

Apprentissage par renforcement

Cours 6: Off-Policy Policy Gradients

Sylvain Lamprier

UE RLD - Master DAC

2020

Off-Policy Policy Gradients

Toutes les méthodes PG type TRPO, contrôlant les déplacements dans l'espace des politiques, utilisent une forme de off-policy employant des ratios d'IS.

Mais elle restent on-policy :

- ▶ Elles ne peuvent se servir des anciennes trajectoires que dans une région de confiance restreinte (peu "sample-efficient")

Importance Sampling pour réutiliser d'anciennes trajectoires :

$$\nabla_{\theta} J^{IS}(\theta) \approx \frac{1}{M} \sum_{\tau^{(i)}} \frac{\pi_{\theta}(\tau^{(i)})}{\pi_{\theta^{(i)}}(\tau^{(i)})} \left[R(\tau^{(i)}) \nabla_{\theta} \log \pi_{\theta}(\tau^{(i)}) \right]$$

Problème : Très forte variance dès lors que $\pi_{\theta^{(i)}}$ et π_{θ} trop différentes

Proposition : Weighted Importance Sampling pour contrôler la variance

$$J^{WIS}(\theta) = \frac{1}{Z} \sum_{\tau^{(i)}} w(\tau^{(i)}; \theta, \theta^{(i)}) R(\tau^{(i)})$$

avec : $w(\tau_i; \theta, \theta^{(i)}) = \frac{\pi_{\theta}(\tau^{(i)})}{\pi_{\theta^{(i)}}(\tau^{(i)})}$ et $Z = \sum_{\tau^{(i)}} w(\tau^{(i)}; \theta, \theta^{(i)})$

- ⇒ Objectif : limiter la variance en normalisant selon les écarts de politiques considérés
- ⇒ Estimateur biaisé (mais consistant : biais qui s'annule asymptotiquement)

On peut alors prouver le gradient [Doe+19] :

$$\nabla_{\theta} J^{WIS}(\theta) \approx \frac{1}{Z} \sum_{\tau^{(i)}} \nabla_{\theta} w(\tau_i; \theta, \theta^{(i)}) \left[R(\tau^{(i)}) - J^{WIS}(\theta) \right]$$

Off-Policy Policy Gradients

... Mais encore une forte variance. On aimerait se ramener à des rapports d'Importance Sampling sur des décisions uniques plutôt que sur des trajectoires entières.

Soit $d^\beta(s) = \lim_{t \rightarrow \infty} P(S_t = s | \beta)$ la distribution stationnaire des états selon la politique de "behavior" β , on considère en Off-Policy :

$$J(\theta) = \sum_{s \in \mathcal{S}} d^\beta(s) \sum_{a \in \mathcal{A}} Q^\pi(s, a) \pi_\theta(a|s) = \mathbb{E}_{s \sim d^\beta} \left[\sum_{a \in \mathcal{A}} Q^\pi(s, a) \pi_\theta(a|s) \right]$$

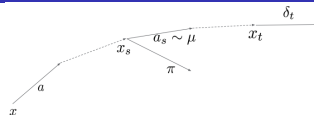
De nombreuses méthodes Off-Policy (Off-PAC [DWS12], RIS-Off-PAC [HC18], ACER [Wan+16], DDPG [Lil+15], etc.) considèrent l'approximation suivante :

$$\begin{aligned} \nabla_\theta J(\theta) &= \nabla_\theta \mathbb{E}_{s \sim d^\beta} \left[\sum_{a \in \mathcal{A}} Q^\pi(s, a) \pi_\theta(a|s) \right] \\ &= \mathbb{E}_{s \sim d^\beta} \left[\sum_{a \in \mathcal{A}} (Q^\pi(s, a) \nabla_\theta \pi_\theta(a|s) + \pi_\theta(a|s) \nabla_\theta Q^\pi(s, a)) \right] && \text{; Derivative product rule.} \\ &\stackrel{(i)}{\approx} \mathbb{E}_{s \sim d^\beta} \left[\sum_{a \in \mathcal{A}} Q^\pi(s, a) \nabla_\theta \pi_\theta(a|s) \right] && \text{; Ignore the red part : } \pi_\theta(a|s) \nabla_\theta Q^\pi(s, a). \\ &= \mathbb{E}_{s \sim d^\beta} \left[\sum_{a \in \mathcal{A}} \beta(a|s) \frac{\pi_\theta(a|s)}{\beta(a|s)} Q^\pi(s, a) \frac{\nabla_\theta \pi_\theta(a|s)}{\pi_\theta(a|s)} \right] \\ &= \mathbb{E}_\beta \left[\frac{\pi_\theta(a|s)}{\beta(a|s)} Q^\pi(s, a) \nabla_\theta \ln \pi_\theta(a|s) \right] && \text{; The blue part is the importance weight.} \end{aligned}$$

Bien que biaisé, [DWS12] montre que ce gradient permet de converger vers la politique optimale (localement) dans le cas tabulaire (gradient = 0 pour les mêmes optimas locaux).

Off-Policy Policy Gradients

Comment estimer efficacement Q^π en off-policy ?



Temporal Difference ok... :

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \delta_t \text{ avec } \delta_t = r_t + \gamma \mathbb{E}_{a \sim \pi} Q(s_{t+1}, a) - Q(s_t, a_t)$$

... mais propagation (trop) lente des valeurs

Comment mettre en place un algorithme plus efficace “multi-steps learning” ? :

$$Q(s, a) \leftarrow Q(s, a) +$$

$$\alpha \mathbb{E}_\mu \left[\sum_{t=0}^k \gamma^t \left(\prod_{i=0}^t c_i \right) (r_t + \gamma \mathbb{E}_\pi [Q(s_{t+1}, a_{t+1})] - Q(s_t, a_t)) \mid s_0 = s, a_0 = a \right]$$

Avec $c_i = \lambda$, on a un algorithme $Q(\lambda)$ classique

Problème : la trajectoire dépend de μ plutôt que de π

[Har+16] montre qu'en utilisant $\lambda \leq \frac{1-\gamma}{\gamma \|\mu - \pi\|_1}$, on converge vers la vraie valeur Q^π .

Mais on ne connaît pas $\|\mu - \pi\|_1 \Rightarrow$ ne fonctionne que si μ et π très proches.

Off-Policy Policy Gradients

Multi-steps learning (forward view, version tabulaire) :

$$Q(s, a) \leftarrow Q(s, a) + \alpha \mathbb{E}_{\mu} \left[\sum_{t=0}^k \gamma^t \left(\prod_{i=0}^t c_i \right) (r_t + \gamma \mathbb{E}_{\pi} [Q(s_{t+1}, a_{t+1})] - Q(s_t, a_t)) \mid s_0 = s, a_0 = a \right]$$

Propositions :

- ▶ Importance Sampling : $c_i = \frac{\pi(a_i | s_i)}{\mu(a_i | s_i)}$
 - 👍 Non biaisé
 - 👎 Variance très élevée (possiblement infinie) \Rightarrow Pas stable
- ▶ Tree backup TB(λ) [Pre00] : $c_i = \lambda \pi(a_i | s_i)$
 - ▶ Repondération des traces par leur probabilité dans la politique π
 - ▶ Atténuation exponentielle des importances des δ_t selon leur éloignement dans le temps
 - 👍 Fonctionne même pour des politiques μ et π éloignées
 - 👎 Coupures prématurées. Pas efficace lorsque π et μ sont proches
- ▶ Retrace [Mun+16] : $c_i = \lambda \min(1, \frac{\pi(a_i | s_i)}{\mu(a_i | s_i)})$
 - ▶ Mix des approches précédentes
 - 👍 Pas d'explosion de la variance
 - 👍 Pas de coupure prématurée ($\min(1, \frac{\pi(a_i | s_i)}{\mu(a_i | s_i)}) > \pi(a_i | s_i)$)
 - 👍 Garanties de convergence

