Apprentissage par renforcement appliqué

Méthodes approximatives et DRL [revision 3.3]

Brahim Chaib-draa

Brahim.Chaib-Draa@ift.ulaval.ca

Université Laval

2020-07-04

Concepts clés en RL (la suite)

Concepts clés vus à la partie précédente

Concepts HAUT niveau :

- Apprentissage **sans modèle** vs apprentissage **basé sur un modèle**
  [*Model-free* vs *model-based*]
- **Apprentissage** vs **Planification**
  [*Learning* vs *Planning*]
- Méthodes **basées sur les valeurs** vs méthodes par **recherche de politique**
  [*Value based methods* vs *Policy search methods*]
- Méthodes **tabulaires** vs méthodes **approximatives**
  [*Tabular solution methods* vs *Approximate solution methods*]

Concept BAS niveau :

- **Prédiction** vs **Contrôle**
  [*Prediction* vs *Control*]
- Apprentissage **EN-ligne** vs **HORS-ligne**
  [*ON-line learning* vs *OFF-line learning*]
- ***ON-policy*** vs ***OFF-Policy***

Concepts clés vus aujourd'hui

Concepts HAUT niveau :

- Apprentissage **sans modèle** vs apprentissage **basé sur un modèle**
  [*Model-free* vs *model-based*]
- **Apprentissage** vs **Planification**
  [*Learning* vs *Planning*]
- Méthodes **basées sur les valeurs** vs méthodes par **recherche de politique**
  [*Value based methods* vs *Policy search methods*]
- Méthodes **tabulaires** vs méthodes **approximatives**
  [*Tabular solution methods* vs *Approximate solution methods*]

Concept BAS niveau :

- **Prédiction** vs **Contrôle**
  [*Prediction* vs *Control*]
- Apprentissage **EN-ligne** vs **HORS-ligne**
  [*ON-line learning* vs *OFF-line learning*]
- ***ON-policy*** vs ***OFF-Policy***

Concepts clés en RL (la suite)

Méthodes basées sur les valeurs vs méthodes par recherche de politique

Méthodes basées sur les valeurs vs méthodes par recherche de politique

Concept HAUT niveau

Méthodes basées sur les valeurs *[Value based methods]*

L'algorithme **apprend à prédire V^π et/ou Q^π**
pour ensuite trouver $\pi^\star$ pas inférence

Méthodes par recherche de politique *[Policy search methods]*

L'algorithme **apprend directement $\pi^\star$**

Concepts clés en RL (la suite)

Méthodes OFF-policy vs ON-Policy

Méthodes *ON-Policy* vs méthodes *OFF-policy*

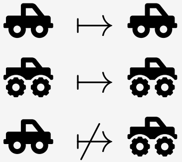
Concept BAS niveau

Méthodes *ON-Policy*

La phase d'apprentissage est faite en **suivant l'assomption** que les échantillons proviennent tous de la même distribution.

$\Longrightarrow$

Si la politique apprise change (même juste un peu), alors on doit utiliser de nouveaux échantillons tirés de cette nouvelle politique.



Analogie On doit réapprendre à conduire de zéro à chaque fois qu'on change de pneu (hivers/été/usure différente/adhérence différente).

Méthodes *ON-Policy* vs méthodes *OFF-policy*

Concept BAS niveau

Méthodes *OFF-policy*

La phase d'apprentissage est faite **sans faire d'assomption** par rapport à la distribution d'où à été tiré chacun des échantillons.

$\Longrightarrow$

On peut utiliser des échantillons plus anciens, provenant de distribution différente, provenant d'une politique entraînée sur un problème plus général ...

Analogie : Ma vaste expérience à conduire des autos tamponneuses et des tracteurs à gazon m'a beaucoup appris sur l'art de piloter des *Formules 1*.

« Du point de vue de l'algorithme, les échantillons collectés ne sont pas liés à la politique apprise »

Concepts clés en RL (la suite)

Méthodes tabulaires vs méthodes approximatives

Méthodes tabulaires vs méthodes approximatives

Concept HAUT niveau

Méthodes tabulaires [*Tabular solution methods*]

L'algorithme **conserve en mémoire** toutes les valeurs prises aux travers **tout l'espace d'état** possible ;

Ex. $Q^\pi(\mathbf{s}, \mathbf{a}) =$

	$\mathbf{a}_1$	$\mathbf{a}_2$	$\mathbf{a}_3$	$\mathbf{a}_4$
$\mathbf{s}_1$	0.4	0.2	1.0	0.8
$\mathbf{s}_2$	0.6	0.7	0.7	0.6
$\mathbf{s}_3$	0.7	-0.3	0.4	0.2
$\mathbf{s}_4$	0.5	0.1	0.2	0.4

Méthodes approximatives [*Approximate solution methods*]

L'algorithme **apprend à généraliser** sur la base de son expérience afin de **composer avec l'inconnue** en utilisant approximateur de fonction ;

Ex. $Q^\pi(\mathbf{s}, \mathbf{a}) \approx \widehat{Q}^\pi_\theta(\mathbf{s}, \mathbf{a})$

Méthodes approximatives

Cette partie va introduire les algorithmes d'apprentissage par renforcement approximatif **en mettant l'accent** sur les algorithmes utilisant des **réseaux neuronaux profonds** comme approximateur de fonction

... d'où le nom **apprentissage par renforcement profond** [*DRL*] .

Remarque : méthodes DRL $\in$ méthodes RL approximatives

Les algorithmes d'apprentissage par renforcement profond **ne sont pas les seules méthodes approximatives** en RL. Il est même possible d'avoir de meilleur résultat (dans certaines circonstances) en utilisant d'autres types d'approximateur de fonction :

- Arbre de décision
- *K-nearest*
- Combinaison linéaire de fonction caractéristique (aka. *feature*)
- ...

Méthodes approximatives

Problèmes des Méthodes tabulaires :

1. Fonctionnent sous l'assomption qu'il est possible pendant l'apprentissage de **traverser tout l'espace des possibilités** [1]
$\Longrightarrow$
plus l'espace est large, plus l'apprentissage devient lent . . . au point de devenir intractable ;
2. Nécessite de **conserver en mémoire tout l'espace** des possibilités ;

Solution : **Apprendre un approximateur de fonction**

- à partir d'un **petit sous-ensemble de l'espace des possibilités**
- de façon à être capable de **généraliser à des situations jamais rencontré** en entrainement ;

Con : Les **garanties théoriques** établient pour les méthodes tabulaires **ne se transfèrent pas nécessairement** aux méthodes RL avec approximateurs de fonction.

1. Içi, l'espace des possibilités sous-entend l'espace d'état $\mathcal{S}$ ou d'état-action $\mathcal{A}$

Méthodes approximatives

Problèmes des Méthodes tabulaires :

1. Fonctionnent sous l'assomption qu'il est possible pendant l'apprentissage de **traverser tout l'espace des possibilités**[1]
   $\Longrightarrow$ plus l'espace est large, plus l'apprentissage devient lent . . . au point de devenir intractable ;
2. Nécessite de **conserver en mémoire tout l'espace** des possibilités ;

Solution : Apprendre un approximateur de fonction

- à partir d'un **petit sous-ensemble de l'espace des possibilités**
- de façon à être capable de **généraliser à des situations jamais rencontré** en entrainement ;

Con : Les **garanties théoriques** établient pour les méthodes tabulaires **ne se transfèrent pas nécessairement** aux méthodes RL avec approximateurs de fonction.

1. Içi, l'espace des possibilités sous-entend l'espace d'état $\mathcal{S}$ ou d'état-action $\mathcal{A}$

Méthodes approximatives

Problèmes des Méthodes tabulaires :

1. Fonctionnent sous l'assomption qu'il est possible pendant l'apprentissage de **traverser tout l'espace des possibilités**[1]
$\Longrightarrow$ plus l'espace est large, plus l'apprentissage devient lent . . . au point de devenir intractable ;
2. Nécessite de **conserver en mémoire tout l'espace** des possibilités ;

Solution : Apprendre un approximateur de fonction

- à partir d'un **petit sous-ensemble de l'espace des possibilités**
- de façon à être capable de **généraliser à des situations jamais rencontré** en entrainement ;

Con : Les **garanties théoriques** établient pour les méthodes tabulaires **ne se transfèrent pas nécessairement** aux méthodes RL avec approximateurs de fonction.

1. Içi, l'espace des possibilités sous-entend l'espace d'état $\mathcal{S}$ ou d'état-action $\mathcal{A}$

Algorithme vu à la partie précédente

Problème de décision séquentiel (PDS)

« Modèle connu **et** parfait »

Programmation dynamique (exacte)

Itération de Q-valeur

Itération de politique

« Modèle inconnu **ou** imparfait »

Apprentissage par renforcement

Basé sur les **valeurs**

Monte Carlo
avec *Départ Exploratoire*

Différence Temporelle
Q-learning (tabulaire)

Par recherche de **politique**

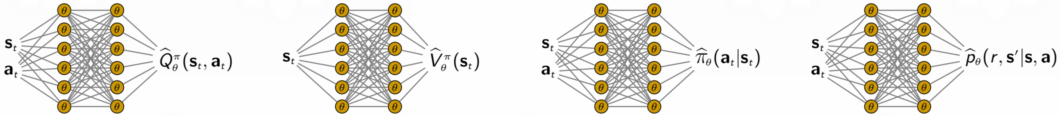
Algorithmes vus dans cette partie

Apprentissage par renforcement [profond]

- Apprentissage **sans** modèle
  - Basé sur les **valeurs**
    - **Q-learning** (approximatif)
    - [Double] **DQN**
  - Par recherche de **politique**
    - **Policy Gradient**
    - **Actor-Critic**
- Apprentissage **avec** modèle
  - Modèle **connu** mais pas de solution analytique
    - **Monte Carlo Tree Search**
    - **AlphaGo**
  - Modèle **appris** (Pas vue en cours)

Apprentissage par renforcement profond

Apprentissage par renforcement profond [*DRL*]



Caractéristiques : Apprend à **généraliser** son expérience **à des situations jamais rencontré** ;

Idées clés :

- Utiliser des réseaux neuronaux comme approximateur de fonction pour **représenter** (un ou plusieurs) :
  - fonction de valeur V^{π} et/ou Q^{π}
  - politique π
  - modèle $p(r, \mathbf{s}'|\mathbf{s}, \mathbf{a})$
- Optimiser l'objectif par **descente de gradient stochastique** ;

Apprentissage par renforcement profond [*DRL*]

Pro :

- Capable de fonctionner sur des espaces de possibilité massifs ;
- Capable de gérer les situations inconnues (en principe) ;
- Pas de fonction caractéristique à développer (end-to-end) ;

Con :

- Perte de la plupart des garanties théoriques des méthodes tabulaires ;

Enjeux spécifiques au mariage RL/DNN

Processus de décision séquentielle $\Longrightarrow$ données séquentielles $\Longrightarrow$

- les données sont **corrélées** ;
- les données ne sont **pas indépendantes et identiquement distribuées** ;

⚠ **Problème :** les réseaux neuronaux profonds **fonctionnent sous l'assomption** que les données sont iid.

Comment faire pour réconcilier les deux ?

