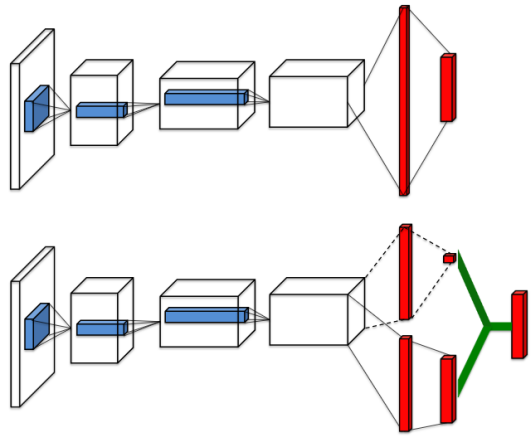
- Plutôt que d'utiliser le même réseau pour estimer le retour d'une action et choisir la meilleure action, on peut diviser le travail.
- Utilise un réseau pour choisir les actions et un pour évaluer les valeurs.

$$\delta = R(s_t, a_t, s_{t+1}) + \gamma max_{a_{t+1}} Q_{\theta^-}(s_{t+1}, a_{t+1})$$

$$a_{t+1} = \underset{a_{t+1}}{\text{argmax}}\, Q_\theta(s_{t+1}, a_{t+1})$$

Van Hasselt, Guez, and Silver 2016

Estimation de la Fonction Avantage



- Il est possible de mettre à jour la valeur d'un état $V(s)$ après chaque itération.
- Ce qui donne un apprentissage plus stable.

$$A(s,a) = Q(s,a) - V(s)$$
$$Q(s,a) = A(s,a) + V(s)$$

Wang, Schaul, et al. 2015

Apprentisage Hors-Ligne et Banque de Données

- Il est important de pouvoir apprendre à partir des expériences venant d'une autre politique.
- Imaginez avoir une transition $(s_t, a_t, s_{t+1}, r_t) \sim \mu$.
- Il est possible d'utiliser la transition pour optimiser notre politique π.

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(r_t + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t))$$

- Il est possible de stocker d'anciennes trajectoires et de les utiliser pour apprendre Q_θ.
- Permet d'avoir des données i.i.d. ce qui réduit la variance des gradients.

Exploration

ϵ-Greedy

- Pour éviter de converger à une politique sub-optimale, on doit s'assurer d'explorer l'environnement.

$$\pi'(a|s) = \begin{cases} 1 + \epsilon - \epsilon/m & \text{argmax}_{a \in \mathcal{A}} Q(s,a) \\ \epsilon/m & \text{otherwise} \end{cases}$$

Exploration II

- Ajouter du bruit aux paramètres du modèle semble aussi aider l'exploration.

$$y = wx + b \text{ sans bruit}$$
$$y = (\mu^w + \sigma^w \odot \epsilon^w)x + \mu^b + \sigma^b \odot \epsilon^b \text{ avec bruit}$$

- σ^w et σ^b sont des paramètres appris.

Fortunato et al. 2017; Plappert et al. 2017

Apprentissage

1: Initialisez un réseau pour les actions Q et paramètre θ
2: Initialisez un réseau cible $\hat{Q}$ et paramètre $\theta^- = \theta$
3: **for** episode - 1, . . . , M **do**
4: Observer la première observation x_1
5: **for** t-1, . . . , T **do**
6: Choisir une action au hasard avec probabilité ϵ et $a_t = \text{argmax}_a\, Q * (\phi(s_t), a; \phi)$ avec probabilité $1 - \epsilon$
7: Executer l'action dans le simulateur, observer l'observation suivante x_{t+1} et la récompense r_t
8: Ajouter $\{(s_t, a_t, s_{t+1}, r_t)\}$ à la base de données $\mathcal{B}$.
9: Sélectionner un batch de données (s_i, a_i, s'_i, r_i) de $\mathcal{B}$
10: $$y_j = \begin{cases} r_j \text{ si c'est la fin de l'épisode} \\ r_j + \gamma \max_{a'} \hat{Q}(_{j+1}, a'; \theta^-) \text{ autrement} \end{cases}$$
11: $\phi \leftarrow \phi - \alpha \sum_i \frac{dQ_\phi}{d\phi}(y_j - Q_\phi(s_i, a_i))^2$
12: **end for** =0

Détails de L'implémentation

- RMSProp (ou ADAM) plutôt que SGD.
- Huber loss plutôt que l'erreur au carré (MSE)

$$L_\delta(a) = \begin{cases} \frac{1}{2}a^2 \text{ for } |a| \leq \delta \\ \delta(|a| - \frac{1}{2}\delta), \text{ otherwise} \end{cases}$$

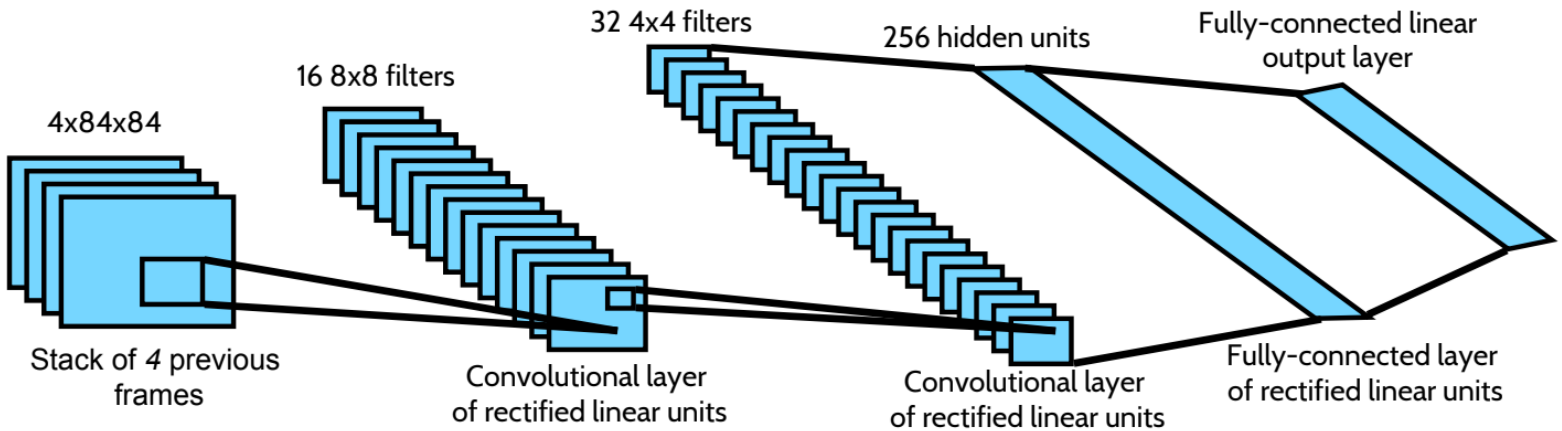
- Commencer avec un taux d'exploration de 1 et le diminuer le jusqu'à 0.1 pendant les premieres 1M situations.
- Prioriser les transitions avec les plus grandes erreurs de Bellman $\|r + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}; \theta^-) - Q(s, a; \theta)\|$

