

Policy Gradient

Clément Rambour

Plan

- 1 Introduction
- 2 Différences Finies
- 3 *policy gradient*
- 4 Monte Carlo Policy gradient
- 5 Actor Critic Policy Gradient
- 6 Bibliographie

Cours précédents

- Estimation de la valeur par échantillonnage
- Deux grande familles d'approches :
 - MC : approximation de la valeur par moyenne des retour
 - TD : Minimisation de l'erreur de Bellman
- Contrôle : estimation de la fonction de valeur état-action
- Valeurs stockées dans la table Q

Cours précédents

- Approximation de la fonction de valeur par une fonction de paramètre θ :

$$Q(s, a) \simeq f_{\theta}(s, a)$$

- Optimisation de f_{θ} par descente de gradient
- Loss Mean Square Error
 - MC : par rapport au retour
 - TD : par rapport à la TD-target Q-learning
- Deep Q Networks : $f_{\theta} \rightarrow$ réseau de neurones

Aujourd'hui

Motivation :

- Plutôt que de passer par une fonction décrivant le potentiel de chaque état/action de l'agent, pourquoi ne pas apprendre directement la politique ?
- Politique donnée par une distribution paramétrée par θ : $\pi_{\theta}(s, a)$

optimisation de la valeur vs. optimisation de la politique

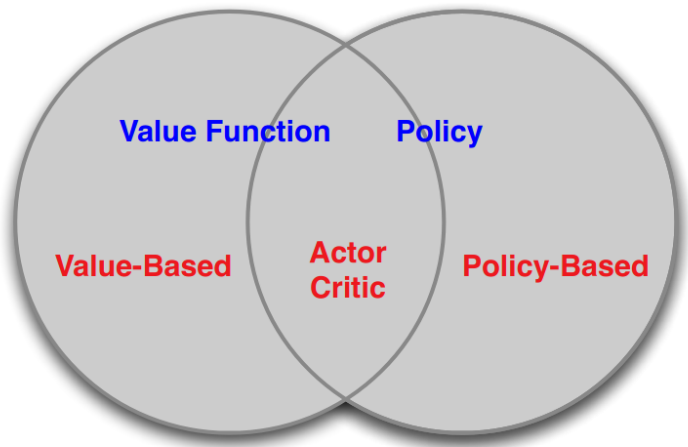
Avantages de l'optimisation de politique :

- Meilleure convergence
- Fonctionne en grande dimension ou si les actions sont continues
- Peut apprendre des politiques stochastiques

Désavantages :

- Converge vers un minimum local
- Grande variance

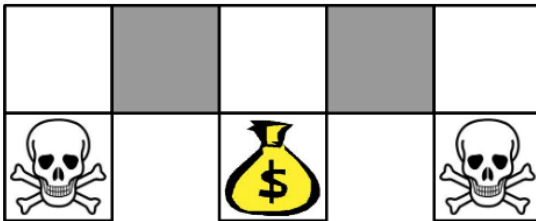
RL et Policy based



Exemple : chifoumi



Exemple : gridworld avec alias



- Descripteurs (*features*) de la forme

$$\phi(s, a) = \begin{cases} 1 & \text{si action } a \text{ possible depuis } s \text{ (pas de mur sur le chemin)} \\ 0 & \text{sinon} \end{cases}$$

Impossibilité de différencier les cases grises

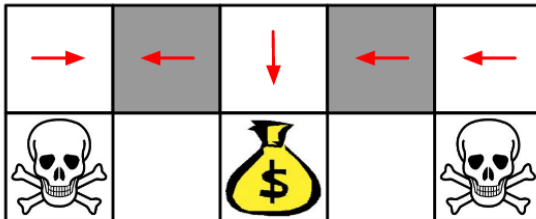
Value based RL :

$$Q_{\theta}(s, a) = f_{\theta}(\phi(s, a))$$

Policy based RL :