DRL et méthode basée sur les valeurs

DRL et méthode basée sur les valeurs

Q-learning approximatif

Rappel

★ | « The max trick »

Sauter l'étape d'avoir à représenter explicitement la politique π

$$\max_{\mathbf{a}\in\mathcal{A}(\mathbf{s})} Q^{\pi}(\mathbf{s},\mathbf{a}) \quad \Longleftrightarrow \quad Q^{\pi}\Big(\mathbf{s},\ \pi_{greedy}(\mathbf{s})\Big) \quad \text{avec} \quad \pi_{greedy}(\mathbf{s}) \ := \ \arg\max_{\mathbf{a}\in\mathcal{A}(\mathbf{s})} Q(\mathbf{s},\mathbf{a})$$

Algorithme | Q-Learning (tabulaire)

Soit $\mu(\cdot|\mathbf{s}_t)$, une politique exploratoire de votre choix, (ex. : aléatoirement, « greedy », « ϵ-greedy », . . .)
la politique cible $\pi(\mathbf{s}) \doteq \pi_{greedy}$, le « learning rate » $\alpha \in \,]0,1]$ et le « global step » $k \leftarrow 0$

Répéter *pour chaque trajectoire τ* ***jusqu'à convergence*** *vers* $\pi^{\star}$ **:**

  Répéter *pour chaque « timestep »* $t = 0,1,2,\ldots,t,t{+}1,\ldots,\mathrm{T}$ *de la trajectoire τ* **:**

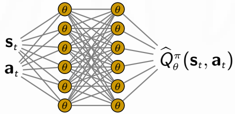
$$r_{t+1}^{(k)},\ \mathbf{s}_{t+1}^{(k)} \quad \longleftarrow \quad \text{exécuter } \mathbf{a}_t \sim \mu(\cdot|\mathbf{s}_t) \text{ dans l'environnement et observer la réaction}$$

$$Q_{k+1}^{\pi}(\mathbf{s}_t,\mathbf{a}_t) \quad \longleftarrow \quad Q_k^{\pi}(\mathbf{s}_t,\mathbf{a}_t) + \alpha\left[r_{t+1}^{(k)} + \gamma \max_{\mathbf{a}'\in\mathcal{A}(\mathbf{s}_{t+1})} Q_k^{\pi}(\mathbf{s}_{t+1}^{(k)},\mathbf{a}') - Q_k^{\pi}(\mathbf{s}_t,\mathbf{a}_t)\right]$$

$$\mathbf{s}_t \leftarrow \mathbf{s}_{t+1}^{(k)} \quad \text{et} \quad k \leftarrow k+1$$

Q-learning approximatif *[Approximate Q-learning]*

$$\widehat{Q}^{\pi}_{\theta}(\mathbf{s}_t, \mathbf{a}_t) \;=\; \theta_1 f_1(\mathbf{s}_t, \mathbf{a}_t) + \theta_2 f_2(\mathbf{s}_t, \mathbf{a}_t) + \ldots + \theta_n f_n(\mathbf{s}_t, \mathbf{a}_t)$$



Caractéristique : Apprentisage **EN-ligne** et **OFF-policy** (comme Q-learning tabulaire)

Idées clés :

1. **Représenter** la fonction de valeur par un approximateur de fonction :

$$Q^{\pi}(\mathbf{s}, \mathbf{a}) \;\approx\; \widehat{Q}^{\pi}_{\theta}(\mathbf{s}, \mathbf{a}) \qquad \text{avec } \theta \text{ un vecteur de paramètres}$$

2. Définir une **fonction objectif** par *erreur quadratique moyenne* [*MSE*] et son **gradient** :

$$L_k(\theta_k) \;=\; \mathbb{E}_{\mathbf{s}'\sim p(\cdot\,|\,\mathbf{s},\mathbf{a})}\left[\left(\mathbf{y} - \widehat{Q}^{\pi}_{\theta}(\mathbf{s}, \mathbf{a})\right)^2 \Big|_{\theta=\theta_k}\right] \qquad \text{avec } \mathbf{y} = r(\mathbf{s}, \mathbf{a}, \mathbf{s}') + \gamma \max_{\mathbf{a}'\in\mathcal{A}(\mathbf{s}')} \widehat{Q}^{\pi}_{\theta}(\mathbf{s}', \mathbf{a}')$$

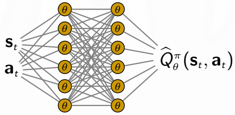
$$\tfrac{1}{2}\nabla_{\theta_k} L(\theta_k) \;=\; \mathbb{E}_{\mathbf{s}'\sim p(\cdot\,|\,\mathbf{s},\mathbf{a})}\left[\left(r(\mathbf{s}, \mathbf{a}, \mathbf{s}') + \gamma \max_{\mathbf{a}'\in\mathcal{A}(\mathbf{s}')} \widehat{Q}^{\pi}_{\theta}(\mathbf{s}', \mathbf{a}') - \widehat{Q}^{\pi}_{\theta}(\mathbf{s}, \mathbf{a})\right) \nabla_{\theta_k} \widehat{Q}^{\pi}_{\theta}(\mathbf{s}, \mathbf{a}) \Big|_{\theta=\theta_k}\right]$$

3. **Optimiser** l'objectif par *descente de gradient stochastique* [*SGD*] :

$$\theta_{k+1} \;\longleftarrow\; \theta_k - \alpha \nabla_{\theta_k} L(\theta_k)$$

Q-learning approximatif *[Approximate Q-learning]*

$$\widehat{Q}^{\pi}_{\theta}(\mathbf{s}_t,\mathbf{a}_t) \;=\; \theta_1 f_1(\mathbf{s}_t,\mathbf{a}_t) + \theta_2 f_2(\mathbf{s}_t,\mathbf{a}_t) + \ldots + \theta_n f_n(\mathbf{s}_t,\mathbf{a}_t)$$



Caractéristique : Apprentisage **EN-ligne** et **OFF-policy** (comme Q-learning tabulaire)

Idées clés :

1. **Représenter** la fonction de valeur par un approximateur de fonction :

$$Q^{\pi}(\mathbf{s},\mathbf{a}) \;\approx\; \widehat{Q}^{\pi}_{\theta}(\mathbf{s},\mathbf{a}) \quad \text{avec } \theta \text{ un vecteur de paramètres}$$

2. Définir une **fonction objectif** par *erreur quadratique moyenne* [*MSE*] et son **gradient** :

$$L_k(\theta_k) \;=\; \mathbb{E}_{\mathbf{s}'\sim p(\cdot\,|\,\mathbf{s},\mathbf{a})}\left[\left(\mathbf{y} - \widehat{Q}^{\pi}_{\theta}(\mathbf{s},\mathbf{a})\right)^2 \Big|_{\theta=\theta_k}\right] \qquad \text{avec } \mathbf{y} = r(\mathbf{s},\mathbf{a},\mathbf{s}') + \gamma \max_{\mathbf{a}'\in\mathcal{A}(\mathbf{s}')} \widehat{Q}^{\pi}_{\theta}(\mathbf{s}',\mathbf{a}')$$

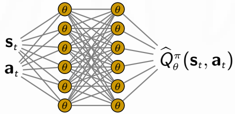
$$\tfrac{1}{2}\nabla_{\theta_k} L(\theta_k) \;=\; \mathbb{E}_{\mathbf{s}'\sim p(\cdot\,|\,\mathbf{s},\mathbf{a})}\left[\left(r(\mathbf{s},\mathbf{a},\mathbf{s}') + \gamma \max_{\mathbf{a}'\in\mathcal{A}(\mathbf{s}')} \widehat{Q}^{\pi}_{\theta}(\mathbf{s}',\mathbf{a}') - \widehat{Q}^{\pi}_{\theta}(\mathbf{s},\mathbf{a})\right) \nabla_{\theta_k}\widehat{Q}^{\pi}_{\theta}(\mathbf{s},\mathbf{a}) \Big|_{\theta=\theta_k}\right]$$

3. **Optimiser** l'objectif par *descente de gradient stochastique* [*SGD*] :

$$\theta_{k+1} \;\longleftarrow\; \theta_k - \alpha \nabla_{\theta_k} L(\theta_k)$$

Q-learning approximatif *[Approximate Q-learning]*

$$\widehat{Q}^{\pi}_{\theta}(\mathbf{s}_t, \mathbf{a}_t) = \theta_1 f_1(\mathbf{s}_t, \mathbf{a}_t) + \theta_2 f_2(\mathbf{s}_t, \mathbf{a}_t) + \ldots + \theta_n f_n(\mathbf{s}_t, \mathbf{a}_t)$$



Caractéristique : Apprentisage **EN-ligne** et **OFF-policy** (comme Q-learning tabulaire)

Idées clés :

1. **Représenter** la fonction de valeur par un approximateur de fonction :

$$Q^{\pi}(\mathbf{s}, \mathbf{a}) \approx \widehat{Q}^{\pi}_{\theta}(\mathbf{s}, \mathbf{a}) \quad \text{avec } \theta \text{ un vecteur de paramètres}$$

2. Définir une **fonction objectif** par *erreur quadratique moyenne* [*MSE*] et son **gradient** :

$$L_k(\theta_k) = \mathbb{E}_{\mathbf{s}' \sim p(\cdot\,|\mathbf{s},\mathbf{a})}\left[\left(\mathbf{y} - \widehat{Q}^{\pi}_{\theta}(\mathbf{s}, \mathbf{a}) \right)^2 \Bigg|_{\theta=\theta_k} \right] \quad \text{avec } \mathbf{y} = r(\mathbf{s}, \mathbf{a}, \mathbf{s}') + \gamma \max_{\mathbf{a}' \in \mathcal{A}(\mathbf{s}')} \widehat{Q}^{\pi}_{\theta}(\mathbf{s}', \mathbf{a}')$$

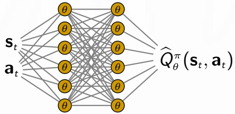
$$\tfrac{1}{2}\nabla_{\theta_k} L(\theta_k) = \mathbb{E}_{\mathbf{s}' \sim p(\cdot\,|\mathbf{s},\mathbf{a})}\left[\left(r(\mathbf{s}, \mathbf{a}, \mathbf{s}') + \gamma \max_{\mathbf{a}' \in \mathcal{A}(\mathbf{s}')} \widehat{Q}^{\pi}_{\theta}(\mathbf{s}', \mathbf{a}') - \widehat{Q}^{\pi}_{\theta}(\mathbf{s}, \mathbf{a}) \right) \nabla_{\theta_k} \widehat{Q}^{\pi}_{\theta}(\mathbf{s}, \mathbf{a}) \Bigg|_{\theta=\theta_k} \right]$$

3. **Optimiser** l'objectif par *descente de gradient stochastique* [*SGD*] :

$$\theta_{k+1} \longleftarrow \theta_k - \alpha \nabla_{\theta_k} L(\theta_k)$$

Q-learning approximatif [*Approximate Q-learning*]

$$\widehat{Q}^{\pi}_{\theta}(\mathbf{s}_t, \mathbf{a}_t) \;=\; \theta_1 f_1(\mathbf{s}_t, \mathbf{a}_t) + \theta_2 f_2(\mathbf{s}_t, \mathbf{a}_t) + \ldots + \theta_n f_n(\mathbf{s}_t, \mathbf{a}_t)$$



Caractéristique : Apprentisage **EN-ligne** et **OFF-policy** (comme Q-learning tabulaire)

Idées clés :

1. **Représenter** la fonction de valeur par un approximateur de fonction :

$$Q^{\pi}(\mathbf{s}, \mathbf{a}) \;\approx\; \widehat{Q}^{\pi}_{\theta}(\mathbf{s}, \mathbf{a}) \qquad \text{avec } \theta \text{ un vecteur de paramètres}$$

2. Définir une **fonction objectif** par *erreur quadratique moyenne* [*MSE*] et son **gradient** :

$$L_k(\theta_k) \;=\; \mathbb{E}_{\mathbf{s}' \sim p(\cdot\,|\mathbf{s},\mathbf{a})}\left[\left(\mathbf{y} \;-\; \widehat{Q}^{\pi}_{\theta}(\mathbf{s}, \mathbf{a}) \right)^2 \Big|_{\theta=\theta_k} \right] \qquad \text{avec } \mathbf{y} \;=\; r(\mathbf{s}, \mathbf{a}, \mathbf{s}') + \gamma \max_{\mathbf{a}' \in \mathcal{A}(\mathbf{s}')} \widehat{Q}^{\pi}_{\theta}(\mathbf{s}', \mathbf{a}')$$

$$\frac{1}{2}\nabla_{\theta_k} L(\theta_k) \;=\; \mathbb{E}_{\mathbf{s}' \sim p(\cdot\,|\mathbf{s},\mathbf{a})}\left[\left(r(\mathbf{s}, \mathbf{a}, \mathbf{s}') + \gamma \max_{\mathbf{a}' \in \mathcal{A}(\mathbf{s}')} \widehat{Q}^{\pi}_{\theta}(\mathbf{s}', \mathbf{a}') \;-\; \widehat{Q}^{\pi}_{\theta}(\mathbf{s}, \mathbf{a}) \right) \nabla_{\theta_k} \widehat{Q}^{\pi}_{\theta}(\mathbf{s}, \mathbf{a}) \Big|_{\theta=\theta_k} \right]$$

3. **Optimiser** l'objectif par *descente de gradient stochastique* [*SGD*] :

$$\theta_{k+1} \;\longleftarrow\; \theta_k \;-\; \alpha \nabla_{\theta_k} L(\theta_k)$$

Algorithme | Q-Learning (approximatif)

Soit la politique exploratoire $\mu(\cdot|\mathbf{s})$, la politique cible $\pi(\mathbf{s}) \doteq \pi_{greedy}$, le « learning rate » $\alpha \in]0,1]$, le « global step » $k \leftarrow 0$, et l'approximateur de fonction $\widehat{Q}^{\pi}_{\theta}$ avec le vecteur de paramètre θ

Répéter *pour chaque trajectoire* $\mathcal{T}$ ***jusqu'à convergence*** *vers* $\pi^{\star}$ **:**

  Répéter *pour chaque « timestep »* $t = 0, 1, 2, \ldots, t, t{+}1, \ldots, \mathrm{T}$ *de la trajectoire* $\mathcal{T}$ **:**

$$r^{(k)}_{t+1},\ \mathbf{s}^{(k)}_{t+1} \longleftarrow \text{exécuter } \mathbf{a}_t \sim \mu(\cdot|\mathbf{s}_t) \text{ dans l'environnement et observer la réaction}$$

$$\mathbf{y} = r^{(k)}_{t+1} + \gamma \max_{\mathbf{a}' \in \mathcal{A}(\mathbf{s}_t)} \widehat{Q}^{\pi}_{\theta}(\mathbf{s}^{(k)}_{t+1}, \mathbf{a}')$$

$$\theta_{k+1} \longleftarrow \theta_k - \alpha \left(\mathbf{y} - \widehat{Q}^{\pi}_{\theta}(\mathbf{s}_t, \mathbf{a}_t) \right) \frac{\partial}{\partial \theta_{\mathbf{s},\mathbf{a}}} \widehat{Q}^{\pi}_{\theta}(\mathbf{s}_t, \mathbf{a}_t)$$

$$\mathbf{s}_t \leftarrow \mathbf{s}^{(k)}_{t+1} \quad \text{et} \quad k \leftarrow k + 1$$