Dans le cas approximé, définition récursive pour chaque (sous-) trace τ :

$$Q_{\tau}^{ret}(s_t, a_t) = r_t + \gamma c_{t+1} [Q_{\tau}^{ret}(s_{t+1}, a_{t+1}) - Q_w(s_{t+1}, a_{t+1})] + \gamma \mathbb{E}_a Q_w(s_{t+1}, a)$$

avec $Q_w(s, a)$ appris selon moindres carrés avec Q^{ret}

Off-Policy Policy Gradients : ACER

ACER[Wan+16] utilise Retrace pour l'estimation de Q et considère le gradient :

$$\begin{aligned} g &= \mathbb{E}_{\mu} \left[\frac{\pi_{\theta}(a|s)}{\mu(a|s)} Q^{\pi}(s, a) \nabla_{\theta} \ln \pi_{\theta}(a|s) \right] \\ &= \mathbb{E}_{\mu} \left[\min(c, \omega_t(a)) (Q^{\text{ret}}(s, a) - V_w(s)) \nabla_{\theta} \ln \pi_{\theta}(a|s) \right. \\ &\quad \left. + \mathbb{E}_{a' \sim \pi} \left[\max(0, \frac{\omega_t(a') - c}{\omega_t(a')} \right) (Q_w(s, a') - V_w(s)) \nabla_{\theta} \ln \pi_{\theta}(a'|s) \right] \right] \quad ; \text{ Avec } \omega_t(a) = \frac{\pi(a|S_t)}{\mu(a|S_t)} \end{aligned}$$

Où :

- ▶ Le premier terme “clippe” le ratio d'importance sampling pour réduire la variance.
 - ▶ Utilise Q^{ret} sur les états-actions des traces collectées selon μ .
- ▶ Le second terme corrige le biais engendré par cet ajustement.
 - ▶ Activé seulement si $w_t(a) > c$. Borné par 1
 - ▶ Lorsque $w_t(a)$ est trop grand, on utilise alors des actions échantillonnées selon π
 - ▶ On n'a pas de traces correspondant à ces actions : il est alors possible que l'on n'ait pas accès à des Q^{ret} correspondants
 - ▶ Utilisation d'une approximation réseau de neurones Q_w obtenue par minimisation moindres carrés selon Q^{ret}

Off-Policy Policy Gradients : ACER

ACER emploie une approche Trust-Region, mais contrairement à TRPO :

- ▶ Considère un pool de trajectoires issues de diverses politiques précédentes (pas seulement la dernière)
- ▶ Considère une contrainte KL avec une politique moyenne $\pi_{\theta_a}(\cdot|s) = f(\cdot|\phi_{\theta_a}(s))$, mise à jour régulièrement par : $\theta_a \leftarrow \alpha\theta_a + (1 - \alpha)\theta$
- ▶ Calcul du gradient selon $\phi_{\theta}(s)$ plutôt que selon θ :

$$\hat{g}^{acer}(s) = \min(c, \omega(s, a))(Q^{\text{ret}}(s, a) - V_w(s))\nabla_{\phi_{\theta}(s)} \ln \pi_{\theta}(a|s) + \\ \mathbb{E}_{a' \sim \pi} \left[\max(0, \frac{\omega(s, a') - c}{\omega(s, a')})(Q_w(s, a') - V_w(s))\nabla_{\phi_{\theta}(s)} \ln \pi_{\theta}(a'|s) \right]$$

- ▶ Recherche d'un vecteur proche de $\hat{g}^{acer}(s)$ qui n'implique pas un trop grand déplacement de politique selon s (recherche d'un vecteur le plus orthogonal à $k = \nabla_{\phi_{\theta}(s)} D_{KL}[f(\cdot|\phi_{\theta_a}(s))||f(\cdot|\phi_{\theta}(s))]$). Définition d'une contrainte linéaire pour tout s des trajectoires considérées :

$$z^* = \arg \min_z \frac{1}{2} \|\hat{g}^{acer}(s) - z\|_2^2 \\ \text{s.t. } \nabla_{\phi_{\theta}(s)} D_{KL}[f(\cdot|\phi_{\theta_a}(s))||f(\cdot|\phi_{\theta}(s))]]^T z \leq \delta$$

dont la solution qui satisfait les KKT est : $z^* = \hat{g}^{acer}(s) - \max(0; \frac{k^T \hat{g}^{acer}(s) - \delta}{\|k\|_2^2})k$

- ▶ Policy gradient considéré selon z^* (backpropagation chain-rule) : $z^* \nabla_{\theta} \phi_{\theta}(s)$

Off-Policy Policy Gradients : ACER

ACER est une méthode hybride on-policy/off-policy

- ▶ 1 passe On-Policy à chaque itération pour collecter de nouvelles trajectoires
- ▶ n passes Off-Policy pour exploiter les trajectoires collectées

Algorithm 1 ACER for discrete actions (master algorithm)

// Assume global shared parameter vectors θ and θ_v .

// Assume ratio of replay r .

repeat

 Call ACER on-policy, Algorithm 2.

$n \leftarrow \text{Poisson}(r)$

for $i \in \{1, \dots, n\}$ **do**

 Call ACER off-policy, Algorithm 2.

end for

until Max iteration or time reached.

Off-Policy Policy Gradients : ACER

Algorithm 2 ACER for discrete actions

Reset gradients $d\theta \leftarrow 0$ and $d\theta_v \leftarrow 0$.
Initialize parameters $\theta' \leftarrow \theta$ and $\theta'_v \leftarrow \theta_v$.
if not On-Policy **then**
 Sample the trajectory $\{x_0, a_0, r_0, \mu(\cdot|x_0), \dots, x_k, a_k, r_k, \mu(\cdot|x_k)\}$ from the replay memory.
else
 Get state x_0
end if
for $i \in \{0, \dots, k\}$ **do**
 Compute $f(\cdot|\phi_{\theta'}(x_i))$, $Q_{\theta'_v}(x_i, \cdot)$ and $f(\cdot|\phi_{\theta_a}(x_i))$.
 if On-Policy **then**
 Perform a_i according to $f(\cdot|\phi_{\theta'}(x_i))$
 Receive reward r_i and new state x_{i+1}
 $\mu(\cdot|x_i) \leftarrow f(\cdot|\phi_{\theta'}(x_i))$
 end if
 $\bar{\rho}_i \leftarrow \min \left\{ 1, \frac{f(a_i|\phi_{\theta'}(x_i))}{\mu(a_i|x_i)} \right\}$.
end for
 $Q^{ret} \leftarrow \begin{cases} 0 & \text{for terminal } x_k \\ \sum_a Q_{\theta'_v}(x_k, a) f(a|\phi_{\theta'}(x_k)) & \text{otherwise} \end{cases}$ Possiblement trajectoire incomplète
for $i \in \{k-1, \dots, 0\}$ **do**
 $Q^{ret} \leftarrow r_i + \gamma Q^{ret}$
 $V_i \leftarrow \sum_a Q_{\theta'_v}(x_i, a) f(a|\phi_{\theta'}(x_i))$
 Computing quantities needed for trust region updating:

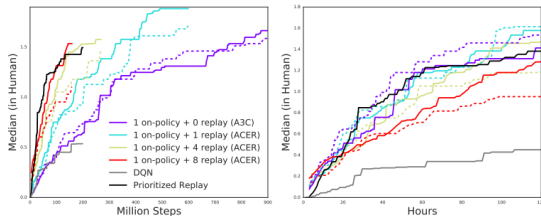
$$g \leftarrow \min \{c, \rho_i(a_i)\} \nabla_{\phi_{\theta'}(x_i)} \log f(a_i|\phi_{\theta'}(x_i)) (Q^{ret} - V_i)$$

$$+ \sum_a \left[1 - \frac{c}{\rho_i(a)} \right]_+ f(a|\phi_{\theta'}(x_i)) \nabla_{\phi_{\theta'}(x_i)} \log f(a|\phi_{\theta'}(x_i)) (Q_{\theta'_v}(x_i, a_i) - V_i)$$

$$k \leftarrow \nabla_{\phi_{\theta'}(x_i)} D_{KL} [f(\cdot|\phi_{\theta_a}(x_i)) \| f(\cdot|\phi_{\theta'}(x_i))]$$

 Accumulate gradients wrt θ' : $d\theta \leftarrow d\theta + \frac{\partial \phi_{\theta'}(x_i)}{\partial \theta'} \left(g - \max \left\{ 0, \frac{k^T g - \delta}{\|k\|_2^2} \right\} k \right)$
 Accumulate gradients wrt θ'_v : $d\theta_v \leftarrow d\theta_v + \nabla_{\theta'_v} (Q^{ret} - Q_{\theta'_v}(x_i, a))^2$
 Update Retrace target: $Q^{ret} \leftarrow \bar{\rho}_i (Q^{ret} - Q_{\theta'_v}(x_i, a_i)) + V_i$ Retrace update
end for
Perform asynchronous update of θ using $d\theta$ and of θ_v using $d\theta_v$.
Updating the average policy network: $\theta_a \leftarrow \alpha \theta_a + (1 - \alpha) \theta$

Off-Policy Policy Gradients : ACER

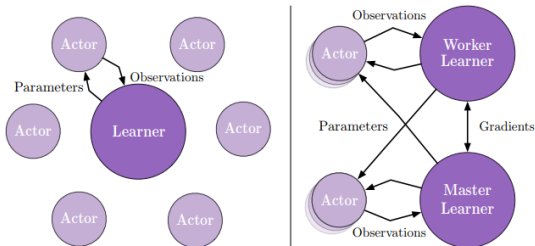


- ▶ Pas forcément plus rapide...
- ▶ ... Mais bien plus "sample efficient" !