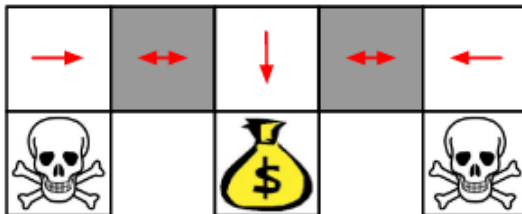
$$\pi_{\theta}(s, a) = g_{\theta}(\phi(s, a))$$

Exemple : gridworld avec alias



- *Value based* : politique optimale déterministe : pas idéale
- Peut être coincée et ne jamais atteindre le but
- Longue convergence

Exemple : gridworld avec alias



- *Policy based* : politique optimale stochastique
- Trouve le but en quelques coups avec une bonne probabilité

Optimisation de la politique

- Problème d'optimisation
- Trouver θ qui maximise l'espérance du retour de l'agent
- De nombreuses approches possibles :
 - Hill climbing
 - Simplex / amoeba / Neleider Mead
 - Genetic algorithms
 - Gradient Ascent ←
 - Conjugate Gradient
 - Quasi-Newton

Plan

- 1 Introduction
- 2 Différences Finies
- 3 *policy gradient*
- 4 Monte Carlo Policy gradient
- 5 Actor Critic Policy Gradient
- 6 Bibliographie

Gradient de la politique

- Quel **objectif** $J(\theta)$ maximiser ?
- On note $v_{\pi_\theta}(s) = v(s; \theta)$ la valeur de l'état s obtenue en suivant la politique π_θ paramétrée par θ
- On prend comme objectif $J(\theta) = v(s; \theta)$
- D'autres objectifs (très semblables) sont aussi possible :
 - La valeur moyenne :

$$J(\theta) = \sum_{s \in \mathcal{S}} d_{\pi_\theta}(s) v_{\pi_\theta}(s)$$

où $d_{\pi_\theta}(s)$ est la probabilité d'être dans l'état s

- La récompense par pas de temps :

$$J(\theta) = \sum_{s \in \mathcal{S}} d_{\pi_\theta}(s) \sum_{a \in \mathcal{A}} \pi_\theta(s, a) \mathcal{R}_s^a$$

Gradient de la politique

- Les algorithmes dits *Policy gradient* cherchent un maximum **local** de $v(s; \theta)$ par montée de gradient par rapport à θ :

$$\Delta\theta = \alpha \nabla_{\theta} v(s; \theta)$$

- α : *learning rate*
- Le *policy gradient* :

$$\nabla_{\theta} v(s; \theta) = \begin{bmatrix} \frac{dv(s; \theta)}{d\theta_1} \\ \vdots \\ \frac{dv(s; \theta)}{d\theta_n} \end{bmatrix}$$

Approche la plus simple : différences finies

Pour chaque dimension $k \in [1, \dots, n]$

- Estimation de la k -ième dérivée de $v(s; \theta)$
- Estimation par une perturbation de θ sur sa composante k
- Nécessite d'être répété pour chaque dimension

Approche la plus simple : différences finies

Pour chaque dimension $k \in [1, \dots, n]$

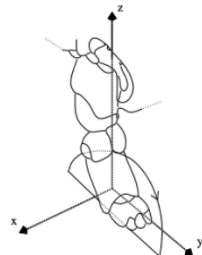
- Estimation de la k -ième dérivée de $v(s; \theta)$
- Estimation par une perturbation de θ sur sa composante k
- Nécessite d'être répété pour chaque dimension
- Simple, fonctionne quelque soit la politique

Approche la plus simple : différences finies

Pour chaque dimension $k \in [1, \dots, n]$

- Estimation de la k -ième dérivée de $v(s; \theta)$
- Estimation par une perturbation de θ sur sa composante k
- Nécessite d'être répété pour chaque dimension
- Simple, fonctionne quelque soit la politique
- Bruité, pas efficace