Pro : Fonctionne bien avec un approximateurs linéaire

Con : Oscille ou diverge avec un réseau neuronaux

Algorithme | Q-Learning (approximatif)

Soit la politique exploratoire $\mu(\cdot|\mathbf{s})$, la politique cible $\pi(\mathbf{s}) \doteq \pi_{greedy}$, le « learning rate » $\alpha \in]0,1]$, le « global step » $k \leftarrow 0$, et l'approximateur de fonction $\widehat{Q}^{\pi}_{\theta}$ avec le vecteur de paramètre θ

Répéter *pour chaque trajectoire* $\mathcal{T}$ ***jusqu'à convergence*** *vers* $\pi^{\star}$ **:**

Répéter *pour chaque « timestep »* $t = 0, 1, 2, \ldots, t, t{+}1, \ldots, \mathrm{T}$ *de la trajectoire* $\mathcal{T}$ **:**

$r^{(k)}_{t+1}, \mathbf{s}^{(k)}_{t+1} \longleftarrow$ exécuter $\mathbf{a}_t \sim \mu(\cdot|\mathbf{s}_t)$ dans l'environnement et observer la réaction

$$\mathbf{y} = r^{(k)}_{t+1} + \gamma \max_{\mathbf{a}' \in \mathcal{A}(\mathbf{s}_t)} \widehat{Q}^{\pi}_{\theta}(\mathbf{s}^{(k)}_{t+1}, \mathbf{a}')$$

$$\theta_{k+1} \longleftarrow \theta_k - \alpha\left(\mathbf{y} - \widehat{Q}^{\pi}_{\theta}(\mathbf{s}_t, \mathbf{a}_t)\right) \frac{\partial}{\partial \theta_{\mathbf{s},\mathbf{a}}} \widehat{Q}^{\pi}_{\theta}(\mathbf{s}_t, \mathbf{a}_t)$$

$\mathbf{s}_t \leftarrow \mathbf{s}^{(k)}_{t+1}$ et $k \leftarrow k+1$

Pro : Fonctionne bien avec un approximateurs linéaire

Con : **Oscille ou diverge** avec un réseau neuronaux

DRL et méthode basée sur les valeurs

Deep Q-Network

⚠ Q-learning avec réseau neuronaux est instable

Causes 1 : Les données sont séquentielles $\Longrightarrow$ les échantillons sont corrélés et non-iid

$$\tau = \mathbf{s}_0, \mathbf{a}_0, r_1, \mathbf{s}_1, \mathbf{a}_1, r_2, \mathbf{s}_2, \ldots, \mathbf{s}_t, \mathbf{a}_t, r_{t+1}, \mathbf{s}_{t+1}, \ldots, \mathbf{s}_{T-1}, \mathbf{a}_{T-1}, r_T, \mathbf{s}_T$$

Conséquence : l'algorithme « *overfit* » une multitude de sous-séquences de l'espace

Solution : **Emmagaziner les transitions** expérimentées dans un « **replay buffer** »

$$\left\{ \left(\mathbf{s}_0^{(1)}, \mathbf{a}_0^{(1)}, r_1^{(1)}, \mathbf{s}_1^{(1)}\right), \left(\mathbf{s}_1^{(2)}, \mathbf{a}_1^{(2)}, r_2^{(2)}, \mathbf{s}_2^{(2)}\right), \ldots, \left(\mathbf{s}_t^{(n)}, \mathbf{a}_t^{(n)}, r_{t+1}^{(n)}, \mathbf{s}_{t+1}^{(n)}\right) \right\}$$

et **apprendre à partir de transitions sélectionnées au hasard** (dans le « replay buffer ») [1].

L'apprentissage se fait donc sur des données qui ne sont plus séquentielles puisque l'ordre d'apparition des transitions est mélangé [2] ce qui brise la corrélation des échantillons et nous ramène dans un état iid.

1. Cette technique s'appelle « Experience replay ». Voir Deep Q-Network | Ressource théorique en fin de section
2. On peu le faire parce que l'algorithme Q-learning est « OFF-policy »

⚠ Q-learning avec réseau neuronaux est instable

Causes 1 : Les données sont séquentielles $\Longrightarrow$ les échantillons sont corrélés et non-iid

$$\tau \quad = \quad \mathbf{s}_0, \mathbf{a}_0, r_1, \mathbf{s}_1, \mathbf{a}_1, r_2, \mathbf{s}_2, \ldots, \mathbf{s}_t, \mathbf{a}_t, r_{t+1}, \mathbf{s}_{t+1}, \ldots, \mathbf{s}_{T-1}, \mathbf{a}_{T-1}, r_T, \mathbf{s}_T$$

Conséquence : l'algorithme « *overfit* » une multitude de sous-séquences de l'espace

Solution : **Emmagaziner les transitions** expérimentées dans un « **replay buffer** »

$$\left\{ \left(\mathbf{s}_0^{(1)}, \mathbf{a}_0^{(1)}, r_1^{(1)}, \mathbf{s}_1^{(1)}\right), \left(\mathbf{s}_1^{(2)}, \mathbf{a}_1^{(2)}, r_2^{(2)}, \mathbf{s}_2^{(2)}\right), \ldots, \left(\mathbf{s}_t^{(n)}, \mathbf{a}_t^{(n)}, r_{t+1}^{(n)}, \mathbf{s}_{t+1}^{(n)}\right) \right\}$$

et **apprendre à partir de transitions sélectionnées au hasard** (dans le « replay buffer »)[1].

L'apprentissage se fait donc sur des données qui ne sont plus séquentielles puisque l'ordre d'apparition des transitions est mélangé[2] ce qui brise la corrélation des échantillons et nous ramène dans un état iid.

1. Cette technique s'appelle « Experience replay ». Voir Deep Q-Network | Ressource théorique en fin de section
2. On peu le faire parce que l'algorithme Q-learning est « OFF-policy »

⚠ Q-learning avec réseau neuronaux est instable

Causes 1 : Les données sont séquentielles $\Longrightarrow$ les échantillons sont corrélés et non-iid

$$\tau \quad = \quad \mathbf{s}_0, \mathbf{a}_0, r_1, \mathbf{s}_1, \mathbf{a}_1, r_2, \mathbf{s}_2, \ldots, \mathbf{s}_t, \mathbf{a}_t, r_{t+1}, \mathbf{s}_{t+1}, \ldots, \mathbf{s}_{\mathsf{T}-1}, \mathbf{a}_{\mathsf{T}-1}, r_{\mathsf{T}}, \mathbf{s}_{\mathsf{T}}$$

Conséquence : l'algorithme « *overfit* » une multitude de sous-séquences de l'espace

Solution : **Emmagaziner les transitions** expérimentées dans un « **replay buffer** »

$$\left\{ \left(\mathbf{s}_0^{(1)}, \mathbf{a}_0^{(1)}, r_1^{(1)}, \mathbf{s}_1^{(1)}\right), \left(\mathbf{s}_1^{(2)}, \mathbf{a}_1^{(2)}, r_2^{(2)}, \mathbf{s}_2^{(2)}\right), \ldots, \left(\mathbf{s}_t^{(n)}, \mathbf{a}_t^{(n)}, r_{t+1}^{(n)}, \mathbf{s}_{t+1}^{(n)}\right) \right\}$$

et **apprendre à partir de transitions sélectionnées au hasard** (dans le « replay buffer ») [1].

L'apprentissage se fait donc sur des données qui ne sont plus séquentielles puisque l'ordre d'apparition des transitions est mélangé [2] ce qui brise la corrélation des échantillons et nous ramène dans un état iid.

1. Cette technique s'appelle « Experience replay ». Voir Deep Q-Network | Ressource théorique en fin de section
2. On peu le faire parce que l'algorithme Q-learning est « OFF-policy »

⚠ Q-learning avec réseau neuronaux est instable

Causes 2 : Le « *TD-target* » :

- est non-stationnaire contrairement au contexte de l'apprentissage supervisé ;
- est corrélé avec l'approximateur $\widehat{Q}^{\pi}_{\theta}$ qu'on cherche à apprendre ;

$$L_k(\theta_k) = \mathbb{E}_{\mathbf{s}'\sim p(\cdot|\mathbf{s},\mathbf{a})}\left[\left(\underbrace{r(\mathbf{s},\mathbf{a},\mathbf{s}') + \gamma \max_{\mathbf{a}'\in\mathcal{A}(\mathbf{s}')} \widehat{Q}^{\pi}_{\theta}(\mathbf{s}',\mathbf{a}')}_{\textit{TD-target}} - \widehat{Q}^{\pi}_{\theta}(\mathbf{s},\mathbf{a})\right)^2\right]$$

Conséquence : $\widehat{Q}^{\pi}_{\theta}$ change lentement et π change potentiellement drastiquement

Solution : **Geler périodiquement l'approximateur du « TD target ».** Pratiquement, ça veut dire créer un deuxième réseau neuronaux apellé le « **target Q-network** » et « fetcher » les paramètres du « Q-network » à tout les N « timestep ».

$$\widehat{Q}^{\pi}_{\theta_{\text{target}}} \longleftarrow \widehat{Q}^{\pi}_{\theta}$$

- Diminue la possibilité d'oscillation ou de divergence de la politique ;
- Réduit la corrélation entre le « TD target » $r(\mathbf{s},\mathbf{a},\mathbf{s}') + \gamma \max_{\mathbf{a}'} \widehat{Q}^{\pi}_{\theta_{\text{target}}}(\mathbf{s}',\mathbf{a}')$; et l'estimateur $\widehat{Q}^{\pi}_{\theta}(\mathbf{s},\mathbf{a})$

⚠ Q-learning avec réseau neuronaux est instable

Causes 2 : Le « *TD-target* » :

- ▷ est non-stationnaire contrairement au contexte de l'apprentissage supervisé ;
- ▷ est corrélé avec l'approximateur $\widehat{Q}^{\pi}_{\theta}$ qu'on cherche à apprendre ;

$$L_k(\theta_k) = \mathbb{E}_{\mathbf{s}'\sim p(\cdot|\mathbf{s},\mathbf{a})}\left[\left(\underbrace{r(\mathbf{s},\mathbf{a},\mathbf{s}') + \gamma \max_{\mathbf{a}'\in\mathcal{A}(\mathbf{s}')} \widehat{Q}^{\pi}_{\theta}(\mathbf{s}',\mathbf{a}')}_{\textit{TD-target}} - \widehat{Q}^{\pi}_{\theta}(\mathbf{s},\mathbf{a})\right)^2\right]$$

Conséquence : $\widehat{Q}^{\pi}_{\theta}$ change lentement et π change potentiellement drastiquement

Solution : **Geler périodiquement l'approximateur du « TD target ».** Pratiquement, ça veut dire créer un deuxième réseau neuronaux apellé le « **target Q-network** » et « fetcher » les paramètres du « Q-network » à tout les N « timestep ».

$$\widehat{Q}^{\pi}_{\theta_{\text{target}}} \longleftarrow \widehat{Q}^{\pi}_{\theta}$$

- ▷ Diminue la possibilité d'oscillation ou de divergence de la politique ;
- ▷ Réduit la corrélation entre le « TD target » $r(\mathbf{s},\mathbf{a},\mathbf{s}') + \gamma \max_{\mathbf{a}'} \widehat{Q}^{\pi}_{\theta_{\text{target}}}(\mathbf{s}',\mathbf{a}')$; et l'estimateur $\widehat{Q}^{\pi}_{\theta}(\mathbf{s},\mathbf{a})$

⚠ Q-learning avec réseau neuronaux est **instable**

Causes 2 : Le « *TD-target* » :

- ▷ est non-stationnaire contrairement au contexte de l'apprentissage supervisé ;
- ▷ est corrélé avec l'approximateur $\widehat{Q}^{\pi}_{\theta}$ qu'on cherche à apprendre ;

$$L_k(\theta_k) \;=\; \mathbb{E}_{\mathbf{s}'\sim p(\cdot\,|\,\mathbf{s},\mathbf{a})}\left[\left(\underbrace{r(\mathbf{s},\mathbf{a},\mathbf{s}') + \gamma \max_{\mathbf{a}'\in\mathcal{A}(\mathbf{s}')} \widehat{Q}^{\pi}_{\theta}(\mathbf{s}',\mathbf{a}')}_{\textit{TD-target}} \;-\; \widehat{Q}^{\pi}_{\theta}(\mathbf{s},\mathbf{a})\right)^2\right]$$

Conséquence : $\widehat{Q}^{\pi}_{\theta}$ change lentement et π change potentiellement drastiquement