Off-Policy Policy Gradients : Impala

Impala [Esp+18] : Similaire à ACER mais asynchrone

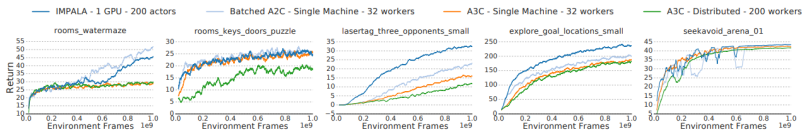
- ▶ A3C : Workers font un backward à la fin de chaque trajectoire et transmettent les gradients au master
 - 👉 Il peut y avoir un décalage entre la politique d'un acteur et les paramètres centraux auxquelles on ajoute le gradient
 - 👉 Optimisation sur trajectoires individuelles (GPU inefficaces)
- ▶ Impala : Workers Actor ne font qu'interagir avec l'environnement et transmettent trajectoires au master qui s'occupe de l'apprentissage
 - ▶ Du fait du lag entre le moment où la trajectoire est échantillonnée et la politique actuelle du learner, prise en compte de ratios IS comme dans ACER
 - ▶ Deux versions : une avec un seul learner, une avec plusieurs
 - ▶ Utilise V-trace plutôt que Retrace (équivalent mais travaille sur V plutôt que Q)



Off-Policy Policy Gradients : Impala

Architecture	CPU ^s	GPU ^s ¹	FPS ²	
Single-Machine			Task 1	Task 2
A3C 32 workers	64	0	6.5K	9K
Batched A2C (sync step)	48	0	9K	5K
Batched A2C (sync step)	48	1	13K	5.5K
Batched A2C (sync traj.)	48	0	16K	17.5K
Batched A2C (dyn. batch)	48	1	16K	13K
IMPALA 48 actors	48	0	17K	20.5K
IMPALA (dyn. batch) 48 actors ³	48	1	21K	24K
Distributed				
A3C	200	0	46K	50K
IMPALA	150	1		80K
IMPALA (optimised)	375	1		200K
IMPALA (optimised) batch 128	500	1		250K

¹ Nvidia P100 ² In frames/sec (4 times the agent steps due to action repeat). ³ Limited by amount of rendering possible on a single machine.



Off-Policy Policy Gradients

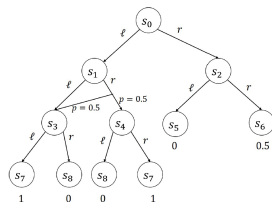
Gradient biaisé de [DWS12] ne permet pas d'assurer la convergence dans le cas général

Cas difficile pour les algos type Off-Pac :

Politique de collecte $\mu : \mu(a|s) = 0.5$ pour tout (s,a)

Classe de Politiques cibles $\Pi : \{\pi_\alpha, \alpha \in [0, 1]\}$,
où α définit la probabilité de choisir l dans les états 1
et 2. Pour tous les autres on a $\pi(a = l|s) = 1$.

Selon α , on a les valeurs V suivantes :
 $V^{\pi_\alpha}(s_0) = V^{\pi_\alpha}(s_1) = \frac{1+\alpha}{2}$, $V^{\pi_\alpha}(s_2) = \frac{1-\alpha}{2}$



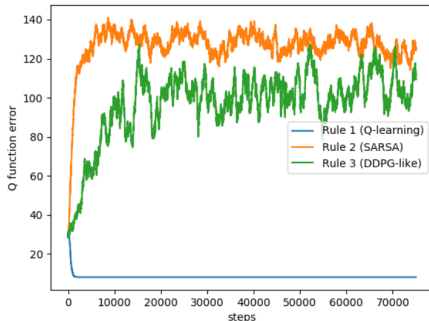
Clairement : $\alpha^* = 1$ (dans ce cas $V^{\pi_\alpha}(s_0) = 1$).

$$\begin{aligned}\text{Or : } g_{\text{offpac}} &= \mathbb{E}_{s \sim d^\mu(s), a \sim \mu(a|s)} \left[\frac{\nabla_\alpha \pi_\alpha(a|s)}{\mu(a|s)} Q^{\pi_\alpha}(s, a) \right] \\ &= d^\mu(s_1)(Q^{\pi_\alpha}(s_1, l) - Q^{\pi_\alpha}(s_1, r)) + d^\mu(s_2)(Q^{\pi_\alpha}(s_2, l) - Q^{\pi_\alpha}(s_2, r)) \\ &= 0.5(1 - 0.5) + 0.5(0 - 0.5) = 0\end{aligned}$$

Gradient nul quelle que soit la valeur de α !

Off-Policy Policy Gradients

Expérimentation sur GridWorld (petite carte) selon distribution β uniforme sur l'ensemble des états :



⇒ DDPG [Lil+15] pas vraiment Off-Policy

- Dû à l'approximation des Off-Policy Policy Gradients qui ignore les différences de distributions stationnaires sur les états

Off-Policy Policy Gradients

[Liu+18] propose de dépasser les limitations dues à l'approximation de [DWS12], en cherchant à caractériser $\frac{d\pi}{d\pi_0}$.

Pour l'évaluation d'une politique π à partir de traces collectées à partir d'une autre politique π_0 , la quantité $J(\theta)$ estimée par (Trajectory-Wise Importance Sampling) :

$$R_\pi = \mathbb{E}_{\tau \sim \pi_0} \left[\sum_{t=0}^T w_{0:T} \gamma^t r_t \right] \text{ avec } w_{0:T} = \prod_{i=0}^T \frac{\pi(a_i | s_i)}{\pi_0(a_i | s_i)}$$

En notant que $w_{0:t} = \prod_{i=0}^t \frac{\pi(a_i | s_i)}{\pi_0(a_i | s_i)} = \mathbb{E}_{\tau \sim \pi_0} [w_{0:T} | \tau_{0:t}]$ (Rao-Blackwellisation), on peut réduire la variance en considérant (Step-Wise Importance Sampling [Pre00]) :

$$R_\pi = \mathbb{E}_{\tau \sim \pi_0} \left[\sum_{t=0}^T w_{0:t} \gamma^t r_t \right]$$

[Liu+18] propose d'aller plus loin en considérant une formulation de $J(\theta)$ selon une distribution sur les états :

$$J(\theta) = \mathbb{E}_{s \sim d^\pi(s), a \sim \pi(a|s), s' \sim P(s'|s,a)} [R(s, a, s')], \text{ avec } d^\pi(s) = (1-\gamma) \sum_{t=0}^{\infty} \gamma^t d_t^\pi(s_t = s)$$

Selon des traces collectées par π_0 , on peut alors considérer l'estimateur :

$$R_\pi = \mathbb{E}_{s \sim d^{\pi_0}(s), a \sim \pi_0(a|s), s' \sim P(s'|s,a)} [w(s) \frac{\pi(a|s)}{\pi_0(a|s)} R(s, a, s')], \text{ avec } w(s) = \frac{d^\pi(s)}{d^{\pi_0}(s)}$$

Toute la difficulté réside alors dans l'estimation de $w(\cdot)$

Off-Policy Policy Gradients

Toute la difficulté réside alors dans l'estimation de $w_\theta(s) = \frac{d^\pi(s)}{d^{\pi_0}(s)} \forall s$
On commence par noter que, pour tout s' , on a :

$$d^\pi(s') = \gamma \sum_s d^\pi(s) \sum_a \pi(a|s) P(s'|s, a) + (1 - \gamma) P(s_0 = s')$$

Ou de manière équivalente, si on ajoute une transition fictive (de reward nul) (s_{-1}, a_{-1}, s_0) au temps -1 de chaque trajectoire :

$$\tilde{d}^\pi(s') \propto \sum_s \tilde{d}^\pi(s) \sum_a \pi(a|s) P(s'|s, a) \text{ avec } \tilde{d}^\pi(s) = (1 - \gamma) \sum_{t=-1}^{\infty} \gamma^{t+1} d_t^\pi(s_t = s)$$