Volodymyr Mnih, Kavukcuoglu, et al. 2013; Schaul et al. 2015

Architecture de DQN

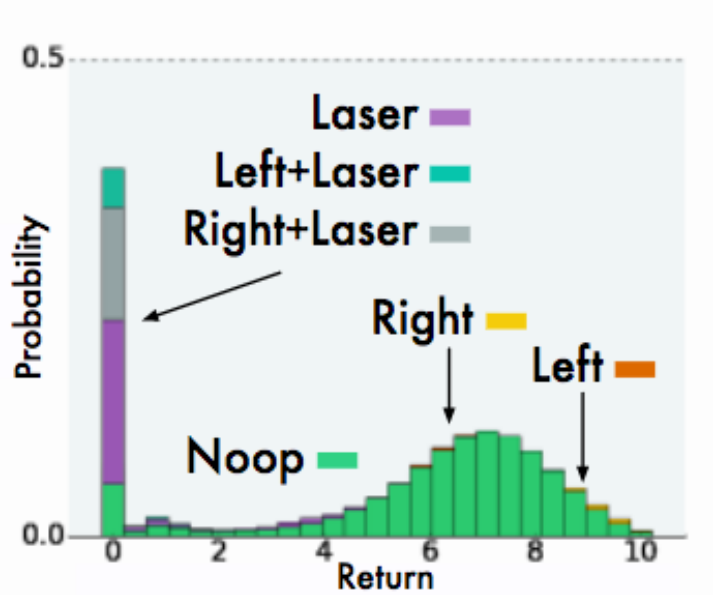
ATARI Network Architecture

- Convolutional neural network architecture:
  - History of frames as input.
  - One output per action - expected reward for that action $Q(s, a)$.
  - Final results used a slightly bigger network (3 convolutional + 1 fully-connected hidden layers).



Mnih's slides:
https://drive.google.com/file/d/0BxXI_RttTZAhVUhpbDhiSUFFNjg/view

Estimer la Valeur Q avec une Distribution



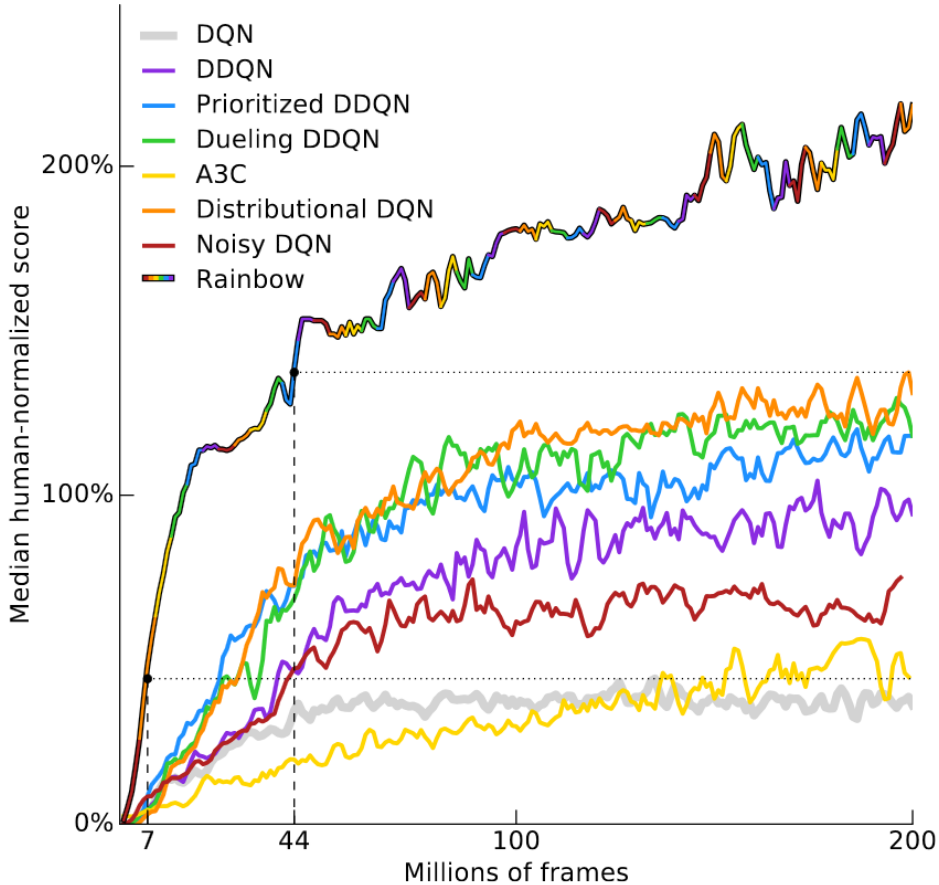
$$Q(s_t, a_t) = \mathbb{E}[R(s_t, a_t)] + \gamma \mathbb{E}[Q(s_{t+1}, a_{t+1})]$$

$$Z(s_t, a_t) \overset{D}{=} R(s_t, a_t) + \gamma Z(s_{t+1}, a_{t+1})$$

Bellemare, Dabney, and Rémi Munos 2017

Rainbow

- Il y a moyen d'intégrer plusieurs algorithmes ensemble pour obtenir de meilleurs résultats.

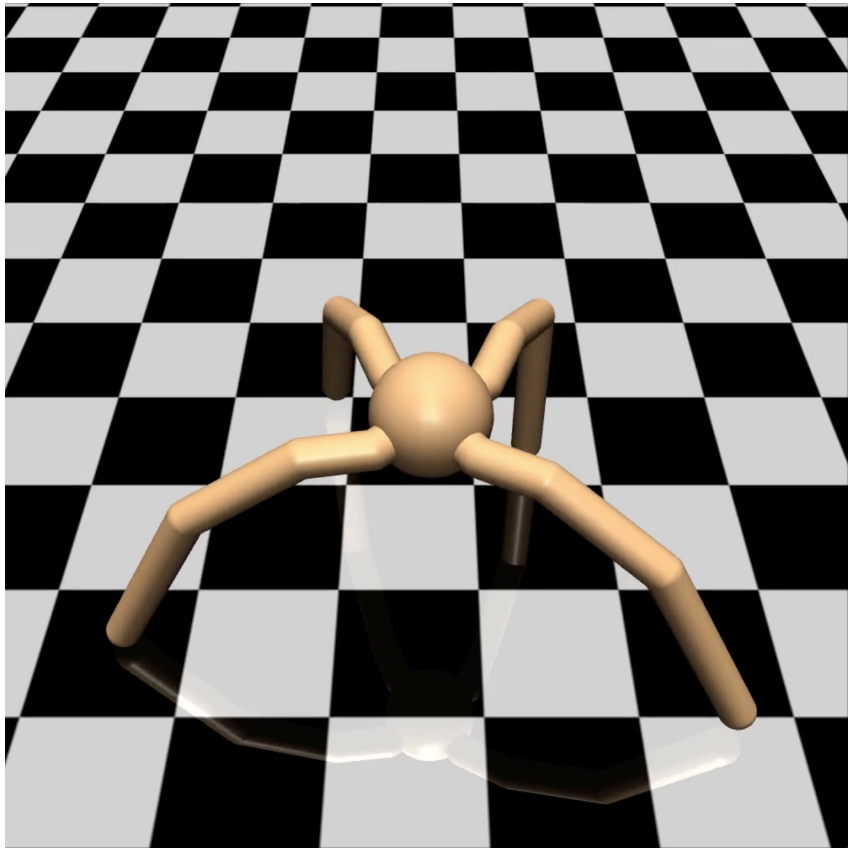


Hessel et al. 2017

Outline

Robotique

Mujoco



`https://www.youtube.com/watch?v=Ajjc08-iPx8`

- Apprentissage par gradient de la politique obtiennent présentement certains des meilleurs résultats sur Mujoco et Atari.