Solution : **Geler périodiquement l'approximateur du « TD target ».** Pratiquement, ça veut dire créer un deuxième réseau neuronaux apellé le « **target Q-network** » et « fetcher » les paramètres du « Q-network » à tout les N « timestep ».

$$\widehat{Q}^{\pi}_{\theta_{\text{target}}} \quad \longleftarrow \quad \widehat{Q}^{\pi}_{\theta}$$

- ▷ Diminue la possibilité d'oscillation ou de divergence de la politique ;
- ▷ Réduit la corrélation entre le « TD target » $r(\mathbf{s},\mathbf{a},\mathbf{s}') + \gamma \max_{\mathbf{a}'} \widehat{Q}^{\pi}_{\theta_{\text{target}}}(\mathbf{s}',\mathbf{a}')$; et l'estimateur $\widehat{Q}^{\pi}_{\theta}(\mathbf{s},\mathbf{a})$

Deep Q-Network *[DQN]*

Caractéristique : Apprentisage **EN-ligne** et **OFF-policy** (comme Q-learning approximatif)

Idées clés :

1. **Représenter** $Q^{\pi} \approx \widehat{Q}^{\pi}_{\theta}$ un réseau neuronaux de vecteur de paramètres θ (le « **Q-network** »)
2. Utiliser un « **target Q-network** » $\widehat{Q}^{\pi}_{\theta_{\text{target}}}$ cloner périodiquement de $\widehat{Q}^{\pi}_{\theta}$ pour réduire l'oscillation
3. Utiliser un « **replay buffer** » $\mathcal{D}$ pour briser la corrélation entre les échantillons
4. Définir une **fonction objectif** par *erreur quadratique moyenne* [*MSE*] et son **gradient** :

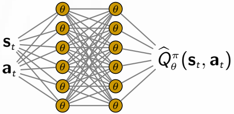
$$L_k(\theta_k) = \mathbb{E}_{(s,a,r,s')\sim\mathcal{U}(\mathcal{D})}\left[\left(y - \widehat{Q}^{\pi}_{\theta}(s,a)\right)^2 \Big|_{\theta=\theta_k}\right] \quad \text{avec } y = r(s,a,s') + \gamma \max_{a'\in\mathcal{A}(s')} \widehat{Q}^{\pi}_{\theta_{\text{target}}}(s',a')$$

$$\tfrac{1}{2}\nabla_{\theta_k} L(\theta_k) = \mathbb{E}_{(s,a,r,s')\sim\mathcal{U}(\mathcal{D})}\left[\left(r(s,a,s') + \gamma \max_{a'\in\mathcal{A}(s')} \widehat{Q}^{\pi}_{\theta_{\text{target}}}(s',a') - \widehat{Q}^{\pi}_{\theta}(s,a)\right) \nabla_{\theta_k} \widehat{Q}^{\pi}_{\theta}(s,a) \Big|_{\theta=\theta_k}\right]$$

5. **Optimiser** l'objectif par *descente de gradient stochastique* [*SGD*] :

$$\theta_{k+1} \longleftarrow \theta_k - \alpha \nabla_{\theta_k} L(\theta_k)$$

Deep Q-Network *[DQN]*



Caractéristique : Apprentisage **EN-ligne** et **OFF-policy** (comme Q-learning approximatif)

Idées clés :

1. **Représenter** $Q^{\pi} \approx \widehat{Q}^{\pi}_{\theta}$ un réseau neuronaux de vecteur de paramètres θ (le « **Q-network** »)
2. Utiliser un « **target Q-network** » $\widehat{Q}^{\pi}_{\theta_{\text{target}}}$ cloner périodiquement de $\widehat{Q}^{\pi}_{\theta}$ pour réduire l'oscillation
3. Utiliser un « **replay buffer** » $\mathcal{D}$ pour briser la corrélation entre les échantillons
4. Définir une **fonction objectif** par *erreur quadratique moyenne* [*MSE*] et son **gradient** :

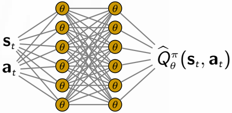
$$L_k(\theta_k) = \mathbb{E}_{(\mathbf{s},\mathbf{a},r,\mathbf{s}')\sim\mathcal{U}(\mathcal{D})}\left[\left(\mathbf{y} - \widehat{Q}^{\pi}_{\theta}(\mathbf{s},\mathbf{a})\right)^2\bigg|_{\theta=\theta_k}\right] \qquad \text{avec } \mathbf{y} = r(\mathbf{s},\mathbf{a},\mathbf{s}') + \gamma \max_{\mathbf{a}'\in\mathcal{A}(\mathbf{s}')} \widehat{Q}^{\pi}_{\theta_{\text{target}}}(\mathbf{s}',\mathbf{a}')$$

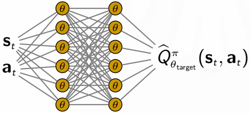
$$\frac{1}{2}\nabla_{\theta_k} L(\theta_k) = \mathbb{E}_{(\mathbf{s},\mathbf{a},r,\mathbf{s}')\sim\mathcal{U}(\mathcal{D})}\left[\left(r(\mathbf{s},\mathbf{a},\mathbf{s}') + \gamma \max_{\mathbf{a}'\in\mathcal{A}(\mathbf{s}')} \widehat{Q}^{\pi}_{\theta_{\text{target}}}(\mathbf{s}',\mathbf{a}') - \widehat{Q}^{\pi}_{\theta}(\mathbf{s},\mathbf{a})\right) \nabla_{\theta_k}\widehat{Q}^{\pi}_{\theta}(\mathbf{s},\mathbf{a})\bigg|_{\theta=\theta_k}\right]$$

5. **Optimiser** l'objectif par *descente de gradient stochastique* [*SGD*] :

$$\theta_{k+1} \longleftarrow \theta_k - \alpha\,\nabla_{\theta_k} L(\theta_k)$$

Deep Q-Network *[DQN]*





Caractéristique : Apprentisage **EN-ligne** et **OFF-policy** (comme Q-learning approximatif)

Idées clés :

1. **Représenter** $Q^\pi \approx \widehat{Q}^\pi_\theta$ un réseau neuronaux de vecteur de paramètres θ (le « **Q-network** »)
2. Utiliser un « **target Q-network** » $\widehat{Q}^\pi_{\theta_{\text{target}}}$ cloner périodiquement de $\widehat{Q}^\pi_\theta$ pour réduire l'oscillation
3. Utiliser un « **replay buffer** » $\mathcal{D}$ pour briser la corrélation entre les échantillons
4. Définir une **fonction objectif** par *erreur quadratique moyenne* [*MSE*] et son **gradient** :

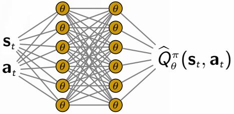
$$L_k(\theta_k) = \mathbb{E}_{(\mathrm{s},\mathrm{a},r,\mathrm{s}')\sim\mathcal{U}(\mathcal{D})}\left[\left(\mathrm{y} - \widehat{Q}^\pi_\theta(\mathrm{s},\mathrm{a})\right)^2\Big|_{\theta=\theta_k}\right] \quad \text{avec } \mathrm{y} = r(\mathrm{s},\mathrm{a},\mathrm{s}') + \gamma \max_{\mathrm{a}'\in\mathcal{A}(\mathrm{s}')} \widehat{Q}^\pi_{\theta_{\text{target}}}(\mathrm{s}',\mathrm{a}')$$

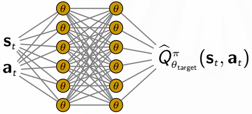
$$\frac{1}{2}\nabla_{\theta_k} L(\theta_k) = \mathbb{E}_{(\mathrm{s},\mathrm{a},r,\mathrm{s}')\sim\mathcal{U}(\mathcal{D})}\left[\left(r(\mathrm{s},\mathrm{a},\mathrm{s}') + \gamma \max_{\mathrm{a}'\in\mathcal{A}(\mathrm{s}')} \widehat{Q}^\pi_{\theta_{\text{target}}}(\mathrm{s}',\mathrm{a}') - \widehat{Q}^\pi_\theta(\mathrm{s},\mathrm{a})\right) \nabla_{\theta_k} \widehat{Q}^\pi_\theta(\mathrm{s},\mathrm{a})\Big|_{\theta=\theta_k}\right]$$

5. **Optimiser** l'objectif par *descente de gradient stochastique* [*SGD*] :

$$\theta_{k+1} \longleftarrow \theta_k - \alpha \nabla_{\theta_k} L(\theta_k)$$

Deep Q-Network *[DQN]*





Caractéristique : Apprentisage **EN-ligne** et **OFF-policy** (comme Q-learning approximatif)

Idées clés :

1. **Représenter** $Q^{\pi} \approx \widehat{Q}^{\pi}_{\theta}$ un réseau neuronaux de vecteur de paramètres θ (le « **Q-network** »)
2. Utiliser un « **target Q-network** » $\widehat{Q}^{\pi}_{\theta_{\text{target}}}$ cloner périodiquement de $\widehat{Q}^{\pi}_{\theta}$ pour réduire l'oscillation
3. Utiliser un « **replay buffer** » $\mathcal{D}$ pour briser la corrélation entre les échantillons
4. Définir une **fonction objectif** par *erreur quadratique moyenne* [*MSE*] et son **gradient** :

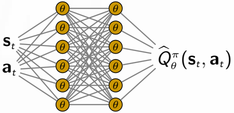
$$L_k(\theta_k) = \mathbb{E}_{(\mathbf{s},\mathbf{a},r,\mathbf{s}')\sim\mathcal{U}(\mathcal{D})}\left[\left(\mathbf{y} - \widehat{Q}^{\pi}_{\theta}(\mathbf{s},\mathbf{a})\right)^2\Big|_{\theta=\theta_k}\right] \qquad \text{avec } \mathbf{y} = r(\mathbf{s},\mathbf{a},\mathbf{s}') + \gamma \max_{\mathbf{a}'\in\mathcal{A}(\mathbf{s}')} \widehat{Q}^{\pi}_{\theta_{\text{target}}}(\mathbf{s}',\mathbf{a}')$$

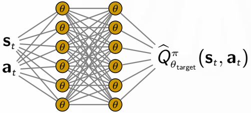
$$\frac{1}{2}\nabla_{\theta_k} L(\theta_k) = \mathbb{E}_{(\mathbf{s},\mathbf{a},r,\mathbf{s}')\sim\mathcal{U}(\mathcal{D})}\left[\left(r(\mathbf{s},\mathbf{a},\mathbf{s}') + \gamma \max_{\mathbf{a}'\in\mathcal{A}(\mathbf{s}')} \widehat{Q}^{\pi}_{\theta_{\text{target}}}(\mathbf{s}',\mathbf{a}') - \widehat{Q}^{\pi}_{\theta}(\mathbf{s},\mathbf{a})\right) \nabla_{\theta_k}\widehat{Q}^{\pi}_{\theta}(\mathbf{s},\mathbf{a})\Big|_{\theta=\theta_k}\right]$$

5. **Optimiser** l'objectif par *descente de gradient stochastique* [*SGD*] :

$$\theta_{k+1} \longleftarrow \theta_k - \alpha \nabla_{\theta_k} L(\theta_k)$$

Deep Q-Network [DQN]





Caractéristique : Apprentisage **EN-ligne** et **OFF-policy** (comme Q-learning approximatif)

Idées clés :

1. **Représenter** $Q^{\pi} \approx \widehat{Q}^{\pi}_{\theta}$ un réseau neuronaux de vecteur de paramètres θ (le « **Q-network** »)
2. Utiliser un « **target Q-network** » $\widehat{Q}^{\pi}_{\theta_{\text{target}}}$ cloner périodiquement de $\widehat{Q}^{\pi}_{\theta}$ pour réduire l'oscillation
3. Utiliser un « **replay buffer** » $\mathcal{D}$ pour briser la corrélation entre les échantillons
4. Définir une **fonction objectif** par *erreur quadratique moyenne* [*MSE*] et son **gradient** :

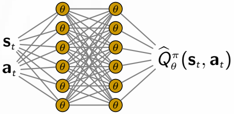
$$L_k(\theta_k) \;=\; \mathbb{E}_{(\mathbf{s},\mathbf{a},r,\mathbf{s}')\sim\mathcal{U}(\mathcal{D})}\left[\left(\mathbf{y} - \widehat{Q}^{\pi}_{\theta}(\mathbf{s},\mathbf{a}) \right)^2 \Big|_{\theta=\theta_k} \right] \qquad \text{avec } \mathbf{y} = r(\mathbf{s},\mathbf{a},\mathbf{s}') + \gamma \max_{\mathbf{a}'\in\mathcal{A}(\mathbf{s}')} \widehat{Q}^{\pi}_{\theta_{\text{target}}}(\mathbf{s}',\mathbf{a}')$$