Pour assurer cette relation, l'estimateur w_θ doit vérifier, pour tout $s' \in \mathcal{S}$:

$$\mathbb{E}_{(s,a) \sim \tilde{d}^{\pi_0}((s,a)|s')} [\Delta(w_\theta, s, a, s') | s'] = 0, \text{ avec } \Delta(w_\theta, s, a, s') = w_\theta(s) \frac{\pi(a|s)}{\pi_0(a|s)} - w_\theta(s')$$

$$\text{avec } \tilde{d}^{\pi_0}((s, a) | s') = \frac{\tilde{d}^{\pi_0}(s) \pi_0(a|s) P(s'|s, a)}{d^{\pi_0}(s')}$$

Problème : on ne connaît pas $\tilde{d}^{\pi_0}((s, a) | s')$

- dans des MDP de grande taille ou continus, peu d'observations de s'
- ⇒ Utilisation d'une fonction $f(s')$ pour exploiter la structure de l'espace d'états

Off-Policy Policy Gradients

Selon π_0 , l'estimateur w_θ doit alors vérifier, pour toute fonction f :

$$L(w_\theta, f) = \mathbb{E}_{(s,a,s') \sim \tilde{d}^{\pi_0}(s,a,s')} [\Delta(w_\theta, s, a, s') f(s')] = 0$$

$$\text{avec } \Delta(w_\theta, s, a, s') = w_\theta(s) \frac{\pi(a|s)}{\pi_0(a|s)} - w_\theta(s')$$

Soit : $w_\theta^* = \arg \min_{w_\theta} \max_{f \in \mathcal{F}} L(w_\theta / z_\theta, f)^2$, avec $z_\theta = \mathbb{E}_{s \sim \tilde{d}^{\pi_0}} [w_\theta(s)]$

- ▶ Normalisation par z_θ pour éviter les solutions triviales $w_\theta(s) = 0$
- ▶ $L(w_\theta / z_\theta, f) = 0$, avec f quelconque : condition nécessaire mais pas suffisante
- ▶ $L(w_\theta / z_\theta, \arg \max_f L(w_\theta / z_\theta, f)) = 0$: condition suffisante
- ▶ On limite f à une famille de fonctions \mathcal{F} qui permettent d'exploiter l'espace des états (peu d'observations de chaque (s, a) pour chaque s')

Off-Policy Policy Gradients

[Liu+18] considère la famille de fonctions dans la boule unitaire d'un RKHS (par exemple noyau gaussien) : $f(s) = \langle f, k(s, \cdot) \rangle_{\mathcal{H}}$, avec $\|f\|_{\mathcal{H}} \leq 1$. On a alors :

$$\begin{aligned} L(w_{\theta}, f) &= \mathbb{E}_{(s,a,s') \sim \tilde{d}^{\pi_{\mathbf{o}}}(s,a,s')} [\Delta(w_{\theta}, s, a, s') \langle f, k(s', \cdot) \rangle_{\mathcal{H}}] \\ &= \langle f, \mathbb{E}_{(s,a,s') \sim \tilde{d}^{\pi_{\mathbf{o}}}(s,a,s')} [\Delta(w_{\theta}, s, a, s') k(s', \cdot)] \rangle_{\mathcal{H}} \end{aligned}$$

Or pour $\mathcal{F} : \{f \in \mathcal{H} : \|f\|_{\mathcal{H}} \leq 1\}$, $\max_{f \in \mathcal{F}} \langle f, g \rangle_{\mathcal{H}} = \|g\|_{\mathcal{H}}$

On a alors une forme close pour :

$$\begin{aligned} \max_{f \in \mathcal{F}} L(w_{\theta}, f)^2 &= \|\mathbb{E}_{(s,a,s') \sim \tilde{d}^{\pi_{\mathbf{o}}}(s,a,s')} [\Delta(w_{\theta}, s, a, s') k(s', \cdot)]\|_{\mathcal{H}}^2 \\ &= \mathbb{E}_{(s,a,s') \sim \tilde{d}^{\pi_{\mathbf{o}}}} \mathbb{E}_{(\bar{s}, \bar{a}, \bar{s}') \sim \tilde{d}^{\pi_{\mathbf{o}}}} \Delta(w_{\theta}, s, a, s') \Delta(w_{\theta}, \bar{s}, \bar{a}, \bar{s}') \langle k(s', \cdot), k(\bar{s}', \cdot) \rangle_{\mathcal{H}} \end{aligned}$$

Off-Policy Policy Gradients

Algorithm 2 Main Algorithm (Discounted Reward Case)

Input: Transition data $\mathcal{D} = \{s_t, a_t, s'_t, r_t\}_t$ from the behavior policy π_0 ; a target policy π for which we want to estimate the expected reward. Denote by $\beta_{\pi/\pi_0}(a|s) = \pi(a|s)/\pi_0(a|s)$. Discount factor $\gamma \in (0, 1]$.

Augment the data with dummy data $\{s_{-1}, a_{-1}, s'_{-1}, r_{-1}\}$ for which $r_{-1} = 0$, $s'_{-1} = s_0$ and $\Delta(w; s_{-1}, a_{-1}, s'_{-1}) := 1 - w(s_0)$. Add them to \mathcal{D} to form an augmented dataset $\tilde{\mathcal{D}}$.

Initial the density ratio $w(s) = w_\theta(s)$ to be a neural network parameterized by θ .

for iteration = 1, 2, ... **do**

Randomly choose a batch $\mathcal{M} \subseteq \{1, \dots, n\}$ from the augmented transition data $\tilde{\mathcal{D}}$, by selecting time t with probability proportional to γ^{t+1} .

Update the parameter θ by $\theta \leftarrow \theta - \epsilon \nabla_\theta \hat{D}(w_\theta/z_{w_\theta})$, where

$$\hat{D}(w) = \frac{1}{|\mathcal{M}|} \sum_{i,j \in \mathcal{M}} \Delta(w, s_i, a_i, s'_i) \Delta(w, s_j, a_j, s'_j) k(s'_i, s'_j),$$

and z_{w_θ} is a normalization constant $z_{w_\theta} = \frac{1}{|\mathcal{M}|} \sum_{i \in \mathcal{M}} w_\theta(s_i)$.

end for

Output: Estimate the expected reward of π by $\hat{R}_\pi = \sum_{i=1}^n v_i r_i / \sum_{i=1}^n v_i$, where $v_i = w_\theta(s_i) \beta_{\pi/\pi_0}(a_i, s_i)$.

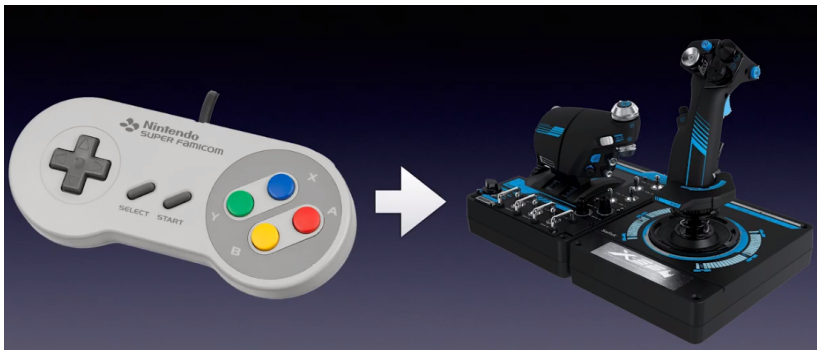
[Liu+19] propose un algo de policy gradient basé sur cette estimation de w (+ une augmentation de données pour garantir $\pi(a|s) = 0$ pour tout s, a tel que $\mu(a|s) = 0$) :

► $\nabla_\theta J(\theta) = \mathbb{E}_{s \sim d^\mu, a \sim \mu} \frac{d^\pi(s)}{d^\mu(s)} \frac{\pi(a|s)}{\mu(a|s)} \nabla_\theta \log \pi(a|s) Q^\pi$, avec $\frac{d^\pi(s)}{d^\mu(s)}$ donné par $w\theta(s)$

Actions Discrètes

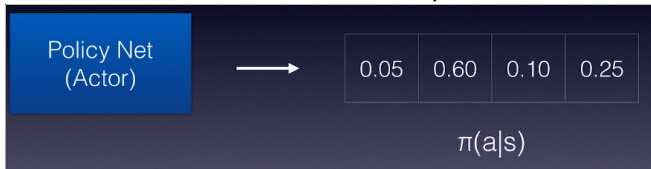


Actions Continues

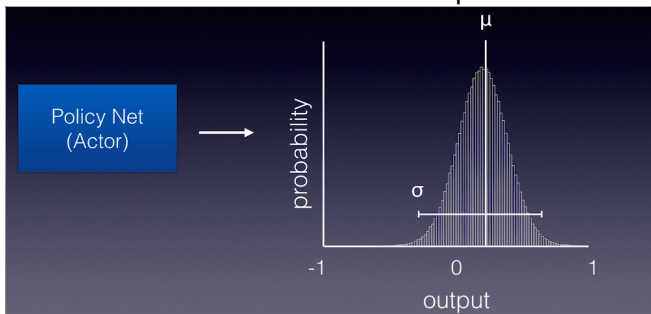


Policy Gradients for Continuous Action Spaces

Discrete Action Space



Continuous Action Space



Policy Gradients for Continuous Action Spaces

Dans le contexte des problèmes à actions continues, on a $\mathcal{A}(s) \subseteq \mathbb{R}^d$ pour tout s , avec d la dimension des actions considérées.