Apprentissage par Renforcement avec une Politique

Gradient de la Politique

- On est intéressé par maximiser $\max_\theta \mathbb{E}[\sum_{t=0}^{T} R(s_t)|\pi_\theta]$, où $\pi_\theta(a|s)$ est une fonction stochastique.
- Il est parfois plus simple d'optimiser la politique directement que de résoudre Q ou V.
- Dans le cas de la fonction Q, on doit être en mesure de résoudre $\text{argmax}_a\, Q_\theta(s, a)$, difficile pour des actions continues.

Sutton et al. 2000; Lillicrap et al. 2015

Problème

- On observe des trajectoires et on veut maximiser la somme des récompenses: $r(\tau) = \sum_{t=1}^{T} r(s_t, a_t)$
- Notez que les trajectoires dependent de π_θ.

$$
\begin{aligned}
J(\theta) &= \mathbb{E}_{\tau \sim \pi_\theta(\tau)}[r(\tau)] \\
&= \int \pi_\theta(\tau) r(\tau) d\tau
\end{aligned}
$$

Ratio de vraisemblance

Gradient de la Politique

$$\nabla_\theta J(\theta) = \int \nabla_\theta \pi_\theta(\tau) r(\tau) d\tau$$

Ratio de vraisemblance

Gradient de la Politique

$$\begin{aligned}\nabla_\theta J(\theta) &= \int \nabla_\theta \pi_\theta(\tau) r(\tau) d\tau \\ &= \int \frac{\pi_\theta(\tau)}{\pi_\theta(\tau)} \nabla_\theta \pi_\theta(\tau) r(\tau) d\tau\end{aligned}$$

Ratio de vraisemblance

Gradient de la Politique

$$
\begin{aligned}
\nabla_\theta J(\theta) &= \int \nabla_\theta \pi_\theta(\tau) r(\tau) d\tau \\
&= \int \frac{\pi_\theta(\tau)}{\pi_\theta(\tau)} \nabla_\theta \pi_\theta(\tau) r(\tau) d\tau \\
&= \int \pi_\theta(\tau) \frac{\nabla_\theta \pi_\theta(\tau)}{\pi_\theta(\tau)} r(\tau) d\tau
\end{aligned}
$$

Ratio de vraisemblance

Gradient de la Politique

$$
\begin{aligned}
\nabla_\theta J(\theta) &= \int \nabla_\theta \pi_\theta(\tau) r(\tau) d\tau \\
&= \int \frac{\pi_\theta(\tau)}{\pi_\theta(\tau)} \nabla_\theta \pi_\theta(\tau) r(\tau) d\tau \\
&= \int \pi_\theta(\tau) \frac{\nabla_\theta \pi_\theta(\tau)}{\pi_\theta(\tau)} r(\tau) d\tau \\
&= \int \pi_\theta(\tau) \nabla_\theta \log \pi_\theta(\tau) r(\tau) d\tau
\end{aligned}
$$

Ratio de vraisemblance

Gradient de la Politique

$$
\begin{aligned}
\nabla_\theta J(\theta) &= \int \nabla_\theta \pi_\theta(\tau) r(\tau) d\tau \\
&= \int \frac{\pi_\theta(\tau)}{\pi_\theta(\tau)} \nabla_\theta \pi_\theta(\tau) r(\tau) d\tau \\
&= \int \pi_\theta(\tau) \frac{\nabla_\theta \pi_\theta(\tau)}{\pi_\theta(\tau)} r(\tau) d\tau \\
&= \int \pi_\theta(\tau) \nabla_\theta \log \pi_\theta(\tau) r(\tau) d\tau \\
&= \mathbb{E}_{\tau \sim \pi_\theta(\tau)} [\nabla_\theta \log \pi_\theta(\tau) r(\tau)]
\end{aligned}
$$

Intuition et estimation

- Les trajectoires avec les meilleurs récompenses sont maximisées.
- Rend plus probables les bonnes trajectoires.
- Maximiser avec des exemples tirés de l'environnement.

$$\nabla_\theta J(\theta) \approx \sum_{i=1}^{m} \nabla_\theta \log \pi_\theta(\tau^{(i)}) r(\tau)$$

Trajectoires

$$\begin{aligned}
\nabla_\theta \log \pi_\theta(\tau^{(i)}) &= \nabla_\theta \log \left[\prod_{t=0}^{H} P(s_{t+1}^{(i)} | s_t^{(i)}, a_t^{(i)}) \pi_\theta(a_t^{(i)} | s_t^{(i)}) \right] \\
&= \nabla_\theta \left[\sum_{t=0}^{H} \log P(s_{t+1}^{(i)} | s_t^{(i)}, a_t^{(i)}) + \sum_{t=0}^{H} \log \pi_\theta(a_t^{(i)} | s_t^{(i)}) \right] \\
&= \nabla_\theta \sum_{t=0}^{H} \log \pi_\theta(a_t^{(i)} | s_t^{(i)})
\end{aligned}$$

On n'a pas besoin de connaitre les probabilités de transitions.

Structure Temporelle

- Les récompenses $\{r_t\}_{t=0}^{l}$ ne dépendent pas des actions $\{a_t\}_{t=l}^{H-1}$

$$\nabla J(\theta) = \frac{1}{m}\sum_{i=1}^{m} \nabla_\theta \log \pi_\theta(a_t^{(i)}|s_t^{(i)} R(\tau^i)$$

$$= \frac{1}{m}\sum_{i=1}^{m}\sum_{t=0}^{H-1} \nabla_\theta \log \pi_\theta(a_t^{(i)}|s_t^{(i)}(\sum_{k=0}^{H-1} R(s_k^{(i)}, a_k^{(i)}))$$

$$= \frac{1}{m}\sum_{i=1}^{m}\sum_{t=0}^{H-1} \nabla_\theta \log \pi_\theta(a_t^{(i)}|s_t^{(i)}(\sum_{k=t}^{H-1} R(s_k^{(i)}, a_k^{(i)}))$$

Algorithme Reinforce

Retour Monte-Carlo

REINFORCE, A Monte-Carlo Policy-Gradient Method (episodic)

Input: a differentiable policy parameterization $\pi(a|s, \boldsymbol{\theta}), \forall a \in \mathcal{A}, s \in \mathcal{S}, \boldsymbol{\theta} \in \mathbb{R}^d$
Initialize policy parameter $\boldsymbol{\theta}$
Repeat forever:
    Generate an episode $S_0, A_0, R_1, \ldots, S_{T-1}, A_{T-1}, R_T$, following $\pi(\cdot|\cdot, \boldsymbol{\theta})$
    For each step of the episode $t = 0, \ldots, T-1$:
        $G \leftarrow$ return from step t
        $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha \gamma^t G \nabla_{\boldsymbol{\theta}} \log \pi(A_t|S_t, \boldsymbol{\theta})$