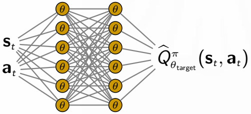
$$\tfrac{1}{2}\nabla_{\theta_k} L(\theta_k) \;=\; \mathbb{E}_{(\mathbf{s},\mathbf{a},r,\mathbf{s}')\sim\mathcal{U}(\mathcal{D})}\left[\left(r(\mathbf{s},\mathbf{a},\mathbf{s}') + \gamma \max_{\mathbf{a}'\in\mathcal{A}(\mathbf{s}')} \widehat{Q}^{\pi}_{\theta_{\text{target}}}(\mathbf{s}',\mathbf{a}') - \widehat{Q}^{\pi}_{\theta}(\mathbf{s},\mathbf{a}) \right) \nabla_{\theta_k} \widehat{Q}^{\pi}_{\theta}(\mathbf{s},\mathbf{a}) \Big|_{\theta=\theta_k} \right]$$

5. **Optimiser** l'objectif par *descente de gradient stochastique* [*SGD*] :

$$\theta_{k+1} \;\longleftarrow\; \theta_k - \alpha \nabla_{\theta_k} L(\theta_k)$$

Deep Q-Network [DQN]





Caractéristique : Apprentisage **EN-ligne** et **OFF-policy** (comme Q-learning approximatif)

Idées clés :

1. **Représenter** $Q^{\pi} \approx \widehat{Q}^{\pi}_{\theta}$ un réseau neuronaux de vecteur de paramètres θ (le « **Q-network** »)
2. Utiliser un « **target Q-network** » $\widehat{Q}^{\pi}_{\theta_{\text{target}}}$ cloner périodiquement de $\widehat{Q}^{\pi}_{\theta}$ pour réduire l'oscillation
3. Utiliser un « **replay buffer** » $\mathcal{D}$ pour briser la corrélation entre les échantillons
4. Définir une **fonction objectif** par *erreur quadratique moyenne* [*MSE*] et son **gradient** :

$$L_k(\theta_k) = \mathbb{E}_{(\mathbf{s},\mathbf{a},r,\mathbf{s}')\sim\mathcal{U}(\mathcal{D})}\left[\left(\mathbf{y} - \widehat{Q}^{\pi}_{\theta}(\mathbf{s},\mathbf{a}) \right)^2 \Big|_{\theta=\theta_k} \right] \quad \text{avec } \mathbf{y} = r(\mathbf{s},\mathbf{a},\mathbf{s}') + \gamma \max_{\mathbf{a}'\in\mathcal{A}(\mathbf{s}')} \widehat{Q}^{\pi}_{\theta_{\text{target}}}(\mathbf{s}',\mathbf{a}')$$

$$\tfrac{1}{2}\nabla_{\theta_k} L(\theta_k) = \mathbb{E}_{(\mathbf{s},\mathbf{a},r,\mathbf{s}')\sim\mathcal{U}(\mathcal{D})}\left[\left(r(\mathbf{s},\mathbf{a},\mathbf{s}') + \gamma \max_{\mathbf{a}'\in\mathcal{A}(\mathbf{s}')} \widehat{Q}^{\pi}_{\theta_{\text{target}}}(\mathbf{s}',\mathbf{a}') - \widehat{Q}^{\pi}_{\theta}(\mathbf{s},\mathbf{a}) \right) \nabla_{\theta_k} \widehat{Q}^{\pi}_{\theta}(\mathbf{s},\mathbf{a}) \Big|_{\theta=\theta_k} \right]$$

5. **Optimiser** l'objectif par *descente de gradient stochastique* [*SGD*] :

$$\theta_{k+1} \longleftarrow \theta_k - \alpha \nabla_{\theta_k} L(\theta_k)$$

Algorithme | Deep Q-Network

Soit la politique exploratoire $\mu(\cdot|\mathbf{s})$, la politique cible $\pi(\mathbf{s}) \doteq \pi_{greedy}$, le « learning rate » $\alpha \in]0,1]$, le « global step » $k \leftarrow 0$, le « replay buffer » $\mathcal{D}$, le « Q-network » $\widehat{Q}^{\pi}_{\theta}$ et le « Target-network » $\widehat{Q}^{\pi}_{\theta_{\text{target}}}$

Répéter *pour chaque trajectoire* $\mathcal{T}$ ***jusqu'à convergence*** *vers* $\pi^\star$ **:**

Répéter *pour chaque « timestep »* $t = 0, 1, 2, \ldots, t, t{+}1, \ldots, \mathrm{T}$ *de la trajectoire* $\mathcal{T}$ **:**

$r^{(k)}_{t+1}, \mathbf{s}^{(k)}_{t+1} \quad \longleftarrow \quad$ exécuter $\mathbf{a}_t \sim \mu(\cdot|\mathbf{s}_t)$ dans l'environnement et observer la réaction

$$\mathcal{D} \quad \longleftarrow \quad \mathcal{D} \cup \left\{ \left(\mathbf{s}_t, \mathbf{a}_t, r^{(k)}_{t+1}, \mathbf{s}^{(k)}_{t+1}\right) \right\}$$

Prélever une « mini-batch » de transitions $\left\{ \left(\mathbf{s}^{(b)}_t, \mathbf{a}^{(b)}_t, r^{(b)}_{t+1}, \mathbf{s}^{(b)}_{t+1}\right) \in \mathcal{U}(\mathcal{D}) \right\}$

$$\mathbf{y} \quad = \quad r^{(b)}_{t+1} + \gamma \max_{\mathbf{a}' \in \mathcal{A}(\mathbf{s}_t)} \widehat{Q}^{\pi}_{\theta_{\text{target}}}(\mathbf{s}^{(b)}_{t+1}, \mathbf{a}')$$

$$\theta_{k+1} \quad \longleftarrow \quad \theta_k - \alpha \sum_{b \in \mathcal{U}(\mathcal{D})} \left(\mathbf{y} - \widehat{Q}^{\pi}_{\theta_k}\left(\mathbf{s}^{(b)}_t, \mathbf{a}^{(b)}_t\right) \right) \frac{\partial}{\partial \theta_k} \widehat{Q}^{\pi}_{\theta_k}(\mathbf{s}^{(b)}_t, \mathbf{a}^{(b)}_t)$$

$\mathbf{s}_t \leftarrow \mathbf{s}^{(k)}_{t+1}$ et $k \leftarrow k + 1$

Lorsque « timesteps » $t = N$ **:** $\theta_{\text{target}} \leftarrow \theta_k$

Algorithme | Deep Q-Network

Soit la politique exploratoire $\mu(\cdot|\mathbf{s})$, la politique cible $\pi(\mathbf{s}) \doteq \pi_{greedy}$, le « learning rate » $\alpha \in]0,1]$, le « global step » $k \leftarrow 0$, le « replay buffer » $\mathcal{D}$, le « Q-network » $\widehat{Q}^{\pi}_{\theta}$ et le « Target-network » $\widehat{Q}^{\pi}_{\theta_{\text{target}}}$

Répéter *pour chaque trajectoire* $\mathcal{T}$ ***jusqu'à convergence*** *vers* $\pi^\star$ **:**

Répéter *pour chaque « timestep »* $t = 0, 1, 2, \ldots, t, t{+}1, \ldots, \mathrm{T}$ *de la trajectoire* $\mathcal{T}$ **:**

$r^{(k)}_{t+1}, \mathbf{s}^{(k)}_{t+1} \longleftarrow$ exécuter $\mathbf{a}_t \sim \mu(\cdot|\mathbf{s}_t)$ dans l'environnement et observer la réaction

$$\mathcal{D} \longleftarrow \mathcal{D} \cup \left\{ \left(\mathbf{s}_t, \mathbf{a}_t, r^{(k)}_{t+1}, \mathbf{s}^{(k)}_{t+1} \right) \right\}$$

Prélever une « mini-batch » de transitions $\left\{ \left(\mathbf{s}^{(b)}_t, \mathbf{a}^{(b)}_t, r^{(b)}_{t+1}, \mathbf{s}^{(b)}_{t+1} \right) \in \mathcal{U}(\mathcal{D}) \right\}$

$$\mathbf{y} = r^{(b)}_{t+1} + \gamma \max_{\mathbf{a}' \in \mathcal{A}(\mathbf{s}_t)} \widehat{Q}^{\pi}_{\theta_{\text{target}}}(\mathbf{s}^{(b)}_{t+1}, \mathbf{a}')$$

$$\theta_{k+1} \longleftarrow \theta_k - \alpha \sum_{b \in \mathcal{U}(\mathcal{D})} \left(\mathbf{y} - \widehat{Q}^{\pi}_{\theta_k}\left(\mathbf{s}^{(b)}_t, \mathbf{a}^{(b)}_t\right) \right) \frac{\partial}{\partial \theta_k} \widehat{Q}^{\pi}_{\theta_k}(\mathbf{s}^{(b)}_t, \mathbf{a}^{(b)}_t)$$

$\mathbf{s}_t \leftarrow \mathbf{s}^{(k)}_{t+1}$ et $k \leftarrow k+1$

Lorsque « timesteps » $t = N$ **:** $\theta_{\text{target}} \leftarrow \theta_k$

Pro :

- « . . . DQN **excels at reactive games**. More recent versions are now better at game involving planning . . . » [1] ;
- Est efficients au niveau de l'utilisation des échantillons ;

Con :

- Peu être **très long avant de voir signe de vie** pendant l'entraînement ;
- Peut être **difficile à stabiliser** ;
- DQN ne fonctionne pas sur les espaces d'actions continues (dans sa forme classique) ;

Autre architecture et amélioration reliée à DQN :

- *Double DQN*[1]
- *Dueling DQN*[2]
- *Q-learning* avec *N-step returns*[3]
- *Prioritized experience replay*[4]
- Pour espace d'action continue : ***D**eep **D**eterministic **P**olicy **G**radient* [2] [5]

1. Vlad Mnih (Deepmind), Deep RL Bootcamp Lecture 3: Deep Q-Networks, Berkeley, August 2017
2. **DDPG** est un mélange de méthode *basée valeur* et *recherche de politique* qui utilise un autre réseau neuronaux pour représenter une politique $\mu_\phi(\mathbf{s}) \approx \arg\max_{\mathbf{a}} Q^\pi_\theta(\mathbf{s}, \mathbf{a})$

Pro :
- « . . . DQN **excels at reactive games**. More recent versions are now better at game involving planning . . . » [1] ;
- Est efficients au niveau de l'utilisation des échantillons ;

Con :
- Peu être **très long avant de voir signe de vie** pendant l'entraînement ;
- Peut être **difficile à stabiliser** ;
- DQN ne fonctionne pas sur les espaces d'actions continues (dans sa forme classique) ;

Autre architecture et amélioration reliée à DQN :

- ***Double DQN*[1]** $\longleftarrow$ vue à la prochaine section
- *Dueling DQN*[2]
- *Q-learning* avec *N-step returns*[3]
- *Prioritized experience replay*[4]
- Pour espace d'action continue : ***D**eep **D**eterministic **P**olicy **G**radient* [2] [5]

1. Vlad Mnih (Deepmind), Deep RL Bootcamp Lecture 3: Deep Q-Networks, Berkeley, August 2017
2. **DDPG** est un mélange de méthode *basée valeur* et *recherche de politique* qui utilise un autre réseau neuronaux pour représenter une politique $\mu_\phi(\mathbf{s}) \approx \arg\max_{\mathbf{a}} Q^\pi_\theta(\mathbf{s}, \mathbf{a})$

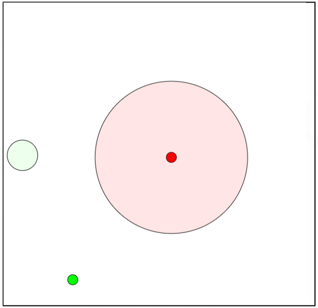
Deep Q-Network [DQN]

Exemple interactif

Deep Q-Network dans l'environnement *PuckWorld/WaterWorld*

▸ Exemple interactif: *PuckWorld*

▸ Exemple interactif: *WaterWorld*



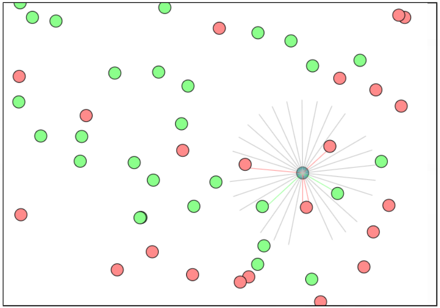


Image : REINFORCEjs, Stanford

- REINFORCEjs ▸ Projet ▸ GitHub

DRL et méthode basée sur les valeurs

Double Deep Q-Network

Biais de maximisation

⚠ **Problème :** Le « TD target » du *DQN* à tendance à être **trop optimiste** dans ses estimations de la Q-valeur du prochain état.