On définit alors souvent la politique comme une gaussienne multivariée :

$\pi_{\theta}(a|s) = \mathcal{N}(a; \mu_{\theta}(s), \Sigma_{\theta}(s))$, avec μ_{θ} un réseau produisant un vecteur de moyenne pour un état s . Deux possibilités pour $\Sigma_{\theta}(s)$:

- ▶ $\Sigma_{\theta}(s) = \alpha I$, où α est un vecteur de paramètres indépendants de s
- ▶ $\Sigma_{\theta}(s) = \sigma_{\theta}(s)^2 I$, où $\sigma_{\theta}(s)^2$ est un réseau (de paramètres souvent partagés avec μ_{θ}) produisant un vecteur de variance pour l'état s

L'ensemble des algos PG présentés précédemment peuvent alors s'étendre au cas continu (en remplaçant les \sum_a par des \int_a).

Notons néanmoins les trois méthodes populaires suivantes, spécifiques au cas des actions continues :

- ▶ DPG [Sil+14] : Deterministic Policy Gradient
- ▶ DDPG [Lil+15] : Deep Deterministic Policy Gradient
- ▶ QPROP [Gu+16] : Sample-Efficient Policy Gradient with an Off-Policy Critic

DPG : Deterministic Policy Gradient

Plutôt que considérer $a \sim \pi(a|s)$, DPG [Sil+14] définit $a = \mu(s)$ afin de limiter la variance des trajectoires.

Objectif :

$$\begin{aligned} J(\theta) &= \int_{s_0} p(s_0) \int_{s_1} p(s_1|\mu(s_0))(r(s_0, \mu(s_0), s_1) + \gamma \int_{s_2} p(s_2|\mu(s_1))(r(s_1, \mu(s_1), s_2) + \gamma \int_{s_3} \dots \\ &= \int_S p(s_0) Q(s_0, \mu(s_0)) ds_0 \end{aligned}$$

Gradient de J selon les paramètres de la politique (voir Preuve) :

$$\nabla_{\theta} J(\theta) = \int_S d^{\theta}(s) \nabla_a Q(s, a)|_{a=\mu(s)} \nabla_{\theta} \mu(s) ds$$

avec $d^{\pi}(s)$ la distribution discountée des futurs états :

$$d^{\pi}(s) = (1 - \gamma) \sum_{t=0}^{\infty} \gamma^t P(s_t = s | \mu)$$

où $P(s_t = s | \mu)$ dénote la probabilité d'être dans l'état s à l'étape t d'une trajectoire en suivant la politique μ .

DPG : Deterministic Policy Gradient

Puisque le cas déterministe est simplement un cas spécial du cas stochastique (Dirac centré sur $\mu(\cdot)$), il est possible d'utiliser tous les algos des PGs. Par exemple, un Actor-Critic (avec a_t l'action choisie à l'instant t) :

$$\delta_t = R_t + \gamma Q_w(s_{t+1}, a_{t+1}) - Q_w(s_t, a_t) \quad ; \text{TD error in SARSA}$$

$$w_{t+1} = w_t + \alpha_w \delta_t \nabla_w Q_w(s_t, a_t)$$

$$\theta_{t+1} = \theta_t + \alpha_\theta \nabla_a Q_w(s_t, a_t) \nabla_\theta \mu_\theta(s) |_{a=\mu_\theta(s)} \quad ; \text{Deterministic policy gradient theorem}$$

Cependant, on risque de s'enfermer rapidement dans des solutions sous-optimales. Afin d'éviter ce problème, DPG propose de considérer une version Off-Policy (avec remise à jour régulière de la politique β) :

$$\nabla_\theta J_\beta(\theta) = \mathbb{E}_{s \sim \rho^\beta} [\nabla_a Q_w(s, a) |_{a=\mu_\theta(s)} \nabla_\theta \mu_\theta(s)]$$

On note l'absence de ratio IS. Cela est dû à l'aspect déterministe de la méthode : on n'a plus de sampling selon la politique (on devrait néanmoins considérer un ratio IS sur les distributions des états pour être correct).

DDPG : Deep Deterministic Policy Gradient

Deep Deterministic Policy Gradient exploite les mécanismes d'Experience Replay et Target Network de DQN, en suivant les principes de DPG.

Algorithm 1 Deep Deterministic Policy Gradient

```
1: Input: initial policy parameters  $\theta$ , Q-function parameters  $\phi$ , empty replay buffer  $\mathcal{D}$ 
2: Set target parameters equal to main parameters  $\theta_{\text{targ}} \leftarrow \theta$ ,  $\phi_{\text{targ}} \leftarrow \phi$ 
3: repeat
4:   Observe state  $s$  and select action  $a = \text{clip}(\mu_{\theta}(s) + \epsilon, a_{\text{Low}}, a_{\text{High}})$ , where  $\epsilon \sim \mathcal{N}$ 
5:   Execute  $a$  in the environment
6:   Observe next state  $s'$ , reward  $r$ , and done signal  $d$  to indicate whether  $s'$  is terminal
7:   Store  $(s, a, r, s', d)$  in replay buffer  $\mathcal{D}$ 
8:   If  $s'$  is terminal, reset environment state.
9:   if it's time to update then
10:     for however many updates do
11:       Randomly sample a batch of transitions,  $B = \{(s, a, r, s', d)\}$  from  $\mathcal{D}$ 
12:       Compute targets
```

$$y(r, s', d) = r + \gamma(1 - d)Q_{\phi_{\text{targ}}}(s', \mu_{\theta_{\text{targ}}}(s'))$$

```
13:   Update Q-function by one step of gradient descent using
```

$$\nabla_{\phi} \frac{1}{|B|} \sum_{(s, a, r, s', d) \in B} (Q_{\phi}(s, a) - y(r, s', d))^2$$

```
14:   Update policy by one step of gradient ascent using
```

$$\nabla_{\theta} \frac{1}{|B|} \sum_{s \in B} Q_{\phi}(s, \mu_{\theta}(s))$$

```
15:   Update target networks with
```

$$\begin{aligned}\phi_{\text{targ}} &\leftarrow \rho \phi_{\text{targ}} + (1 - \rho) \phi \\ \theta_{\text{targ}} &\leftarrow \rho \theta_{\text{targ}} + (1 - \rho) \theta\end{aligned}$$

```
16:   end for
17: end if
18: until convergence
```

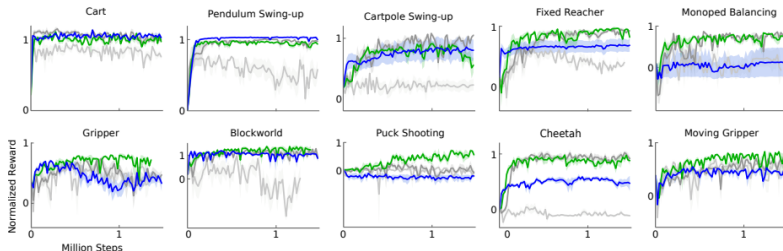
Ajout d'un bruit gaussien pour exploration + clip pour rester dans des valeurs admissibles

Utilisation de réseaux cible (à la fois pour Q et pour μ) pour le calcul de la cible de Q

Gradient similaire à DPG, selon les transitions du batch

Mise à jour "soft" des paramètres des réseaux cible

DDPG : Deep Deterministic Policy Gradient



- ▶ Gris clair : uniquement batch normalization sur les couches des réseaux
- ▶ Gris foncé : uniquement target networks
- ▶ Vert : avec target networks et batch normalization
- ▶ Bleu : comme vert mais avec pixels comme entrée (plutôt qu'une description spécifique au problème)

⇒ Target Networks essentiels pour stabiliser l'apprentissage

A noter les extensions :