Control variate

- Notez que si tous les retours sont positifs, le gradient augmente la probabilité de toutes les trajectoires.
- On peut faire mieux en augmentant seulement la probabilité des trajectoires meilleures que la moyenne.

$$\nabla J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \sum_{t=0}^{H-1} \nabla_\theta \log \pi_\theta(a_t^{(i)} | s_t^{(i)}) (\sum_{k=t}^{H-1} R(s_k^{(i)}, a_k^{(i)}) - b)$$

- Il est possible de démontrer que le gradient est sans biais.
- En pratique $V(s_k^{(i)})$ donne de bons résultats.
- On dénote l'avantage: $A^\pi(s, a) = \sum_{k=t}^{H-1} R(s_k^{(i)}, a_k^{(i)}) - V^\pi(s)$

---

wu2018variance ; Grathwohl et al. 2017

Estimation de V^π

Difference Temporel

- On peut estimer la valeur d'une situation de plusieurs manières.

$$\begin{aligned} V^\pi(s_t) &= \mathbb{E}[r_0 + \gamma r_1 + \gamma^2 r_2 + \ldots | s_0 = s, a_0 = a] = G_{t:\infty} \\ &= \mathbb{E}[r_0 + \gamma V^\pi(s_1) | s_0 = s, a_0 = a] = G_{t:t+1} \\ &= \mathbb{E}[r_0 + \gamma r_1 + \gamma^2 V^\pi(s_2) | s_0 = s, a_0 = a] = G_{t:t+2} \\ &\ldots \end{aligned}$$

- A3C utilise 5 pas pour estimer la valeur.

$$\delta = \sum_{i=0}^{4} \gamma^i r_i + \gamma^5 V_\theta^p i(s_{t+5})$$

$$\min_\theta \|\delta - V_\theta^\pi(s_t)\|^2$$

Volodymyr Mnih, Adria Puigdomenech Badia, et al. 2016

TD(λ)

- Il est possible de faire une moyenne de chaque estimation pour controler le compromis entre la variance et le bias.

$$G_t = (1 - \lambda) \sum_{n=1}^{\infty} \lambda^{n-1} G_{t:t+n}$$

Schulman, Moritz, et al. 2015; Sutton 1988

TD(λ)

- Il est possible de faire une moyenne de chaque estimation pour controler le compromis entre la variance et le bias.

$$G_t = (1-\lambda)\sum_{n=1}^{\infty}\lambda^{n-1}G_{t:t+n}$$

- Une méthode populaire est d'utiliser une moyenne des avantages (GAE).

$$\begin{aligned}\hat{A}_t^{\mathrm{GAE}(\gamma,\lambda)} &= (1-\lambda)(\hat{A}_t^{(1)} + \lambda\hat{A}_t^{(1)} + \lambda^2\hat{A}_t^{(2)} + \ldots) \\ &= \sum_{l=0}^{\infty}(\lambda\gamma)^l\delta_{t+l}^V\end{aligned}$$

Schulman, Moritz, et al. 2015; Sutton 1988

Algorithme Acteur Critique

One-step Actor–Critic (episodic)

Input: a differentiable policy parameterization $\pi(a|s,\boldsymbol{\theta}), \forall a \in \mathcal{A}, s \in \mathcal{S}, \boldsymbol{\theta} \in \mathbb{R}^d$
Input: a differentiable state-value parameterization $\hat{v}(s,\mathbf{w}), \forall s \in \mathcal{S}, \mathbf{w} \in \mathbb{R}^m$
Parameters: step sizes $\alpha > 0$, $\beta > 0$

Initialize policy parameter $\boldsymbol{\theta}$ and state-value weights $\mathbf{w}$
Repeat forever:
  Initialize S (first state of episode)
  $I \leftarrow 1$
  While S is not terminal:
    $A \sim \pi(\cdot|S,\boldsymbol{\theta})$
    Take action A, observe S', R
    $\delta \leftarrow R + \gamma \hat{v}(S',\mathbf{w}) - \hat{v}(S,\mathbf{w})$ (if S' is terminal, then $\hat{v}(S',\mathbf{w}) \doteq 0$)
    $\mathbf{w} \leftarrow \mathbf{w} + \beta \delta \nabla_{\mathbf{w}} \hat{v}(S,\mathbf{w})$
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha I \delta \nabla_{\boldsymbol{\theta}} \log \pi(A|S,\boldsymbol{\theta})$
    $I \leftarrow \gamma I$
    $S \leftarrow S'$

Échantillonnage préférentiel

- Il est possible d'estimer l'espérance d'une distribution Q lorsqu'on a des données de la distribution P.
- On doit être en mesure de calculer $P(X)$ et $Q(X)$

$$\begin{aligned}\mathbb{E}_{X\sim P}[f(X)] &= \sum P(X)f(X) \\ &\sum Q(X)\frac{P(X)}{Q(X)}f(X) \\ &= \mathbb{E}_{X\sim Q}\Big[\frac{P(X)}{Q(X)}f(X)\Big]\end{aligned}$$

Apprentissage Hors Ligne

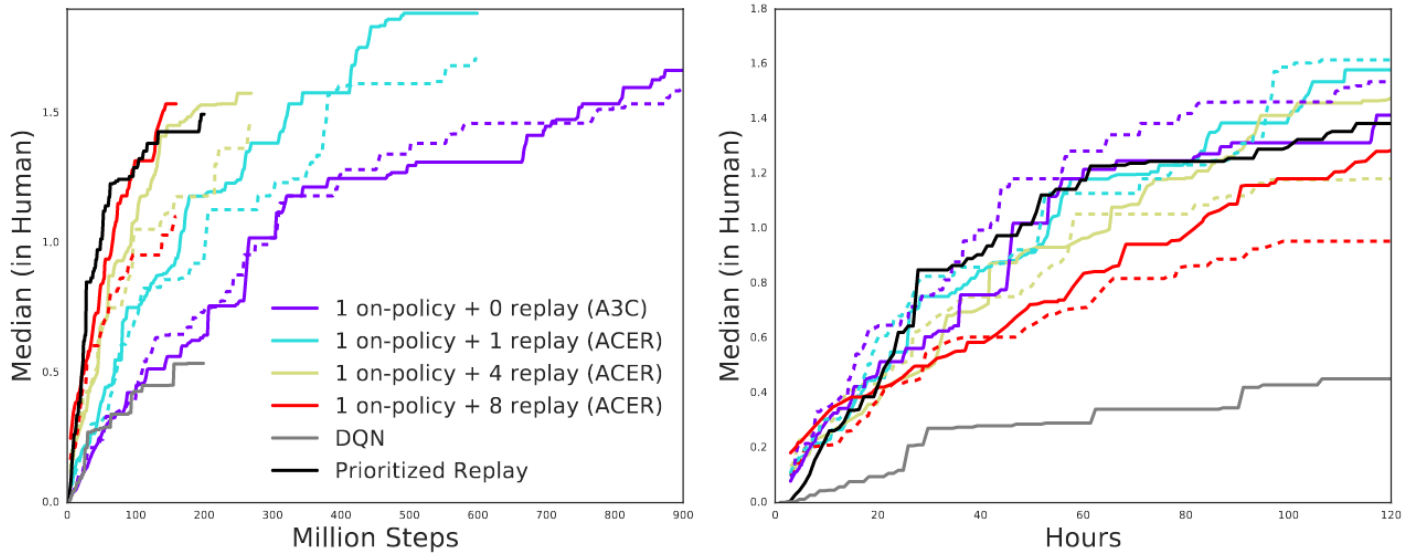
- Il est possible d'utiliser l'échantillonnage préférentiel pour estimer les gradient de la politique lorsque les trajectoires ont été obtenues sous une politique différente
- Différentes modifications existent pour réduire la variance de l'échantillonnage préférentiel