Cause : *DQN* calcule un **maximum sur un approximateur qui est imparfait et bruité** (particulièrement au début de l'entraînement)

$$\mathbf{y} = r(\mathbf{s}, \mathbf{a}, \mathbf{s}') + \gamma \max_{\mathbf{a}' \in \mathcal{A}(\mathbf{s}')} \widehat{Q}^{\pi}_{\theta_{\text{target}}}(\mathbf{s}', \mathbf{a}')$$

Supposons la vrai Q-fonction

$$Q(\mathbf{s}, \mathbf{a}) = 0 \quad \forall (\mathbf{s}, \mathbf{a}) \in \mathcal{S} \times \mathcal{A}$$

et son estimateur

$$\widehat{Q}^{\pi}(\mathbf{s}, \mathbf{a}) \in [-1, 1]$$

alors le maximum de la vraie valeur sera 0 et le maximum de son estimateur sera **biaisé de façon positive.** [1]

Solution : Utiliser un réseau neuronaux **différent pour l'évaluation** des Q-valeur **de celui pour la sélection** d'action.

1. Exemple tiré de « Reinforcement Learning : An introduction », section 6.7 « Maximization Bias and Double Learning » par Sutton & Barto [6]

Biais de maximisation

⚠ **Problème :** Le « TD target » du *DQN* à tendance à être **trop optimiste** dans ses estimations de la Q-valeur du prochain état.

Cause : *DQN* calcule un **maximum sur un approximateur qui est imparfait et bruité**
(particulièrement au début de l'entraînement)

$$\mathbf{y} = r(\mathbf{s}, \mathbf{a}, \mathbf{s}') + \gamma \max_{\mathbf{a}' \in \mathcal{A}(\mathbf{s}')} \widehat{Q}^{\pi}_{\theta_{\text{target}}}(\mathbf{s}', \mathbf{a}')$$

Supposons la vrai Q-fonction

$$Q(\mathbf{s}, \mathbf{a}) = 0 \quad \forall (\mathbf{s}, \mathbf{a}) \in \mathcal{S} \times \mathcal{A}$$

et son estimateur

$$\widehat{Q}^{\pi}(\mathbf{s}, \mathbf{a}) \in [-1, 1]$$

alors le maximum de la vraie valeur sera 0 et le maximum de son estimateur sera **biaisé de façon positive.** [1]

Solution : Utiliser un réseau neuronaux **différent pour l'évaluation** des Q-valeur **de celui pour la sélection** d'action.

1. Exemple tiré de « Reinforcement Learning : An introduction », section 6.7 « Maximization Bias and Double Learning » par Sutton & Barto [6]

Biais de maximisation

⚠ **Problème :** Le « TD target » du *DQN* à tendance à être **trop optimiste** dans ses estimations de la Q-valeur du prochain état.

Cause : *DQN* calcule un **maximum sur un approximateur qui est imparfait et bruité** (particulièrement au début de l'entraînement)

$$\mathbf{y} = r(\mathbf{s}, \mathbf{a}, \mathbf{s}') + \gamma \max_{\mathbf{a}' \in \mathcal{A}(\mathbf{s}')} \widehat{Q}^{\pi}_{\theta_{\text{target}}}(\mathbf{s}', \mathbf{a}')$$

Supposons la vrai Q-fonction

$$Q(\mathbf{s}, \mathbf{a}) = 0 \quad \forall (\mathbf{s}, \mathbf{a}) \in \mathcal{S} \times \mathcal{A}$$

et son estimateur

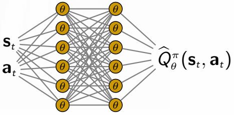
$$\widehat{Q}^{\pi}(\mathbf{s}, \mathbf{a}) \in [-1, 1]$$

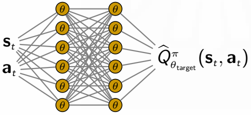
alors le maximum de la vraie valeur sera 0 et le maximum de son estimateur sera **biaisé de façon positive.** [1]

Solution : Utiliser un réseau neuronaux **différent pour l'évaluation** des Q-valeur **de celui pour la sélection** d'action.

1. Exemple tiré de « Reinforcement Learning : An introduction », section 6.7 « Maximization Bias and Double Learning » par Sutton & Barto [6]

Double Deep Q-Network *[DDQN]*





Idées clés :

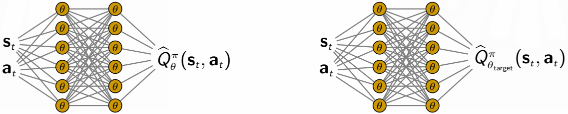
1. Extension de DQN ;
2. Utiliser un **réseau différent pour la sélection d'action** et l'évaluation de Q-valeur dans le « TD target »

$$L_k(\theta_k) \;=\; \mathbb{E}_{(\mathbf{s},\mathbf{a},r,\mathbf{s}')\sim\mathcal{U}(\mathcal{D})}\left[\left(\mathbf{y} \;-\; \widehat{Q}^{\pi}_{\theta}(\mathbf{s},\mathbf{a})\right)^2\Big|_{\theta=\theta_k}\right]$$

avec le « TD target » $\mathbf{y} \;=\; r(\mathbf{s},\mathbf{a},\mathbf{s}') + \gamma\, \widehat{Q}^{\pi}_{\theta_{\text{target}}}\left(\mathbf{s}', \underset{\mathbf{a}'\in\mathcal{A}(\mathbf{s}')}{\arg\max}\, \widehat{Q}^{\pi}_{\theta}\left(\mathbf{s}',\mathbf{a}'\right)\right)$

Pro :
- Amélioration efficace et simple à implémenter ;
- Aucun effet négatif ;

Double Deep Q-Network *[DDQN]*



Idées clés :

1. Extension de DQN ;
2. Utiliser un **réseau différent pour** la sélection d'action et **l'évaluation de Q-valeur** dans le « TD target »

$$L_k(\theta_k) = \mathbb{E}_{(\mathbf{s},\mathbf{a},r,\mathbf{s}')\sim\mathcal{U}(\mathcal{D})}\left[\left(\mathbf{y} - \widehat{Q}^{\pi}_{\theta}(\mathbf{s},\mathbf{a}) \right)^2 \Big|_{\theta=\theta_k} \right]$$

avec le « TD target » $\mathbf{y} = r(\mathbf{s},\mathbf{a},\mathbf{s}') + \gamma\, \widehat{Q}^{\pi}_{\theta_{\text{target}}}\left(\mathbf{s}', \arg\max_{\mathbf{a}'\in\mathcal{A}(\mathbf{s}')} \widehat{Q}^{\pi}_{\theta}\left(\mathbf{s}', \mathbf{a}'\right)\right)$

Pro :

- Amélioration efficace et simple à implémenter ;
- Aucun effet négatif ;

Ressource théorique

Experience replay

- « Reinforcement learning for robots using neural networks » ▸ Publication
  par Lin, L.-J. [7]

Deep Q-network

- « Playing Atari with Deep Reinforcement Learning » ▸ Publication
  par Mnih et al, 2013 [8]
- « Human-level control through deep reinforcement learning », Nature Vol 518 ▸ Publication
  par Mnih et al, 2015 [9]

Double DQN

- « Deep Reinforcement Learning with Double Q-Learning » ▸ Publication
  par Hasselt et al, 2015 [1]
- « Reinforcement Learning : An introduction », section 6.7 « Maximization Bias and Double Learning » [6] ▸ Publication
  par Sutton & Barto

Méthode par recherche de politique

Méthode par recherche de politique (en général)

Caractéristiques :

- **Apprends explicitement la politique $\pi^\star$** sans passer par V^π ou Q^π ;
- C'est un **problème d'optimisation** $\Longrightarrow$ on doit définir un objectif et choisir une méthode d'optimisation ;

Idées clés :

1. **Représenter** $\pi(\mathbf{a}|\mathbf{s}) \approx \hat{\pi}_\theta(\mathbf{a}|\mathbf{s})$ un approximateur de fonction de paramètres θ ;
2. **Définir** la fonction objectif de façon à **mesurer la qualité de la politique** $\hat{\pi}_\theta$

$$J(\theta) = \mathbb{E}_{\tau\sim\pi_\theta}\left[\,G(\tau)\,\right] \qquad \langle \text{ Ex. du cas épisodique } \rangle$$

3. **Trouver le paramètre** $\theta^\star$ qui maximise $J(\theta)$;

★ | L'objectif des méthodes par recherche de politique en RL

$$\begin{aligned}\theta^\star &= \arg\max_\theta J(\theta)\\ &= \arg\max_\theta \mathbb{E}_{\tau\sim\pi_\theta}\left[\,G(\tau)\,\right]\\ &= \arg\max_\theta \sum_{t=1}^{T} \mathbb{E}_{(\mathbf{s}_t,\mathbf{a}_t,\mathbf{s}_{t+1})\sim\pi_\theta(\tau)}\left[r(\mathbf{s}_t,\mathbf{a}_t,\mathbf{s}_{t+1})\right]\end{aligned}$$

Méthode par recherche de politique (en général)

Caractéristiques :

- **Apprends explicitement la politique $\pi^\star$** sans passer par V^π ou Q^π ;
- C'est un **problème d'optimisation** $\Longrightarrow$ on doit définir un objectif et choisir une méthode d'optimisation ;

Idées clés :

1. **Représenter** $\pi(\mathbf{a}|\mathbf{s}) \approx \hat{\pi}_\theta(\mathbf{a}|\mathbf{s})$ un approximateur de fonction de paramètres θ ;
2. **Définir** la fonction objectif de façon à **mesurer la qualité de la politique $\hat{\pi}_\theta$**

$$J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta}[\,G(\tau)\,] \qquad \langle \text{ Ex. du cas épisodique } \rangle$$

3. **Trouver le paramètre $\theta^\star$** qui maximise $J(\theta)$;

★ | **L'objectif des méthodes par recherche de politique en RL**

$$\begin{aligned} \theta^\star &= \arg\max_\theta J(\theta) \\ &= \arg\max_\theta \mathbb{E}_{\tau \sim \pi_\theta}[\,G(\tau)\,] \\ &= \arg\max_\theta \sum_{t=1}^{T} \mathbb{E}_{(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1}) \sim \pi_\theta(\tau)} [r(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}_{t+1})] \end{aligned}$$

Méthode par recherche de politique (en général)

Caractéristiques :

- **Apprends explicitement la politique $\pi^\star$** sans passer par V^π ou Q^π ;
- C'est un **problème d'optimisation** $\Longrightarrow$ on doit définir un objectif et choisir une méthode d'optimisation ;

Idées clés :

1. **Représenter** $\pi(\mathbf{a}|\mathbf{s}) \approx \hat{\pi}_\theta(\mathbf{a}|\mathbf{s})$ un approximateur de fonction de paramètres θ ;
2. **Définir** la fonction objectif de façon à **mesurer la qualité de la politique** $\hat{\pi}_\theta$

$$J(\theta) = \mathbb{E}_{\tau\sim\pi_\theta}[\,G(\tau)\,] \qquad \langle \text{ Ex. du cas épisodique } \rangle$$

3. **Trouver le paramètre $\theta^\star$** qui maximise $J(\theta)$;

★ | **L'objectif des méthodes par recherche de politique en RL**

$$\begin{aligned}
\theta^\star &= \arg\max_\theta J(\theta) \\
&= \arg\max_\theta \mathbb{E}_{\tau\sim\pi_\theta}[\,G(\tau)\,] \\
&= \arg\max_\theta \sum_{t=1}^{T} \mathbb{E}_{(\mathbf{s}_t,\mathbf{a}_t,\mathbf{s}_{t+1})\sim\pi_\theta(\tau)}[r(\mathbf{s}_t,\mathbf{a}_t,\mathbf{s}_{t+1})]
\end{aligned}$$

Méthode par recherche de politique (en général)

Caractéristiques :

- **Apprends explicitement la politique $\pi^\star$** sans passer par V^π ou Q^π ;
- C'est un **problème d'optimisation** $\Longrightarrow$ on doit définir un objectif et choisir une méthode d'optimisation ;

Idées clés :

1. **Représenter** $\pi(\mathbf{a}|\mathbf{s}) \approx \hat{\pi}_\theta(\mathbf{a}|\mathbf{s})$ un approximateur de fonction de paramètres θ ;
2. **Définir** la fonction objectif de façon à **mesurer la qualité de la politique $\hat{\pi}_\theta$**

$$J(\theta) = \mathbb{E}_{\tau\sim\pi_\theta}[\,G(\tau)\,]$$

⟨ Ex. du cas épisodique ⟩

3. **Trouver le paramètre $\theta^\star$** qui maximise $J(\theta)$;

★ | L'objectif des méthodes par recherche de politique en RL

$$\begin{aligned}\theta^\star &= \arg\max_\theta J(\theta)\\ &= \arg\max_\theta \mathbb{E}_{\tau\sim\pi_\theta}[\,G(\tau)\,]\\ &= \arg\max_\theta \sum_{t=1}^{T} \mathbb{E}_{(\mathbf{s}_t,\mathbf{a}_t,\mathbf{s}_{t+1})\sim\pi_\theta(\tau)}\left[r(\mathbf{s}_t,\mathbf{a}_t,\mathbf{s}_{t+1})\right]\end{aligned}$$

Méthode par recherche de politique (en général)

Pro :
- Efficace sur les **espaces en haute dimension** ;
- Capable de manipuler des **espaces d'actions continues** ;
- Capable d'apprendre une politique stochastique ;

Con : **Converge vers un optimum local**[1] (en général) ;

1. Voir UCL Course on RL, Lecture 7: Policy Gradient Methods par David Silver

Méthode par recherche de politique (en général)

Remarque sur les méthodes d'optimisation de politique

Le problème d'optimisation de politique peut être résolu selon deux types d'approches :

- les approches **utilisant le gradient** de la politique [*Policy gradient*]
- les approches **sans gradient** [*Gradient-free*]
  - « Hill climbing », « Simulated Annealing »
  - Algorithme génétique, stratégie évolutive [1] [*ES*]
  - Entropy croisé [*CEM*]
  - ...