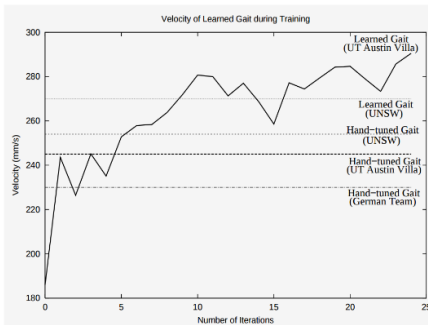
Entraînement par différence finies



- Goal: learn a fast AIBO walk (useful for Robocup)
- Adapt these parameters by finite difference policy gradient
- Evaluate performance of policy by field traversal time

¹Kohl and Stone. Policy gradient reinforcement learning for fast quadrupedal locomotion. ICRA 2004. <http://www.cs.utexas.edu/~ai-lab/pubs/icra04.pdf> ➤

Entraînement par différence finies



- Authors discuss that performance is likely impacted by: initial starting policy parameters, ϵ (how much policies are perturbed), learning rate for how much to change policy, as well as policy parameterization

Plan

- 1 Introduction
- 2 Différences Finies
- 3 *policy gradient***
- 4 Monte Carlo Policy gradient
- 5 Actor Critic Policy Gradient
- 6 Bibliographie

Fonction de score

- *Policy gradient* : Approches basées sur sur une montée de gradient
- On suppose que $\pi_\theta(a|s)$ est différentiable
- Astuce (*likelihood ratio gradient* ou *policy gradient*) :

$$\begin{aligned}\nabla_\theta \pi_\theta(a|s) &= \pi_\theta(a|s) \frac{\nabla_\theta \pi_\theta(a|s)}{\pi_\theta(a|s)} \\ &= \pi_\theta(a|s) \nabla_\theta \log \pi_\theta(a|s)\end{aligned}$$

- $\nabla_\theta \log \pi_\theta(a|s)$: fonction de score

Exemple de score : softmax

- Politique paramétrée comme une régression logistique sur les features :

$$\pi_{\theta}(a|s) = \sigma(\phi(s, a)^T \theta)$$

- σ : softmax :

$$\pi_{\theta}(a|s) = \frac{e^{\phi(s, a)^T \theta}}{\sum_a e^{\phi(s, a)^T \theta}}$$

- Le score est :

$$\nabla_{\theta} \log \pi_{\theta}(a|s) = \phi(s, a) - \mathbb{E}_{\pi_{\theta}}[\phi(s, \cdot)]$$

Exemple de score : politique gaussienne

- Politique possible dans un espace d'action continu
- Moyenne = combinaison linéaire de *features* :

$$\mu(s) = \phi(s, a)^T \theta$$

- Variance σ^2
- Politique :

$$a \sim \mathcal{N}(\mu(s), \sigma^2)$$

- Le score est :

$$\nabla_{\theta} \log \pi_{\theta}(a|s) = \frac{(a - \mu(s))\phi(a, s)}{\sigma^2}$$

Objectif

- Soit une trajectoire τ :

$$\tau = \{s_0, a_0, r_1, \dots, s_{T-1}, a_{T-1}, r_T\}$$

- On note $R(\tau) = \sum_{t=0}^T r_t$ la somme des récompense.
- $T(\tau) = G_0$ pour $\gamma = 1$
- La fonction objectif est donc :

$$v(s_0; \theta) = \mathbb{E}_{\pi_\theta} \left[\sum_{t=0}^T r_t \right] = \sum_{\tau} P(\tau; \theta) R(\tau)$$

- $P(\tau; \theta)$: probabilité de la trajectoire τ en suivant π_θ
- Notre **fonction objectif** est donc :

$$\operatorname{argmax}_{\theta} v(s; \theta) = \operatorname{argmax}_{\theta} \sum_{\tau} P(\tau; \theta) R(\tau)$$