$$\mathbb{E}_{\tau\sim\mu(a|\cdot)}[\nabla \log \pi_\theta(\tau) R(\tau)] = \left(\prod_{t=0}^{k} \rho_t\right) \sum_{t=0}^{k} \left(\sum_{i=t}^{k} \gamma^i r_i\right) \nabla_\theta \log \pi_\theta(a_t|s_t)$$

$$\rho_t = \frac{\pi(a_t|s_t)}{\mu(a_t|s_t)}$$

Wang, Bapst, et al. 2016; Gruslys et al. 2017

ACER



- De bons résultats peuvent être obtenus en tronquant le ratio d'échantillonage préférentiel.

Algorithme à Région de Confiance

Politique Proximale

- En apprentissage supervisé une update trop grande peut être corrigée par la suivante.
- En apprentissage par renforcement, une update trop grande va changer les trajectoires observées et peut amener l'agent a diverger.

Schulman, Levine, et al. 2015; Schulman, Wolski, et al. 2017; Wu et al. 2017

Gradient de la Politique

Échantillonnage préférentiel

- Il est possible de dériver le gradient de la politique basée sur l'échantillonnage préférentiel.

$$J^{IS}_{\theta_{old}}(\theta) = \mathbb{E}_t \left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)} A^{\pi_{\text{old}}}(s_t, a_t) \right]$$
$$= \mathbb{E}_t \left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)} \hat{A}_t \right]$$

Kakade and Langford 2002

Surrogate Objective

- Il est possible de pénaliser la distance entre les politiques $\pi_{\theta_{\text{old}}}$ et π_θ

$$L(\pi) = \mathbb{E}_{\pi_{\theta_{\text{old}}}} \left[\frac{\pi_\theta(a|s)}{\pi_{\theta_{\text{old}}}(a|s)} A^{\pi_{\text{old}}}(s,a) \right]$$

$$L(\pi) = \mathbb{E}_{\pi_{\theta_{\text{old}}}} \left[\frac{\pi_\theta(a|s)}{\pi_{\theta_{\text{old}}}(a|s)} A^{\pi_{\text{old}}}(s,a) \right] - \beta KL(\pi_{\theta_{\text{old}}}, \pi_\theta)$$

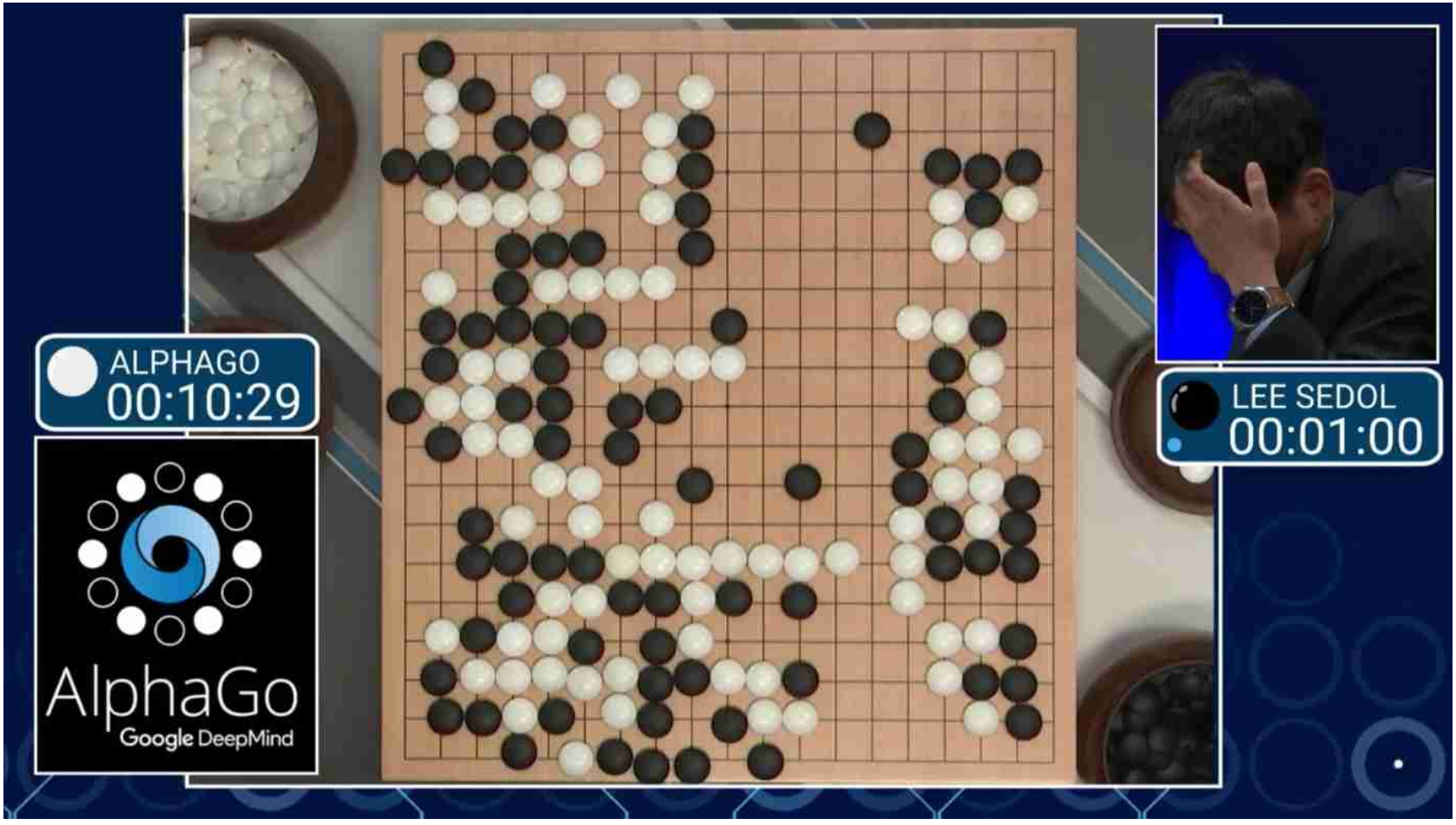
- KL est une mesure de "distance" entre 2 distributions.