- ▶ TD3 [FHM18] : Utilisation de deux réseaux Q pour éviter sa sur-estimation
- ▶ D4G [BM+18] : Prioritized Replay + Distributional Critic (e.g., $Q_w(s, a)$ issu d'une mixture de gaussiennes pour apprentissage plus stable)

Q-Prop : Policy Gradient with an Off-Policy Critic

Constat :

- ▶ Méthodes du type DDPG ne reposent pas sur des estimateurs de gradient type REINFORCE qui sont sujets à une forte variance
- ▶ Peuvent être apprises sur des données off-policy, ce qui les rend bien plus "sample-efficient"...
- ▶ ... Mais les estimateurs de PG qu'ils utilisent sont biaisés, ce qui rend difficile l'analyse de leur convergence et de leur stabilité

Proposition de Q-Prop : Allier les avantages des deux types de méthodes :

$$\begin{aligned}\nabla_{\theta} J(\theta) &= \mathbb{E}_{d^{\pi}, \pi} \left[\nabla_{\theta} \log \pi_{\theta}(a_t | s_t) \left(\hat{A}(s_t, a_t) - \bar{A}(s_t, a_t) \right) \right] \\ &\quad + \mathbb{E}_{d^{\pi}, \pi} \left[\nabla_{\theta} \log \pi_{\theta}(a_t | s_t) \bar{A}(s_t, a_t) \right] \\ &\text{avec } \bar{A}(s_t, a_t) = \bar{Q}_w(s_t, a_t) - \mathbb{E}_{a \sim \pi(a | s_t)} [\bar{Q}_w(s_t, a)] \\ &\text{et } \bar{Q}_w(s_t, a_t) = Q_w(s_t, \mu_{\theta}(s_t)) + \nabla_a Q_w(s_t, a)|_{a=\mu_{\theta}(s_t)} (a_t - \mu_{\theta}(s_t)) \\ &\quad (\text{expansion de Taylor d'ordre 1 de } Q_w \text{ en partant de } \mu_{\theta}(s_t) = \mathbb{E}_{a \sim \pi(a | s_t)}[a])\end{aligned}$$

On a alors (Preuve) :

$$\begin{aligned}\nabla_{\theta} J(\theta) &= \mathbb{E}_{d^{\pi}, \pi} \left[\nabla_{\theta} \log \pi_{\theta}(a_t | s_t) \left(\hat{A}(s_t, a_t) - \nabla_a Q_w(s_t, a)|_{a=\mu_{\theta}(s_t)} (a_t - \mu_{\theta}(s_t)) \right) \right] \\ &\quad + \mathbb{E}_{d^{\pi}} \left[\nabla_a Q_w(s_t, a)|_{a=\mu_{\theta}(s_t)} \nabla_{\theta} \mu_{\theta}(s_t) \right] \text{ avec } \mu_{\theta}(s_t) = \mathbb{E}_{a \sim \pi(a | s_t)}[a]\end{aligned}$$

Q-Prop : Policy Gradient with an Off-Policy Critic

Après l'introduction d'un coefficient $\eta(s_t)$ qui ne biaise pas le gradient mais vise à réduire la variance, on a :

$$\begin{aligned}\nabla_{\theta} J(\theta) = & \underbrace{\mathbb{E}_{d^{\pi}} [\eta(s_t) \nabla_a Q_w(s_t, a)|_{a=\mu_{\theta}(s_t)} \nabla_{\theta} \mu_{\theta}(s_t)]}_{\text{gradient moyen, équivalent à DDPG}} \\ & + \underbrace{\mathbb{E}_{d^{\pi}, \pi} \left[\nabla_{\theta} \log \pi_{\theta}(a_t | s_t) \left(\hat{A}(s_t, a_t) - \eta(s_t) \nabla_a Q_w(s_t, a)|_{a=\mu_{\theta}(s_t)} (a_t - \mu_{\theta}(s_t)) \right) \right]}_{\text{Prise en compte de l'écart avec l'estimateur d'avantage}}\end{aligned}$$

- ⇒ Si les anciennes politiques couvrent suffisamment l'espace des transitions, on peut évaluer la critique Off-Policy (comme dans tous les Actor-Critic)
- ⇒ Q-Prop se sert de cette critique Off-Policy à la façon de DDPG (Off-Policy) et débiaise selon le second terme d'optimisation On-Policy (avec le terme d'optimisation Off-Policy comme baseline) : gradient sans biais

La version conservatrice de Q-Prop utilise : $\eta(s_t) = 1$ si $\hat{A}(s_t, a_t) \bar{A}(s_t, a_t) > 0$, 0 sinon. Cela permet de limiter les prises en compte de gradient selon la critique aux seuls exemples où l'observation d'avantage et son estimation sont au moins d'accord sur le signe (action à favoriser ou non).

Q-Prop : Policy Gradient with an Off-Policy Critic

Algorithm 1 Adaptive Q-Prop

```
1: Initialize  $w$  for critic  $Q_w$ ,  $\theta$  for stochastic policy  $\pi_\theta$ , and replay buffer  $\mathcal{R} \leftarrow \emptyset$ .
2: repeat
3:   for  $e = 1, \dots, E$  do ▷ Collect  $E$  episodes of on-policy experience using  $\pi_\theta$ 
4:      $s_{0,e} \sim p(s_0)$ 
5:     for  $t = 0, \dots, T-1$  do
6:        $\mathbf{a}_{t,e} \sim \pi_\theta(\cdot | s_{t,e}), s_{t+1,e} \sim p(\cdot | s_{t,e}, \mathbf{a}_{t,e}), r_{t,e} = r(s_{t,e}, \mathbf{a}_{t,e})$ 
7:       Add batch data  $\mathcal{B} = \{s_{0:T-1,e}, \mathbf{a}_{0:T-1,e}, r_{0:T-1,e}\}$  to replay buffer  $\mathcal{R}$ 
8:       Take  $E \cdot T$  gradient steps on  $Q_w$  using  $\mathcal{R}$  and  $\pi_\theta$  ← Off-Policy Critic
9:       Fit  $V_\phi(s_t)$  using  $\mathcal{B}$ 
10:      Compute  $\hat{A}_{t,e}$  using GAE( $\lambda$ ) and  $\bar{A}_{t,e} = \nabla_{\mathbf{a}} Q_w(s_t, \mathbf{a})|_{\mathbf{a}=\mu_\theta(s_t)}(\mathbf{a}_t - \mu_\theta(s_t))$ .
11:      Set  $\eta_{t,e}$ 
12:      Compute and center the learning signals  $l_{t,e} = \hat{A}_{t,e} - \eta_{t,e} \bar{A}_{t,e}$  ← Off-Policy Actor
13:      Compute  $\nabla_\theta J(\theta) \approx \frac{1}{ET} \sum_e \sum_t \nabla_\theta \log \pi_\theta(\mathbf{a}_{t,e} | s_{t,e}) l_{t,e} + \eta_{t,e} \nabla_{\mathbf{a}} Q_w(s_{t,e}, \mathbf{a})|_{\mathbf{a}=\mu_\theta(s_{t,e})} \nabla_\theta \mu_\theta(s_{t,e})$ 
14:      Take a gradient step on  $\pi_\theta$  using  $\nabla_\theta J(\theta)$  ← On-Policy Actor (optionally with a trust-region constraint using  $\mathcal{B}$ )
15: until  $\pi_\theta$  converges.
```

Où Q_w est évalué selon le replay buffer avec :

$$w = \arg \min_w \mathbb{E}_{s_t \sim p_\beta(\cdot), \mathbf{a}_t \sim \beta(\cdot | s_t)} [(r(s_t, \mathbf{a}_t) + \gamma \mathbb{E}_\pi [Q'(s_{t+1}, \mathbf{a}_{t+1})] - Q_w(s_t, \mathbf{a}_t))^2].$$

- ▶ Q-Prop très efficace quand la critique a le temps d'être bien apprise entre chaque simulation (dans ce cas, bien meilleur que DDPG). Sinon, si simulations très rapides, équivalent à DDPG et moins bon que TRPO
- ▶ Q-Prop bien plus robuste à une mauvaise critique que les méthodes pures off-policy actor-critic telles que DDPG. Améliorations possibles en utilisant $\text{Retrace}(\lambda)$

Soft Actor-Critic (SAC)

Soft Actor-Critic (SAC) [Haa+18b] étend l'algorithme Soft Q-Learning [Haa+17] qui considère une politique stochastique maximisant l'entropie selon les états (pour l'exploration) :

$$\pi_{MaxEnt}^* = \arg \max_{\pi} \sum_{t=1}^T \mathbb{E}_{(s_t, a_t) \sim \rho_{\pi_{\theta}}} [r(s_t, a_t) + \alpha \mathcal{H}(\pi_{\theta}(\cdot | s_t))]$$