Gradient de la politique

- Notre **fonction objectif** est donc :

$$\operatorname{argmax}_{\theta} v(s; \theta) = \operatorname{argmax}_{\theta} \sum_{\tau} P(\tau; \theta) R(\tau)$$

- On applique l'astuce vue précédemment pour calculer le gradient :

$$\begin{aligned} \nabla_{\theta} v(s_0; \theta) &= \nabla_{\theta} \sum_{\tau} P(\tau; \theta) R(\tau) \\ &= \sum_{\tau} \nabla_{\theta} P(\tau; \theta) R(\tau) \\ &= \sum_{\tau} P(\tau; \theta) \nabla_{\theta} \log P(\tau; \theta) R(\tau) \\ &= \mathbb{E}_{\pi_{\theta}} [R(\tau) \nabla_{\theta} \log P(\tau; \theta)] \end{aligned} \tag{1}$$

Gradient de la politique

- Notre **fonction objectif** est donc :

$$\operatorname{argmax}_{\theta} v(s; \theta) = \operatorname{argmax}_{\theta} \sum_{\tau} P(\tau; \theta) R(\tau)$$

- On applique l'astuce vue précédemment pour calculer le gradient :

$$\nabla_{\theta} v(s; \theta) = \sum_{\tau} P(\tau; \theta) \nabla_{\theta} \log P(\tau; \theta) R(\tau)$$

- Espérance approchée par une moyenne sur un batch de m trajectoires :

$$\nabla_{\theta} v(s; \theta) \simeq \hat{g} = \frac{1}{m} \sum_{i=1}^m \nabla_{\theta} \log P(\tau^{(i)}; \theta) R(\tau^{(i)})$$

Indépendance au modèle

Le gradient de $\log P(\tau; \theta)$ est agnostique à la dynamique du modèle :

$$\nabla_{\theta} \log P(\tau; \theta) = \sum_{t=0}^{T-1} \nabla_{\theta} \log \pi_{\theta}(a_t | s_t)$$

Gradient de la politique

- Notre **fonction objectif** est donc :

$$\operatorname{argmax}_{\theta} V(s; \theta) = \operatorname{argmax}_{\theta} \sum_{\tau} P(\tau; \theta) R(\tau)$$

- On applique l'astuce vue précédemment pour calculer le gradient :

$$\nabla_{\theta} v(s; \theta) = \sum_{\tau} P(\tau; \theta) \nabla_{\theta} \log P(\tau; \theta) R(\tau)$$

- Espérance approchée par une moyenne sur un batch de m trajectoires :

$$\nabla_{\theta} v(s; \theta) \simeq \hat{g} = \frac{1}{m} \sum_{i=1}^m R(\tau_i) \sum_{t=0}^{T-1} \nabla_{\theta} \log \pi_{\theta}(a_t^{(i)} | s_t^{(i)})$$

- Pas besoin de la dynamique du modèle

Plan

- 1 Introduction
- 2 Différences Finies
- 3 *policy gradient*
- 4 Monte Carlo Policy gradient
- 5 Actor Critic Policy Gradient
- 6 Bibliographie

Structure temporelle

- Gradient de l'espérance de la récompense à l'instant t' :

$$\nabla_{\theta} \mathbb{E}[r_{t'}] = \mathbb{E}\left[r_{t'} \sum_{t=0}^{t'} \nabla_{\theta} \log \pi_{\theta}(a_t | s_t)\right]$$

- Structure dans les épisodes : étape par étape $(s_0, a_0, r_1), \dots$, jusqu'à (s_{T-1}, a_{T-1}, r_T)
- En sommant pour tous les instants on a :

$$\begin{aligned} \nabla_{\theta} v(s; \theta) &= \nabla_{\theta} \mathbb{E}[R(\tau)] = \mathbb{E}\left[\sum_{t'=0}^{T-1} r_{t'} \sum_{t=0}^{t'} \nabla_{\theta} \log \pi_{\theta}(a_t | s_t)\right] \\ &= \mathbb{E}\left[\sum_{t=0}^{T-1} \nabla_{\theta} \log \pi_{\theta}(a_t | s_t) \sum_{t'=t}^{T-1} r_{t'}\right] \\ &= \mathbb{E}\left[\sum_{t=0}^{T-1} \nabla_{\theta} \log \pi_{\theta}(a_t | s_t) G_t\right] \end{aligned}$$