$$KL(P||Q) = \int_{-\infty}^{\infty} p(x) \log \frac{p(x)}{q(x)} dx$$

Outline

Plannification et l'Utilisation de Modèles

Alpha Go



https://www.youtube.com/watch?v=8tq1C8spV_g

Plannification

Monte Carlo Tree Search

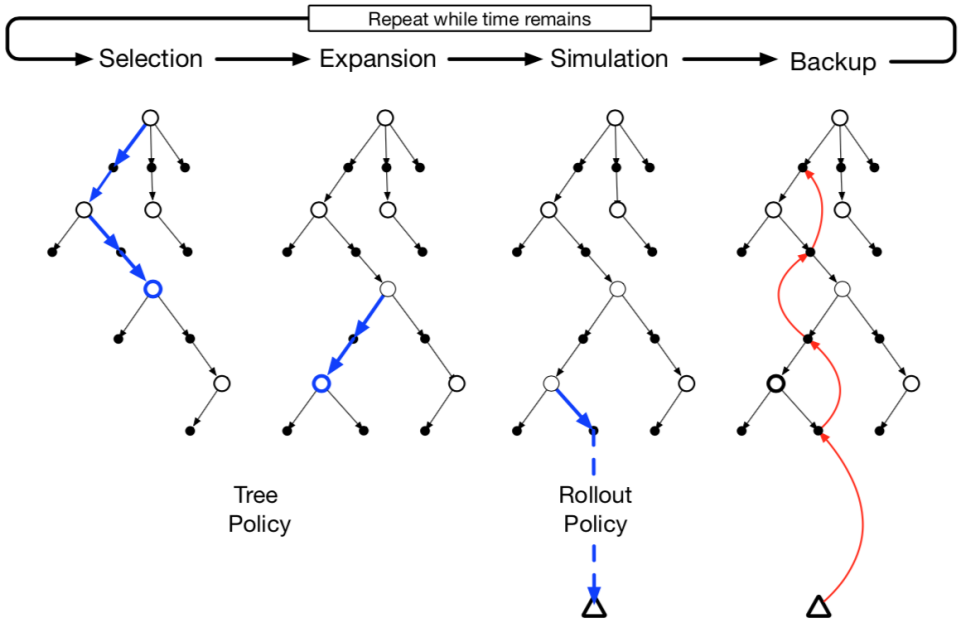
- Lorsque la dynamique du monde est connue et déterministique, on peut utiliser des simulations pour planifier et choisir la meilleure action.
- On peut partir de notre position et prendre des actions aux hasard jusqu'à une profondeur donnée e.g. la fin de la partie.
- On peut utiliser ces simulations pour estimer $Q(s, a)$.
- Utiliser des simulations nous permet d'éviter le fléau de la dimension.

Silver et al. 2017; Guo et al. 2014

Simple Simulation de Monte-Carlo

1: **for** Pour chaque action $a \in \mathcal{A}$ **do**
2: Simuler K épisodes à partir de notre position actuelle: $\{s_t, a, R^k_{t+1}, S^k_{t+1}, A^k_{t+1}, \ldots, S^k_T\}^K_{k=1} \sim \mathcal{M}, \pi_r$.
3: Évaluer la valeur $Q(s_t, a) = \frac{1}{K} \sum^K_{k=1} G_t$
4: **end for**
5: Sélectionner la meilleure action: $a_t = \text{argmax}_{a \in \mathcal{A}} Q(s_t, a)$ =0

Monte Carlo Tree Search



- On peut faire mieux si on garde en mémoire le nombre de fois qu'on a visité une situation s:
  $Q(s,a) = \frac{1}{N(s,a)} \sum_{k=1}^{K} \sum_{u=t}^{T} 1(S_u, A_u = s, a) G_u$.
- On peut ensuite utilisez ϵ-greedy pour les valeurs qu'on connait et simuler pour les situations inconnues.
- Slide 38-43 `http://www0.cs.ucl.ac.uk/staff/d.silver/web/Teaching_files/dyna.pdf`

Pourquoi Utiliser un Modèle

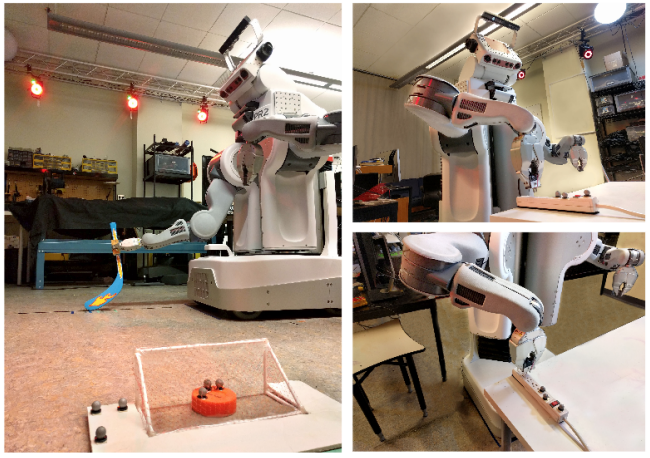


Figure 1. Real robot tasks used to evaluate our method. Left: The hockey task which involves discontinuous dynamics. Right: The power plug task which requires high level of precision. Both of these tasks are learned from scratch without demonstrations.

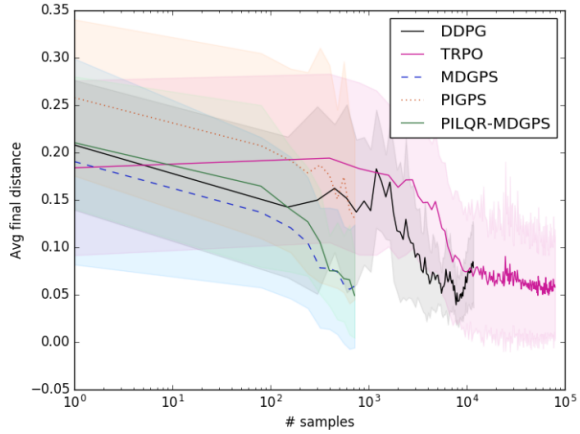


Figure 4. Final distance from the reacher end effector to the target averaged across 300 random test conditions per iteration. MDGPS with LQR-FLM, MDGPS with PILQR, TRPO, and DDPG all perform competitively. However, as the log scale for the x axis shows, TRPO and DDPG require orders of magnitude more samples. MDGPS with PI^2 performs noticeably worse.

- Si on veut utiliser Deep RL dans le monde réel, on doit être très efficace avec les données que nous avons.

Chebotar et al. 2017

Model Predictive Control

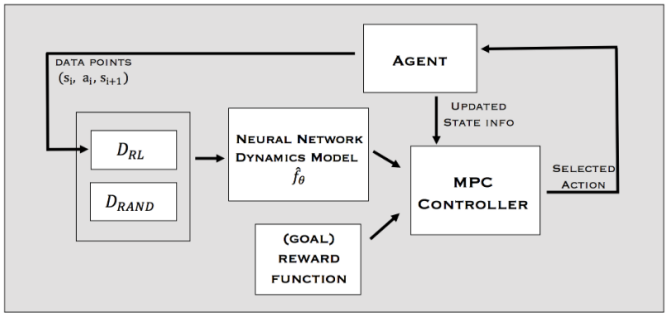


Fig. 2: Illustration of Algorithm 1. On the first iteration, random actions are performed and used to initialize $\mathcal{D}_{\text{RAND}}$. On all following iterations, this iterative procedure is used to train the dynamics model, run the MPC controller for action selection, aggregate data, and retrain the model.

- Si on a un modèle différentiable du monde, il est possible de directement maximiser nos récompenses.