Prenez note que l'approche « *Gradient-free* » peut être plus approprié dans certaine circonstance

par exemple : quand la politique est non différentiable

1. Voir l'excellent post « Evolution Strategies » par Lilian Weng ainsi que « A Visual Guide to Evolution Strategies » par David Ha

Méthode par recherche de politique (en général)

Remarque sur les méthodes d'optimisation de politique

Le problème d'optimisation de politique peut être résolu selon deux types d'approches :

- les approches **utilisant le gradient** de la politique [*Policy gradient*] $\Longleftarrow$ **ce que nous allons voir**
- les approches **sans gradient** [*Gradient-free*]
  - « Hill climbing », « Simulated Annealing »
  - Algorithme génétique, stratégie évolutive [1] [*ES*]
  - Entropy croisé [*CEM*]
  - ...

Prenez note que l'approche « *Gradient-free* » peut être plus approprié dans certaine circonstance
par exemple : quand la politique est non différentiable

1. Voir l'excellent post « Evolution Strategies » par Lilian Weng ainsi que « A Visual Guide to Evolution Strategies » par David Ha

DRL et méthode *Policy Gradient*

Méthode *Policy Gradient* [*Gradient based policy search*]

Assomption :

1. La politique $\hat{\pi}_\theta$ est **différentiable** par rapport à son paramètre θ ; [1]
2. On sait comment **résoudre le gradient** $\nabla_\theta \hat{\pi}_\theta(\mathbf{a}|\mathbf{s})$;

★ **Remarque :** **On peut utiliser n'importe quel politique** dans la mesure où elle respecte l'assomption de différentiabilité. En pratique on utilise une politique stochastique à cause de ses propriétés.

1. Voir « Reinforcement Learning : An introduction », section 13.1 « Policy Approximation and its Advantages » par Sutton & Barto [6]

Méthode *Policy Gradient* [*Gradient based policy search*]

Politique stochastique paramétrée $\pi_\theta(\mathbf{a}|\mathbf{s})$

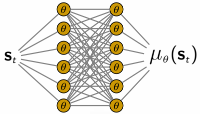
$$\pi_\theta(\mathbf{a}|\mathbf{s}) \;=\; \mathbb{P}\,[\,A_t = \mathbf{a}, S_t = \mathbf{s}\,|\,\theta\,] \qquad \text{avec le vecteur de paramètres } \theta$$

Propriétés :

- **fonction lisse** $\Longrightarrow$ continue $\Longrightarrow$ différentiable ;
- capable de manipuler des espaces d'**actions continues** ;
- capable de gérer les cas où deux actions sont de valeur égale ;

Exemple de politique stochastique paramétré compatible avec les espaces d'actions continues :

$$\pi_\theta(\mathbf{a}|\mathbf{s}) \;=\; \mathcal{N}\left(\,\mu_\theta(\mathbf{s}), \Sigma^2\,\right) \;=\; \frac{1}{\sqrt{2\pi\Sigma^2}}\exp\left(\frac{(\,\mathbf{a}-\mu_\theta(\mathbf{s})\,)^2}{\Sigma^2}\right)$$

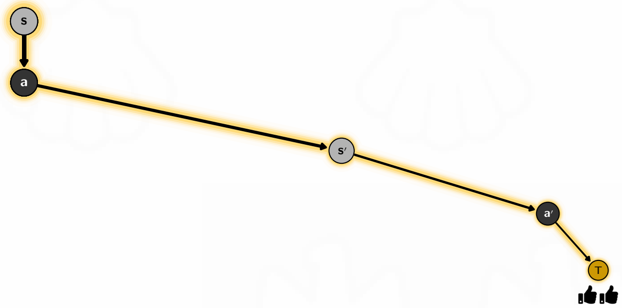


REINFORCE

REINFORCE [*Monte-Carlo Policy Gradient*]

« It's a formalization of the notion of trial & error » – Sergey Levine

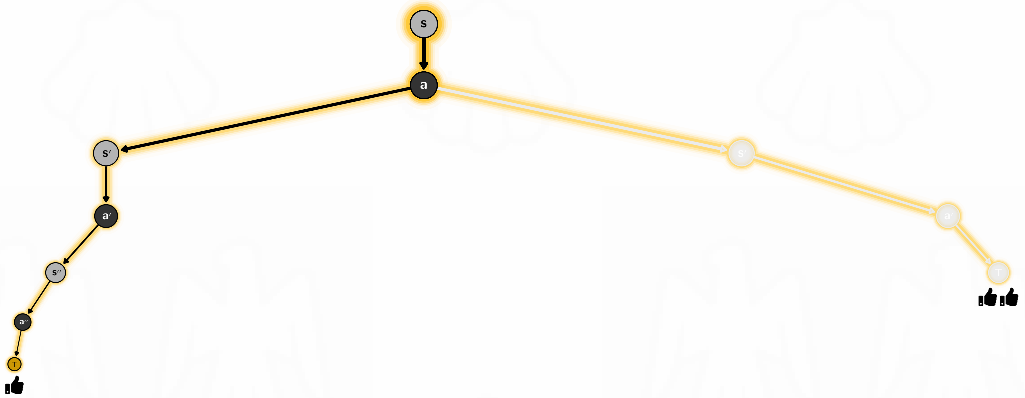
Idée générale : Attribuer un poids à chaque échantillon de trajectoire en fonction de sa qualité.
(Plus le retour est élevé, plus le poids est gros).



REINFORCE [*Monte-Carlo Policy Gradient*]

« It's a formalization of the notion of trial & error » – Sergey Levine

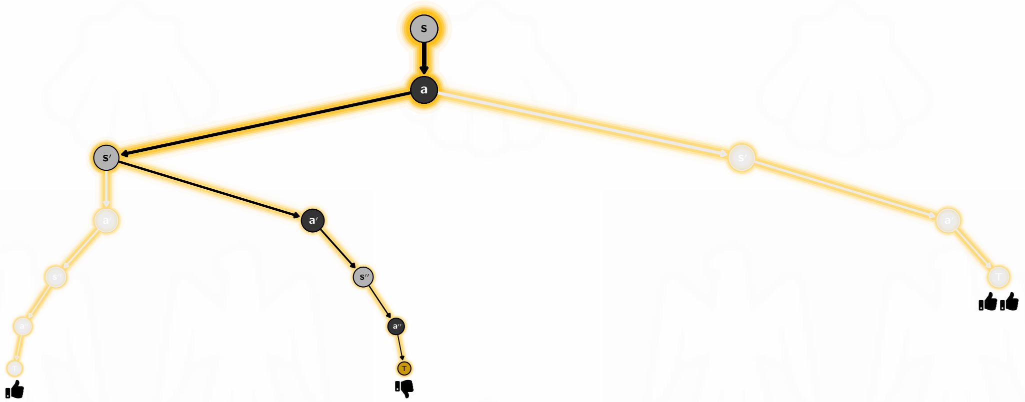
Idée générale : Attribuer un poids à chaque échantillon de trajectoire en fonction de sa qualité.
(Plus le retour est élevé, plus le poids est gros).



REINFORCE [*Monte-Carlo Policy Gradient*]

« It's a formalization of the notion of trial & error » – Sergey Levine

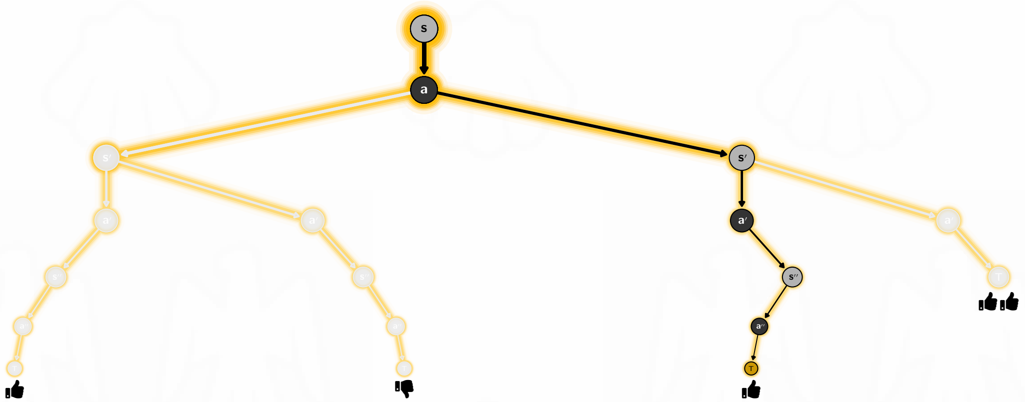
Idée générale : Attribuer un poids à chaque échantillon de trajectoire en fonction de sa qualité.
(Plus le retour est élevé, plus le poids est gros).



REINFORCE [*Monte-Carlo Policy Gradient*]

« It's a formalization of the notion of trial & error » – Sergey Levine

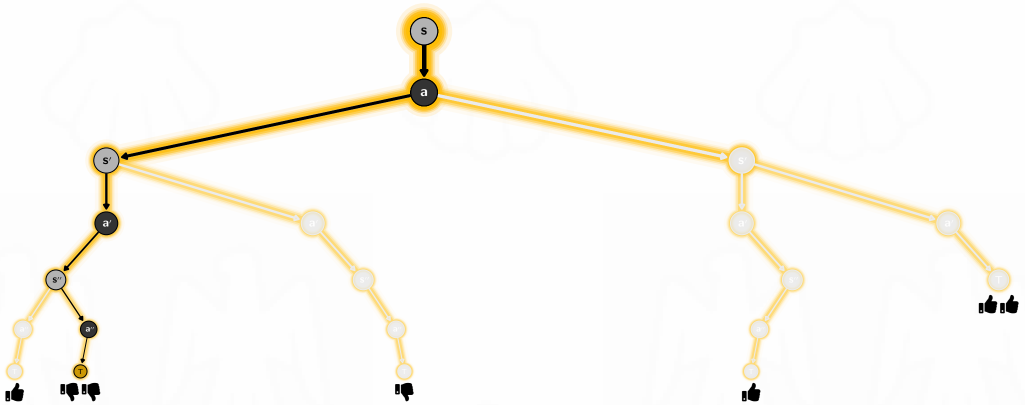
Idée générale : Attribuer un poids à chaque échantillon de trajectoire en fonction de sa qualité.
(Plus le retour est élevé, plus le poids est gros).



REINFORCE [*Monte-Carlo Policy Gradient*]

« It’s a formalization of the notion of trial & error » – Sergey Levine

Idée générale : Attribuer un poids à chaque échantillon de trajectoire en fonction de sa qualité.
(Plus le retour est élevé, plus le poids est gros).



REINFORCE [*Monte-Carlo Policy Gradient*]

Enjeux 1 : On veut résoudre $\nabla_\theta J(\theta)$ pour savoir **dans quelle direction faire l'ascension du gradient** dans le but d'améliorer π_θ ;

Étant donné l'assomption que la politique est différentiable, on procède par différentiation directe de π_θ ;

Différentiation directe de la politique

$$
\begin{aligned}
J(\theta) &= \mathbb{E}_{\tau\sim\pi_\theta(\tau)}\left[\, G(\tau)\,\right] \\
\nabla_\theta J(\theta) &= \nabla_\theta\, \mathbb{E}_{\tau\sim\pi_\theta(\tau)}\left[\, G(\tau)\,\right] \\
&= \int \nabla_\theta\, \pi_\theta(\tau)\, G(\tau)\, d\tau \\
&= \ldots \\
&= \mathbb{E}_{\tau\sim\pi_\theta(\tau)}\left[\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(\mathbf{a}_t|\mathbf{s}_t)\ G(\tau)\right]
\end{aligned}
$$

⟨ Voir l'annexe *Complément théorique* pour la dérivation complète ⟩

Observation : Ce résultat montre qu'on n'a **pas besoin du modèle** pour calculer $\nabla_\theta J(\theta)$. On a seulement besoin de la politique π_θ et de la fonction de récompense $r(\mathbf{s},\mathbf{a})$

REINFORCE [*Monte-Carlo Policy Gradient*]

Enjeux 1 : On veut résoudre $\nabla_\theta J(\theta)$ pour savoir **dans quelle direction faire l'ascension du gradient** dans le but d'améliorer π_θ ;

Étant donné l'assomption que la politique est différentiable, on procède par différentiation directe de π_θ ;

Différentiation directe de la politique

$$
\begin{aligned}
J(\theta) &= \mathbb{E}_{\tau\sim\pi_\theta(\tau)}\left[\,G(\tau)\,\right] \\
\nabla_\theta J(\theta) &= \nabla_\theta\; \mathbb{E}_{\tau\sim\pi_\theta(\tau)}\left[\,G(\tau)\,\right] \\
&= \int \nabla_\theta\, \pi_\theta(\tau)\; G(\tau)\; d\tau \\
&= \ldots \\
&= \mathbb{E}_{\tau\sim\pi_\theta(\tau)}\left[\textstyle\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(\mathbf{a}_t|\mathbf{s}_t)\;\; G(\tau)\right]
\end{aligned}
$$

⟨ Voir l'annexe *Complément théorique* pour la dérivation complète ⟩

Observation : Ce résultat montre qu'on n'a **pas besoin du modèle** pour calculer $\nabla_\theta J(\theta)$. On a seulement besoin de la politique π_θ et de la fonction de récompense $r(\mathbf{s}, \mathbf{a})$

REINFORCE [*Monte-Carlo Policy Gradient*]

Enjeux 2 : Avant de pouvoir optimiser θ **on doit évaluer** $J(\theta)$;

⚠ **Problème :** Les trajectoires τ possibles sont très nombreuses et en haute dimension, donc **on ne peut pas calculer τ pour toute les éventualités** et encore moins de façon exacte ;

Solution : On peut approximer τ en échantillonnant à partir de sa distribution, c.-à-d. en exécutant des trajectoires dans un environnement par rapport à la politique courante ; (C'est un estimer de Monte-Carlo).