Monte Carlo Policy Gradient (REINFORCE)

REINFORCE:

Initialize policy parameters θ arbitrarily

for each episode $\{s_1, a_1, r_2, \dots, s_{T-1}, a_{T-1}, r_T\} \sim \pi_\theta$ **do**

for $t = 1$ to $T - 1$ **do**

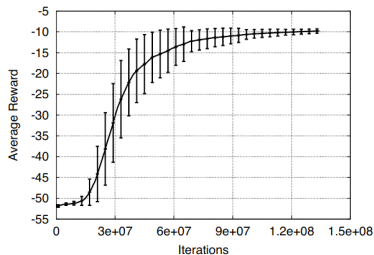
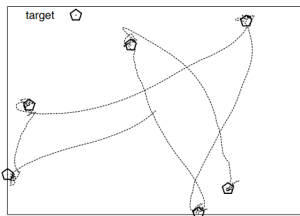
$\theta \leftarrow \theta + \alpha \nabla_\theta \log \pi_\theta(s_t, a_t) G_t$

endfor

endfor

return θ

Exemple : Puck world



- Continuous actions exert small force on puck
- Puck is rewarded for getting close to target
- Target location is reset every 30 seconds
- Policy is trained using variant of Monte-Carlo policy gradient

Monte Carlo Policy Gradient

- MC policy gradient : toujours une grande variance
- Non biaisé

Plan

- 1 Introduction
- 2 Différences Finies
- 3 *policy gradient*
- 4 Monte Carlo Policy gradient
- 5 Actor Critic Policy Gradient**
- 6 Bibliographie

Policy Gradient theorem

Théorème

Pour n'importe quelle politique différentiable $\pi_\theta(a|s)$, pour n'importe quelle fonction objectif $J(\theta)$ (valeur, valeur moyenne, récompense moyenne), le gradient de la politique est donné par :

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta} [\nabla_\theta \log \pi_\theta(a|s) Q_{\pi_\theta}(s, a)]$$

Par rapport à précédemment :

$$\begin{aligned} \nabla_\theta v(s; \theta) &= \mathbb{E} \left[\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t | s_t) G_t \right] \\ &\simeq \mathbb{E} \left[\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a_t | s_t) Q_{\pi_\theta}(s_t, a_t) \right] \end{aligned}$$

Critique

- On a vu :

$$\nabla_{\theta} v(s; \theta) = \mathbb{E} \left[\sum_{t=0}^{T-1} \nabla_{\theta} \log \pi_{\theta}(a_t | s_t) G_t \right]$$

- Peut être estimé sur un batch :

$$\nabla_{\theta} v(s; \theta) \simeq \frac{1}{m} \sum_{i=1}^m \sum_{t=0}^{T-1} \nabla_{\theta} \log \pi_{\theta}(a_t^{(i)} | s_t^{(i)}) G_t^{(i)}$$

- Se baser sur le retour : bruité
- Approche pour réduire la variance : *bootstrapping* TD

$$\nabla_{\theta} v(s; \theta) = \mathbb{E} \left[\sum_{t=0}^{T-1} \nabla_{\theta} \log \pi_{\theta}(a_t | s_t) Q(s_t, a_t) \right]$$

- Estimation de Q est faite par un critique

Actor-critic Methods

- Estimation de Q est fait par un critique :
 - TD
 - SARSA
 - Q-learning
 - Deep Q Networks
- Approches *Actor-critic* maintiennent une représentation explicite de la politique **et** de la valeur
- L'acteur met à jour les paramètres de la politiques θ
- La critique met à jour les paramètres de la fonction de valeur w
- On suit un gradient approché :

$$\nabla J(\theta) \simeq \mathbb{E}[\nabla_{\theta} \log \pi_{\theta}(a|s) Q_w(s, a)]$$