1: Run base policy $\pi_0(a_t, s_t)$ to collect $D = \{(s, a, s')_i\}$
2: Learn model $f_\phi(s, a)$ to minimize $\sum_i \|f_\phi(s_i, a_i) - s'_i\|^2$
3: Backpropagate through $f_\phi(s, a)$ into policy to optimize $\pi_\theta(a_t|s_t)$

Nagabandi et al. 2017

DYNA

- Si on a un modèle du monde, il est aussi possible de simuler des trajectoires et d'apprendre de celles ci.

Tabular Dyna-Q

Initialize $Q(s,a)$ and $Model(s,a)$ for all $s \in \mathcal{S}$ and $a \in \mathcal{A}(s)$
Do forever:
- (a) $S \leftarrow$ current (nonterminal) state
- (b) $A \leftarrow \epsilon$-greedy(S,Q)
- (c) Execute action A; observe resultant reward, R, and state, S'
- (d) $Q(S,A) \leftarrow Q(S,A) + \alpha\big[R + \gamma \max_a Q(S',a) - Q(S,A)\big]$
- (e) $Model(S,A) \leftarrow R, S'$ (assuming deterministic environment)
- (f) Repeat n times:
  - $S \leftarrow$ random previously observed state
  - $A \leftarrow$ random action previously taken in S
  - $R, S' \leftarrow Model(S,A)$
  - $Q(S,A) \leftarrow Q(S,A) + \alpha\big[R + \gamma \max_a Q(S',a) - Q(S,A)\big]$

Sutton 1991; Gu et al. 2016

Outline

Imagination

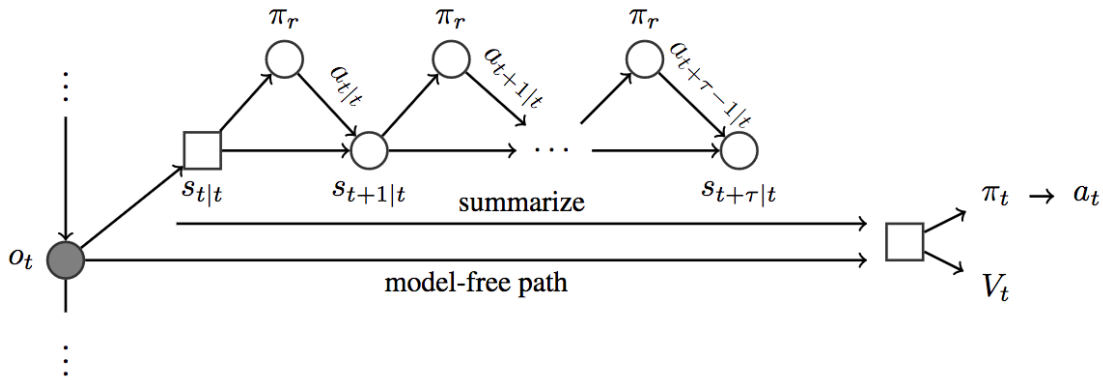


Figure 2. The architecture of the Imagination-Augmented Agent, which computes its policy π_t and value function V_t, by combining information from a model-free path with information from Monte-Carlo rollouts of its environment model.

- Habituellement, nous n'avons pas accès à un modèle parfait du monde, mais il est quand même possible d'utiliser l'information d'un modèle.
- Utilisez le model pour générer des transitions possibles.
- Utilisez ces transitions simulées pour améliorer la représentation de la situation.

Théophane Weber et al. 2017; Buesing et al. 2018

Apprentissage par Renforcement Hierarchique

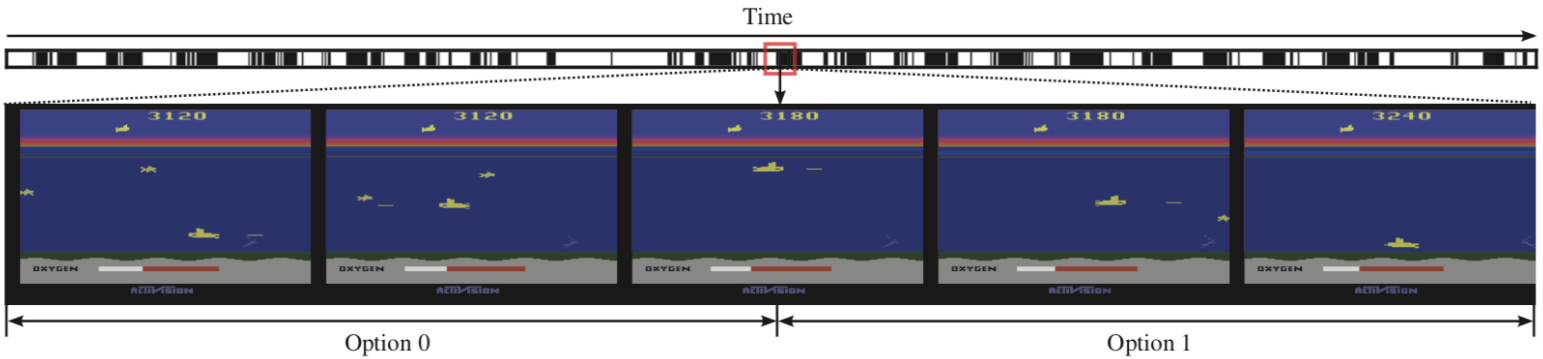


Figure 9: Up/down specialization in the solution found by option-critic when learning with 2 options in Seaquest. The top bar shows a trajectory in the game, with "white" representing a segment during which option 1 was active and "black" for option 2.

- Ce n'est pas nécessaire de modeliser chaque micro-action e.g. le mouvement de nos muscles.
- Apprendre des macro-actions nous permettraient d'apprendre beaucoup plus rapidement dans de nouvelles tâches.

Vezhnevets et al. 2017; Bacon, Harb, and Precup 2017

Apprentissage par Renforcement Sans Récompense

- Ce n'est pas toujours possible d'avoir de la supervision sous terme de récompense, ou les récompenses peuvent être très rares.
- Plusieurs groupes développent des agents qui n'ont pas besoin de supervision pour apprendre.

Thomas et al. 2017; Gregor, D. J. Rezende, and Wierstra 2016

Tâches Auxilliaires

- Il est possible d'augmenter la tâche de notre agent, par exemple on peut le récompenser lorsqu'il change les pixels de l'environment.
- Patager une représentation latente du problème améliore le taux de convergence.
- θ est le set de paramètres pour toutes les politiques π.

$$\underset{\theta}{\text{argmax}}\, \mathbb{E}_{\pi}[R_{1:\infty}] + \lambda_c \sum_{c \in C} \mathbb{E}_{\pi^c}[R^c_{1:\infty}]$$

Jaderberg et al. 2016

Apprendre à Apprendre

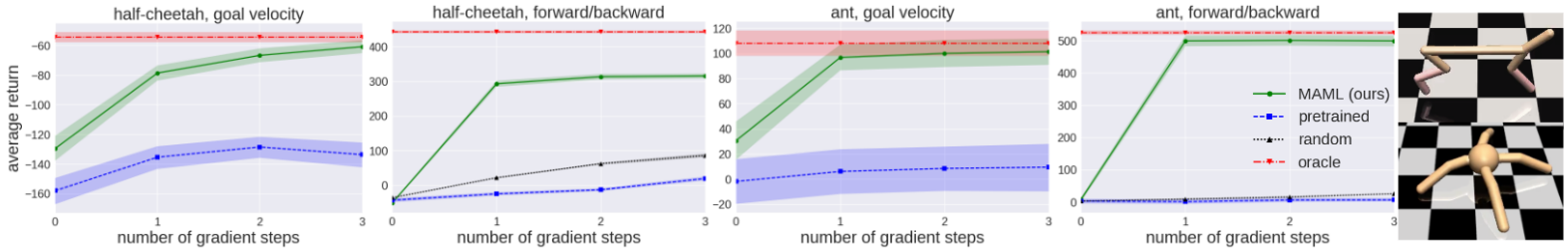


Figure 5. Reinforcement learning results for the half-cheetah and ant locomotion tasks, with the tasks shown on the far right. Each gradient step requires additional samples from the environment, unlike the supervised learning tasks. The results show that MAML can adapt to new goal velocities and directions substantially faster than conventional pretraining or random initialization, achieving good performs in just two or three gradient steps. We exclude the goal velocity, random baseline curves, since the returns are much worse (< -200 for cheetah and < -25 for ant).