Alors que Soft Q-Learning considère une politique de maximum d'entropie $\pi(a|s) \propto \exp(Q_{\text{soft}}^{\pi}(s, a))$ avec

- ▶ $Q_{\text{soft}}^{\pi}(s_t, a_t) = r_t + \gamma \mathbb{E}_{s_{t+1} | s_t, a_t} [V_{\text{soft}}^{\pi}(s_{t+1})]$
- ▶ $V_{\text{soft}}^{\pi}(s_t) = \mathbb{E}_{a_t | s_t} [Q_{\text{soft}}^{\pi}(s_t, a_t) - \alpha \log \pi(a_t | s_t)]$

Soft Actor-Critic (SAC) s'intéresse à chercher une politique (actor) proche de cette distribution de maximum d'entropie selon Q_{soft} (critic). À chaque itération, déplacement de la politique selon :

$$\pi_{\text{new}} = \arg \min_{\pi' \in \Pi} D_{\text{KL}} \left(\pi'(\cdot | s_t) \parallel \frac{\exp(Q_{\text{soft}}^{\pi_{\text{old}}} (s_t, \cdot))}{Z^{\pi_{\text{old}}} (s_t)} \right)$$

avec $Q_{\text{soft}}^{\pi_{\text{old}}}$ la valeur Q_{soft} calculée sur la politique π_{old} et $Z^{\pi_{\text{old}}}(s_t)$ la fonction de partition d'une distribution proportionnelle à $\exp(Q_{\text{soft}}^{\pi_{\text{old}}}(s, a))$. Notons que cette fonction de partition (intractable) ne dépend pas de la politique à optimiser et n'intervient alors pas dans le gradient.

Soft Actor-Critic (SAC)

Pour V_{soft} et Q_{soft} , on considère des approximations optimisées selon les coûts :

$$J_V(\psi) = \mathbb{E}_{\mathbf{s}_t \sim D} \left[\frac{1}{2} \left(V_\psi(\mathbf{s}_t) - \mathbb{E}_{\mathbf{a}_t \sim \pi_\phi} [Q_\theta(\mathbf{s}_t, \mathbf{a}_t) - \alpha \log \pi_\phi(\mathbf{a}_t | \mathbf{s}_t)] \right)^2 \right]$$

$$J_Q(\theta) = \mathbb{E}_{(\mathbf{s}_t, \mathbf{a}_t) \sim D} \left[\frac{1}{2} \left(Q_\theta(\mathbf{s}_t, \mathbf{a}_t) - \hat{Q}(\mathbf{s}_t, \mathbf{a}_t) \right)^2 \right]$$

avec $\hat{Q}(\mathbf{s}_t, \mathbf{a}_t) = r(\mathbf{s}_t, \mathbf{a}_t) + \gamma \mathbb{E}_{\mathbf{s}_{t+1} | \mathbf{s}_t, \mathbf{a}_t} [V_{\hat{\psi}}(\mathbf{s}_{t+1})]$, où $V_{\hat{\psi}}(\mathbf{s}_{t+1})$ est un réseau cible mis à jour périodiquement en fonction de V_ψ (moyenne exponentielle)

Pour la politique on considère :

$$\begin{aligned} J_\pi(\phi) &= \mathbb{E}_{\mathbf{s}_t \sim D} \left[D_{\text{KL}} \left(\pi_\phi(\cdot | \mathbf{s}_t) \middle| \frac{\exp(Q_\theta(\mathbf{s}_t, \cdot))}{Z_\theta(\mathbf{s}_t)} \right) \right] \\ &= \mathbb{E}_{\mathbf{s}_t \sim D} \left[\mathbb{E}_{\mathbf{a}_t \sim \pi_\phi(\cdot | \mathbf{s}_t)} [\log \pi_\phi(\mathbf{a}_t | \mathbf{s}_t) - Q_\theta(\mathbf{s}_t, \mathbf{a}_t)] + \log Z_\theta(\mathbf{s}_t) \right] \end{aligned}$$

Plutôt que d'utiliser le log-trick pour optimiser selon une action échantillonnée (REINFORCE assez inefficace dans le cas continu), on considère une re-paramétrisation $\mathbf{a}_t = f_\phi(\epsilon_t; \mathbf{s}_t)$, avec $\epsilon_t \sim p(\epsilon_t)$, tel que pour tout h $\mathbb{E}_{\mathbf{a}_t | \mathbf{s}_t} [h(\mathbf{s}_t, \mathbf{a}_t)] = \mathbb{E}_{\epsilon_t \sim \mathcal{N}(0,1)} [h(\mathbf{s}_t, f_\phi(\epsilon_t; \mathbf{s}_t))]$ (**Reparameterization-trick**).

Soft Actor-Critic (SAC)

⇒ Re-paramétrisation permet de transférer la stochasticité sur un bruit blanc indépendant des paramètres.

Par exemple si on considère $\pi_\phi(\cdot | \mathbf{s}_t) = \mathcal{N}(\mu_\phi(\mathbf{s}_t), \sigma_\phi(\mathbf{s}_t)^2)$, on peut choisir $\epsilon_t \sim \mathcal{N}(0, 1)$ et définir $\mathbf{a}_t = f_\phi(\epsilon_t; \mathbf{s}_t) = \epsilon_t \times \sigma_\phi(\mathbf{s}_t) + \mu_\phi(\mathbf{s}_t)$ (génération d'une VA $X \sim \mathcal{N}(\mu, \sigma^2)$ se fait en prenant $X = \sigma \times \epsilon + \mu$, avec ϵ issu d'une loi normale standard). On peut alors calculer le gradient de manière classique, en samplant ce bruit selon une gaussienne standard.

On a alors :

$$J_\pi(\phi) = \mathbb{E}_{\mathbf{s}_t \sim \mathcal{D}, \epsilon_t \sim \mathcal{N}(0,1)} [\log \pi_\phi(f_\phi(\epsilon_t; \mathbf{s}_t) | \mathbf{s}_t) - Q_\theta(\mathbf{s}_t, f_\phi(\epsilon_t; \mathbf{s}_t))] + \mathbb{E}_{\mathbf{s}_t \sim \mathcal{D}} [\log Z_\theta(\mathbf{s}_t)]$$

Le gradient que l'on considère est donc :

$$\begin{aligned} \hat{\nabla}_\phi J_\pi(\phi) = \mathbb{E}_{\mathbf{s}_t \sim \mathcal{D}, \epsilon_t \sim \mathcal{N}(0,1)} & \left[\nabla_\phi \log \pi_\phi(\mathbf{a}_t | \mathbf{s}_t) \Big|_{\mathbf{a}_t = f_\phi(\epsilon_t; \mathbf{s}_t)} \right. \\ & + \nabla_{\mathbf{a}_t} \log \pi_\phi(\mathbf{a}_t | \mathbf{s}_t) \Big|_{\mathbf{a}_t = f_\phi(\epsilon_t; \mathbf{s}_t)} \nabla_\phi f_\phi(\epsilon_t; \mathbf{s}_t) \\ & \left. - \nabla_{\mathbf{a}_t} Q(\mathbf{s}_t, \mathbf{a}_t) \Big|_{\mathbf{a}_t = f_\phi(\epsilon_t; \mathbf{s}_t)} \nabla_\phi f_\phi(\epsilon_t; \mathbf{s}_t) \right] \end{aligned}$$

*Soft Actor-Critic (SAC)

Algorithm 1 Soft Actor-Critic1: Input: initial policy parameters θ , Q-function parameters ϕ_1, ϕ_2 , V-function parameters ψ , empty replay buffer \mathcal{D}

2: Set target parameters equal to main parameters $\psi_{\text{targ}} \leftarrow \psi$

3: repeat

- 4: Observe state s and select action $a \sim \pi_\theta(\cdot|s)$

5: Execute a in the environment

- 6: Observe next state s' , reward r , and done signal d to indicate whether s' is terminal

7: Store (s, a, r, s', d) in replay buffer \mathcal{D}

8: If s' is terminal, reset environment state.

```

9:   if it's time to update then

```

```
10:   for  $j$  in range(however many updates) do
```

11: Randomly sample a batch of transitions, $B = \{(s, a, r, s', d)\}$ from \mathcal{D}

12: Compute targets for Q and V functions:

$$y_v(s) = \min_{i=1,2} Q_{\phi_i}(s, \tilde{a}) - \alpha \log \pi_{\theta}(\tilde{a}|s), \quad \tilde{a} \sim \pi_{\theta}(\cdot|s)$$

2 réseaux Q pour éviter les sur-estimations

13: Update Q-functions by one step of gradient descent using

$$\nabla_{\phi_i} \frac{1}{|B|} \sum_{(s,a,r,s',d) \in B} (Q_{\phi_i}(s,a) - y_q(r,s',d))^2 \quad \text{for } i = 1, 2$$

14: Update V-function by one step of gradient descent using

$$\nabla_{\psi} \frac{1}{|B|} \sum_{s \in B} (V_{\psi}(s) - y_v(s))^2$$

15: Update policy by one step of gradient ascent using

$$\nabla_{\theta} \frac{1}{|B|} \sum_{s \in B} \left(Q_{\phi,1}(s, \tilde{a}_{\theta}(s)) - \log \pi_{\theta}(\tilde{a}_{\theta}(s) | s) \right),$$

where $\tilde{a}_\theta(s)$ is a sample from $\pi_\theta(\cdot|s)$ which is differentiable wrt θ via the reparametrization trick.