- La mise à jour est donnée par :

$$\Delta\theta = \alpha \nabla_{\theta} \log \pi_{\theta}(a|s) Q_w(s, a)$$

Action-Value Actor Critic

- Critique : mise à jour de w par TD avec $Q_w(s, a) = \phi(s, a)^T w$
- Acteur : mise à jour de θ par monté de gradient

function QAC

Initialise s, θ

Sample $a \sim \pi_\theta$

for each step **do**

Sample reward $r = \mathcal{R}_s^a$; sample transition $s' \sim \mathcal{P}_{s'}^a$.

Sample action $a' \sim \pi_\theta(s', a')$

$\delta = r + \gamma Q_w(s', a') - Q_w(s, a)$

$\theta = \theta + \alpha \nabla_\theta \log \pi_\theta(s, a) Q_w(s, a)$

$w \leftarrow w + \beta \delta \phi(s, a)$

$a \leftarrow a', s \leftarrow s'$

end for

end function

Fonction de valeur compatibles

Theorem

Si les conditions suivantes sont réunies :

- La fonction d'approximation de Q est compatible avec la politique :

$$\nabla_w Q_w(s, a) = \nabla_\theta \log \pi_\theta(a|s)$$

- Les paramètres w minimisent l'erreur quadratique moyenne (MSE)

Alors, le gradient de la politique est exact :

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta} [\nabla_\theta \log \pi_\theta(s, a) Q_w(s, a)]$$

Utilisation de Baseline

- Espérance de la politique estimée : grande variance
- Réduction de la variance grâce à une *baseline* $B(s)$:

$$\nabla_{\theta} J(\theta) = \mathbb{E}_{\pi_{\theta}} [\nabla_{\theta} \log \pi_{\theta}(a|s)(Q_{\pi_{\theta}}(s, a) - B(s))]$$

- Aucun impact sur l'espérance
- Une bonne *baseline* est la fonction de valeur elle même : $B(s) = V_{\pi_{\theta}}(s)$
- Le gradient est donc donné en fonction de l'avantage $A_{\pi_{\theta}}$:

$$\nabla_{\theta} J(\theta) = \mathbb{E}_{\pi_{\theta}} [\nabla_{\theta} \log \pi_{\theta}(a|s)(Q_{\pi_{\theta}}(s, a) - V_{\pi_{\theta}}(s))]$$

$$\nabla_{\theta} J(\theta) = \mathbb{E}_{\pi_{\theta}} [\nabla_{\theta} \log \pi_{\theta}(a|s)A_{\pi_{\theta}}(s, a)]$$

Estimation de l'avantage

- L'erreur TD est un estimateur de l'avantage :

$$\delta_{\pi_{\theta}}(s, s') = r + \gamma V_{\pi_{\theta}}(s') - V_{\pi_{\theta}}(s)$$

- Le gradient est donc donné par :

$$\nabla_{\theta} J(\theta) = \mathbb{E}_{\pi_{\theta}}[\nabla_{\theta} \log \pi_{\theta}(a|s) \delta_{\pi_{\theta}}(s, s')]$$

- On a seulement à estimer la valeur d'état

Plan

- 1 Introduction
- 2 Différences Finies
- 3 *policy gradient*
- 4 Monte Carlo Policy gradient
- 5 Actor Critic Policy Gradient
- 6 Bibliographie

Bibliographie

- Reinforcement Learning : an introduction, second edition, Richard S. Sutton and Adrew G. Barto
- Reinforcement Learning courses, David Silver, DeepMind (<https://www.davidsilver.uk/>)
- A3C (Mnih et al. ICML 2016)
- T. P. Lillicrap et al., Continuous control with deep reinforcement learning.