- Il est possible d'entrainer un agent à apprendre sur plusieurs tâches pour qu'il puisse apprendre plus rapidement sur une nouvelle tâche.
- `https://vimeo.com/250400382#t=567s`

Finn, Abbeel, and Levine 2017; Duan et al. 2016

Projet

- OpenAI gym: https://github.com/openai/gym
- Mujoco: https://www.roboti.us/license.html, https://github.com/openai/mujoco-py
- Implementation: https://github.com/ikostrikov/pytorch-a2c-ppo-acktr

TP

https://cdn-images-1.medium.com/max/1600/1*oMSg2_mKguAGKy1C64UFlw.gif

References I

- Pierre-Luc Bacon, Jean Harb, and Doina Precup. "The Option-Critic Architecture.". In: *AAAI*. 2017, pp. 1726–1734.
- Marc G Bellemare, Will Dabney, and Rémi Munos. "A distributional perspective on reinforcement learning". In: *arXiv preprint arXiv:1707.06887* (2017).
- Lars Buesing et al. "Learning and Querying Fast Generative Models for Reinforcement Learning". In: *arXiv preprint arXiv:1802.03006* (2018).
- Yevgen Chebotar et al. "Combining model-based and model-free updates for trajectory-centric reinforcement learning". In: *arXiv preprint arXiv:1703.03078* (2017).
- Yan Duan et al. "RL ^{2}: Fast Reinforcement Learning via Slow Reinforcement Learning". In: *arXiv preprint arXiv:1611.02779* (2016).

References II


Chelsea Finn, Pieter Abbeel, and Sergey Levine. "Model-agnostic meta-learning for fast adaptation of deep networks". In: *arXiv preprint arXiv:1703.03400* (2017).


Meire Fortunato et al. "Noisy networks for exploration". In: *arXiv preprint arXiv:1706.10295* (2017).


Will Grathwohl et al. "Backpropagation through the Void: Optimizing control variates for black-box gradient estimation". In: *arXiv preprint arXiv:1711.00123* (2017).


Karol Gregor, Danilo Jimenez Rezende, and Daan Wierstra. "Variational Intrinsic Control". In: *arXiv preprint arXiv:1611.07507* (2016).


Audrunas Gruslys et al. "The Reactor: A Sample-Efficient Actor-Critic Architecture". In: *CoRR* abs/1704.04651 (2017). arXiv: 1704.04651. URL: http://arxiv.org/abs/1704.04651.

References III


Shixiang Gu et al. "Continuous deep q-learning with model-based acceleration". In: *International Conference on Machine Learning*. 2016, pp. 2829–2838.


Xiaoxiao Guo et al. "Deep learning for real-time Atari game play using offline Monte-Carlo tree search planning". In: *Advances in neural information processing systems*. 2014, pp. 3338–3346.


Matteo Hessel et al. "Rainbow: Combining Improvements in Deep Reinforcement Learning". In: *arXiv preprint arXiv:1710.02298* (2017).


Max Jaderberg et al. "Reinforcement learning with unsupervised auxiliary tasks". In: *arXiv preprint arXiv:1611.05397* (2016).


Sham Kakade and John Langford. "Approximately optimal approximate reinforcement learning". In: *ICML*. Vol. 2. 2002, pp. 267–274.

References IV

- Timothy P Lillicrap et al. "Continuous control with deep reinforcement learning". In: *arXiv preprint arXiv:1509.02971* (2015).
- Volodymyr Mnih, Adria Puigdomenech Badia, et al. "Asynchronous methods for deep reinforcement learning". In: *International Conference on Machine Learning*. 2016, pp. 1928–1937.
- Volodymyr Mnih, Koray Kavukcuoglu, et al. "Playing atari with deep reinforcement learning". In: *arXiv preprint arXiv:1312.5602* (2013).
- Anusha Nagabandi et al. "Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning". In: *arXiv preprint arXiv:1708.02596* (2017).
- Matthias Plappert et al. "Parameter space noise for exploration". In: *arXiv preprint arXiv:1706.01905* (2017).

References V

- Tom Schaul et al. "Prioritized experience replay". In: *arXiv preprint arXiv:1511.05952* (2015).
- John Schulman, Sergey Levine, et al. "Trust region policy optimization". In: *International Conference on Machine Learning*. 2015, pp. 1889–1897.
- John Schulman, Philipp Moritz, et al. "High-dimensional continuous control using generalized advantage estimation". In: *arXiv preprint arXiv:1506.02438* (2015).
- John Schulman, Filip Wolski, et al. "Proximal policy optimization algorithms". In: *arXiv preprint arXiv:1707.06347* (2017).
- David Silver et al. "Mastering the game of go without human knowledge". In: *Nature* 550.7676 (2017), p. 354.

References VI


Richard S Sutton. "Dyna, an integrated architecture for learning, planning, and reacting". In: *ACM SIGART Bulletin* 2.4 (1991), pp. 160–163.


Richard S Sutton. "Learning to predict by the methods of temporal differences". In: *Machine learning* 3.1 (1988), pp. 9–44.


Richard S Sutton et al. "Policy gradient methods for reinforcement learning with function approximation". In: *Advances in neural information processing systems*. 2000, pp. 1057–1063.


Valentin Thomas et al. "Independently Controllable Features". In: *arXiv preprint arXiv:1708.01289* (2017).


Hado Van Hasselt, Arthur Guez, and David Silver. "Deep Reinforcement Learning with Double Q-Learning.". In: *AAAI*. Vol. 16. 2016, pp. 2094–2100.

References VII

- Alexander Sasha Vezhnevets et al. "Feudal networks for hierarchical reinforcement learning". In: *arXiv preprint arXiv:1703.01161* (2017).
- Ziyu Wang, Victor Bapst, et al. "Sample efficient actor-critic with experience replay". In: *arXiv preprint arXiv:1611.01224* (2016).
- Ziyu Wang, Tom Schaul, et al. "Dueling network architectures for deep reinforcement learning". In: *arXiv preprint arXiv:1511.06581* (2015).
- Théophane Weber et al. "Imagination-augmented agents for deep reinforcement learning". In: *arXiv preprint arXiv:1707.06203* (2017).
- Yuhuai Wu et al. "Scalable trust-region method for deep reinforcement learning using Kronecker-factored approximation". In: *Advances in neural information processing systems*. 2017, pp. 5285–5294.