16: Update target value network with

$$\psi_{\text{targ}} \leftarrow \rho \psi_{\text{targ}} + (1 - \rho) \psi$$

17: end for

18: end if

19: **until** convergence

*Soft Actor-Critic (SAC)

Une difficulté de SAC est de définir un paramètre α efficace.

[Haa+18a] propose de l'ajuster automatiquement, en considérant le problème sous contrainte :

$$\max_{\pi_0, \dots, \pi_T} \mathbb{E} \left[\sum_{t=0}^T r(s_t, a_t) \right] \text{ s.t. } \forall t, \mathcal{H}(\pi_t) \geq \mathcal{H}_0$$

avec \mathcal{H}_0 un seuil d'entropie à assurer. Cela revient à considérer :

Algorithm 1 Soft Actor-Critic

Input: θ_1, θ_2, ϕ	▷ Initial parameters
$\theta_1 \leftarrow \theta_1, \theta_2 \leftarrow \theta_2$	▷ Initialize target network weights
$\mathcal{D} \leftarrow \emptyset$	▷ Initialize an empty replay pool
for each iteration do	
for each environment step do	
$\mathbf{a}_t \sim \pi_\phi(\mathbf{a}_t \mathbf{s}_t)$	▷ Sample action from the policy
$\mathbf{s}_{t+1} \sim p(\mathbf{s}_{t+1} \mathbf{s}_t, \mathbf{a}_t)$	▷ Sample transition from the environment
$\mathcal{D} \leftarrow \mathcal{D} \cup \{(\mathbf{s}_t, \mathbf{a}_t, r(\mathbf{s}_t, \mathbf{a}_t), \mathbf{s}_{t+1})\}$	▷ Store the transition in the replay pool
end for	
for each gradient step do	
$\theta_i \leftarrow \theta_i - \lambda_Q \nabla_{\theta_i} J_Q(\theta_i)$ for $i \in \{1, 2\}$	▷ Update the Q-function parameters
$\phi \leftarrow \phi - \lambda_\pi \nabla_\phi J_\pi(\phi)$	▷ Update policy weights
$\alpha \leftarrow \alpha - \lambda \nabla_\alpha J(\alpha)$	▷ Adjust temperature
$\bar{\theta}_i \leftarrow \tau \theta_i + (1 - \tau) \theta_i$ for $i \in \{1, 2\}$	▷ Update target network weights
end for	
Output: θ_1, θ_2, ϕ	▷ Optimized parameters

avec $J(\alpha) = \mathbb{E}_{\mathbf{a}_t \sim \pi_t} [-\alpha \log \pi_t(\mathbf{a}_t | \mathbf{s}_t) - \alpha \mathcal{H}_0]$, le coût d'optimisation de α , obtenu selon le Lagrangien (α est le coefficient de Lagrange).

On note que dans cette version V est défini de manière implicite plutôt que de requérir un réseau indépendant de Q .

- ▶ Sergey Levine (UC Berkeley, Spring 2017)
- ▶ Daniel Takeshi :
<https://danieltakeshi.github.io/2017/03/28/going-deeper-into-reinforcement-learning-fundamentals-o>
- ▶ Jonathan Hui : https://medium.com/@jonathan_hui/rl-deep-reinforcement-learning-series-833319a95530
- ▶ Olivier Sigaud, Reinforcement Learning Class :
<http://pages.isir.upmc.fr/~sigaud/teach/ofp.pdf>
- ▶ Lilian Weng : <https://lilianweng.github.io/lil-log/2018/04/08/policy-gradient-algorithms.html>
- ▶ Felix Yu :
<https://flyyufelix.github.io/2017/10/12/dqn-vs-pg.html>

- ▶ Joshua Achiam :
http://rail.eecs.berkeley.edu/deeprlcourse-fa17/f17docs/lecture_13_advanced_pg.pdf
- ▶ Nathan Ratliff : http://ipvs.informatik.uni-stuttgart.de/mlr/wp-content/uploads/2015/01/mathematics_for_intelligent_systems_lecture12_notes_I.pdf
- ▶ OpenAI : <https://spinningup.openai.com/en/latest/algorithms/trpo.html>

References I

- [BM+18] Gabriel Barth-Maroon et al. « Distributional Policy Gradients ». In : *International Conference on Learning Representations*. 2018.
- [Doe+19] Andreas Doerr et al. « Trajectory-Based Off-Policy Deep Reinforcement Learning ». In : *arXiv preprint arXiv :1905.05710* (2019).
- [DWS12] Thomas Degris, Martha White et Richard S Sutton. « Off-policy actor-critic ». In : *arXiv preprint arXiv :1205.4839* (2012).
- [Esp+18] Lasse Espeholt et al. « IMPALA : Scalable Distributed Deep-RL with Importance Weighted Actor-Learner Architectures ». In : *CoRR* abs/1802.01561 (2018). arXiv : 1802.01561.
- [FHM18] Scott Fujimoto, Herke van Hoof et David Meger. « Addressing Function Approximation Error in Actor-Critic Methods ». In : *CoRR* abs/1802.09477 (2018). arXiv : 1802.09477.
- [Gu+16] Shixiang Gu et al. « Q-Prop : Sample-Efficient Policy Gradient with An Off-Policy Critic ». In : *CoRR* abs/1611.02247 (2016). arXiv : 1611.02247.
- [Haa+17] Tuomas Haarnoja et al. « Reinforcement Learning with Deep Energy-Based Policies ». In : *CoRR* abs/1702.08165 (2017). arXiv : 1702.08165.

References II

- [Haa+18a] Tuomas Haarnoja et al. « Soft Actor-Critic Algorithms and Applications ». In : *CoRR* abs/1812.05905 (2018). arXiv : 1812.05905.
- [Haa+18b] Tuomas Haarnoja et al. « Soft Actor-Critic : Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor ». In : *CoRR* abs/1801.01290 (2018). arXiv : 1801.01290.
- [Har+16] Anna Harutyunyan et al. « Q(\$\lambda\$) with Off-Policy Corrections ». In : *CoRR* abs/1602.04951 (2016). arXiv : 1602.04951.
- [HC18] Mahammad Humayoo et Xueqi Cheng. « Relative Importance Sampling For Off-Policy Actor-Critic in Deep Reinforcement Learning ». In : *arXiv preprint arXiv :1810.12558* (2018).
- [Lil+15] Timothy P Lillicrap et al. « Continuous control with deep reinforcement learning ». In : *arXiv preprint arXiv :1509.02971* (2015).
- [Liu+18] Qiang Liu et al. « Breaking the curse of horizon : Infinite-horizon off-policy estimation ». In : *Advances in Neural Information Processing Systems*. 2018, p. 5356-5366.
- [Liu+19] Yao Liu et al. « Off-Policy Policy Gradient with State Distribution Correction ». In : *arXiv preprint arXiv :1904.08473* (2019).

References III

- [Mun+16] Rémi Munos et al. « Safe and efficient off-policy reinforcement learning ». In : *Advances in Neural Information Processing Systems*. 2016, p. 1054-1062.
- [Pre00] Doina Precup. « Eligibility traces for off-policy policy evaluation ». In : *Computer Science Department Faculty Publication Series* (2000), p. 80.
- [Sil+14] David Silver et al. « Deterministic policy gradient algorithms ». In : 2014.
- [Wan+16] Ziyu Wang et al. « Sample efficient actor-critic with experience replay ». In : *arXiv preprint arXiv :1611.01224* (2016).