Estimé de Monte-Carlo

$$
\begin{aligned}
J(\theta) &= \mathbb{E}_{\tau\sim\pi_\theta(\tau)}\left[\sum_{t=1}^{T} r(\mathbf{s}_t, \mathbf{a}_t)\right] \\
&\approx \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T} r_t^{(i)}
\end{aligned}
$$

avec N le nombre d'échantillons de trajectoires τ

REINFORCE [*Monte-Carlo Policy Gradient*]

Enjeux 2 : Avant de pouvoir optimiser θ **on doit évaluer $J(\theta)$** ;

⚠ **Problème :** Les trajectoires τ possibles sont très nombreuses et en haute dimension, donc **on ne peut pas calculer τ pour toute les éventualités** et encore moins de façon exacte ;

Solution : On peut approximer τ en échantillonnant à partir de sa distribution, c.-à-d. en exécutant des trajectoires dans un environnement par rapport à la politique courante ; (C'est un estimer de Monte-Carlo).

Estimé de Monte-Carlo

$$J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta(\tau)}\left[\sum_{t=1}^{T} r(\mathbf{s}_t, \mathbf{a}_t)\right]$$

$$\approx \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T} r_t^{(i)}$$

avec N le nombre d'échantillons de trajectoires τ

REINFORCE [*Monte-Carlo Policy Gradient*]

Enjeux 2 : Avant de pouvoir optimiser θ **on doit évaluer** $J(\theta)$;

⚠ **Problème :** Les trajectoires τ possibles sont très nombreuses et en haute dimension, donc **on ne peut pas calculer τ pour toute les éventualités** et encore moins de façon exacte ;

Solution : On peut approximer τ en échantillonnant à partir de sa distribution, c.-à-d. en exécutant des trajectoires dans un environnement par rapport à la politique courante ; (C'est un estimer de Monte-Carlo).

Estimé de Monte-Carlo

$$J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta(\tau)} \left[\sum_{t=1}^{\mathrm{T}} r(\mathbf{s}_t, \mathbf{a}_t) \right]$$

$$\approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=1}^{\mathrm{T}} r_t^{(i)}$$

avec N le nombre d'échantillons de trajectoires τ

REINFORCE [*Monte-Carlo Policy Gradient*]

En combinant le résultat de la **différentiation directe de la politique** et celui de l'**estimer de Monte-Carlo** on obtient :

$$\begin{aligned}
\nabla_\theta J(\theta) &= \mathbb{E}_{\tau\sim\pi_\theta(\tau)}\left[\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(\mathbf{a}_t|\mathbf{s}_t)\, G(\tau)\right] \\
&= \ldots \\
&\approx \frac{1}{N}\sum_{i=1}^{N} \nabla_\theta \log \pi_\theta\left(\tau^{(i)}\right) G\left(\tau^{(i)}\right)
\end{aligned}$$

⟨ Voir l'annexe *Complément théorique* pour la dérivation complète ⟩

REINFORCE [*Monte-Carlo Policy Gradient*]

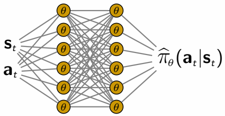
En combinant le résultat de la **différentiation directe de la politique** et celui de l'**estimer de Monte-Carlo** on obtient :

$$\begin{aligned}\nabla_\theta J(\theta) &= \mathbb{E}_{\tau\sim\pi_\theta(\tau)}\left[\sum_{t=1}^{T}\nabla_\theta \log \pi_\theta(\mathbf{a}_t|\mathbf{s}_t)\, G(\tau)\right] \\ &= \dots \\ &\approx \frac{1}{N}\sum_{i=1}^{N}\nabla_\theta \log \pi_\theta\left(\tau^{(i)}\right)\, G\left(\tau^{(i)}\right)\end{aligned}$$

⟨ Voir l'annexe *Complément théorique* pour la dérivation complète ⟩

- G est utilisé comme **mesure de qualité** de τ
- G est un **poid** attribué à chaque échantillon

REINFORCE *[Monte-Carlo Policy Gradient]*



Caractéristiques :

- Apprentissage **HORS-ligne** et **ON-policy** ;
- Optimise une politique stochastique $\pi_\theta(\mathbf{a}_t|\mathbf{s}_t) \; = \; \mathbb{P}[A_t = \mathbf{s}, S_t = \mathbf{a} \,|\, \theta]$

Idées clés :

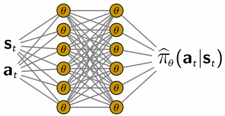
1. **Représenter** $\pi(\mathbf{a}|\mathbf{s}) \approx \widehat{\pi}_\theta(\mathbf{a}|\mathbf{s})$ un réseau neuronaux de vecteur de paramètres θ (le « **Policy-network** ») ;
2. **Définir** la fonction objectif : $J(\theta) \; = \; \mathbb{E}_{\tau\sim\pi_\theta}[\,G(\tau)\,]$
3. Résoudre l'objectif par **différentiation direct de la politique**
4. et **approximer τ par un estimer de Monte-Carlo** (c.a.d. e-à-cuter la politique dans l'environnement)

$$\nabla_{\theta_k} J(\theta_k) \;\; \approx \;\; \frac{1}{N}\sum_{i=1}^{N} \nabla_{\theta_k} \log \widehat{\pi}_{\theta_k}\left(\tau^{(i)}\right) \, G\left(\tau^{(i)}\right)$$

5. **Optimiser** l'objectif par *ascension du gradient* :

$$\theta_{k+1} \;\; \longleftarrow \;\; \theta_k + \alpha \nabla_\theta J(\theta_k)$$

REINFORCE [*Monte-Carlo Policy Gradient*]



Caractéristiques :

- Apprentissage **HORS-ligne** et **ON-policy** ;
- Optimise une politique stochastique $\pi_\theta(\mathbf{a}_t|\mathbf{s}_t) \;=\; \mathbb{P}\,[A_t = \mathbf{s}, S_t = \mathbf{a} \,|\, \theta\,]$

Idées clés :

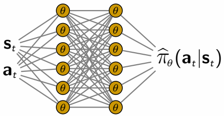
1. **Représenter** $\pi(\mathbf{a}|\mathbf{s}) \;\approx\; \widehat{\pi}_\theta(\mathbf{a}|\mathbf{s})$ un réseau neuronaux de vecteur de paramètres θ (le « **Policy-network** ») ;
2. **Définir** la fonction objectif : $J(\theta) \;=\; \mathbb{E}_{\tau\sim\pi_\theta}\,[\,G(\tau)\,]$
3. Résoudre l'objectif par **différentiation direct de la politique**
4. et **approximer τ par un estimer de Monte-Carlo** (c.a.d. e-à-cuter la politique dans l'environnement)

$$\nabla_{\theta_k} J(\theta_k) \;\approx\; \frac{1}{N}\sum_{i=1}^{N} \nabla_{\theta_k} \log \widehat{\pi}_{\theta_k}\left(\tau^{(i)}\right)\, G\left(\tau^{(i)}\right)$$

5. **Optimiser** l'objectif par *ascension du gradient* :

$$\theta_{k+1} \;\longleftarrow\; \theta_k + \alpha\,\nabla_\theta J(\theta_k)$$

REINFORCE [*Monte-Carlo Policy Gradient*]



Caractéristiques :

- Apprentissage **HORS-ligne** et **ON-policy** ;
- Optimise une politique stochastique $\pi_\theta(\mathbf{a}_t|\mathbf{s}_t) = \mathbb{P}[A_t = \mathbf{s}, S_t = \mathbf{a} \,|\, \theta]$

Idées clés :

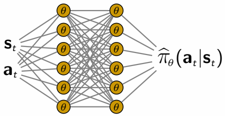
1. **Représenter** $\pi(\mathbf{a}|\mathbf{s}) \approx \widehat{\pi}_\theta(\mathbf{a}|\mathbf{s})$ un réseau neuronaux de vecteur de paramètres θ (le « **Policy-network** ») ;
2. **Définir** la fonction objectif : $J(\theta) = \mathbb{E}_{\tau\sim\pi_\theta}[\,G(\tau)\,]$
3. Résoudre l'objectif par **différentiation direct de la politique**
4. et **approximer τ par un estimer de Monte-Carlo** (c.a.d. e-à-cuter la politique dans l'environnement)

$$\nabla_{\theta_k} J(\theta_k) \approx \frac{1}{N}\sum_{i=1}^{N} \nabla_{\theta_k} \log \widehat{\pi}_{\theta_k}\left(\tau^{(i)}\right) G\left(\tau^{(i)}\right)$$

5. **Optimiser** l'objectif par *ascension du gradient* :

$$\theta_{k+1} \longleftarrow \theta_k + \alpha \nabla_\theta J(\theta_k)$$

REINFORCE [*Monte-Carlo Policy Gradient*]



Caractéristiques :

- Apprentissage **HORS-ligne** et **ON-policy** ;
- Optimise une politique stochastique $\pi_\theta(\mathbf{a}_t|\mathbf{s}_t) = \mathbb{P}[A_t = \mathbf{s}, S_t = \mathbf{a} \mid \theta]$

Idées clés :

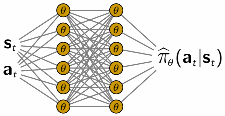
1. **Représenter** $\pi(\mathbf{a}|\mathbf{s}) \approx \widehat{\pi}_\theta(\mathbf{a}|\mathbf{s})$ un réseau neuronaux de vecteur de paramètres θ (le « **Policy-network** ») ;
2. **Définir** la fonction objectif : $J(\theta) = \mathbb{E}_{\tau\sim\pi_\theta}[\,G(\tau)\,]$
3. Résoudre l'objectif par **différentiation direct de la politique**
4. et **approximer τ par un estimer de Monte-Carlo** (c.a.d. e-à-cuter la politique dans l'environnement)

$$\nabla_{\theta_k} J(\theta_k) \approx \frac{1}{N}\sum_{i=1}^{N} \nabla_{\theta_k} \log \widehat{\pi}_{\theta_k}(\tau^{(i)})\; G(\tau^{(i)})$$

5. **Optimiser** l'objectif par *ascension du gradient* :

$$\theta_{k+1} \longleftarrow \theta_k + \alpha \nabla_\theta J(\theta_k)$$

REINFORCE *[Monte-Carlo Policy Gradient]*



Caractéristiques :

- Apprentissage **HORS-ligne** et **ON-policy** ;
- Optimise une politique stochastique $\pi_\theta(\mathbf{a}_t|\mathbf{s}_t) \;=\; \mathbb{P}\,[A_t = \mathbf{s}, S_t = \mathbf{a} \,|\, \theta\,]$

Idées clés :

1. **Représenter** $\pi(\mathbf{a}|\mathbf{s}) \approx \widehat{\pi}_\theta(\mathbf{a}|\mathbf{s})$ un réseau neuronaux de vecteur de paramètres θ (le « **Policy-network** ») ;
2. **Définir** la fonction objectif : $J(\theta) \;=\; \mathbb{E}_{\tau\sim\pi_\theta}\,[\,G(\tau)\,]$
3. Résoudre l'objectif par **différentiation direct de la politique**
4. et **approximer τ par un estimer de Monte-Carlo** (c.a.d. e-à-cuter la politique dans l'environnement)

$$\nabla_{\theta_k} J(\theta_k) \;\approx\; \tfrac{1}{N}\textstyle\sum_{i=1}^{N} \nabla_{\theta_k} \log \widehat{\pi}_{\theta_k}\left(\tau^{(i)}\right)\, G\left(\tau^{(i)}\right)$$

5. **Optimiser** l'objectif par *ascension du gradient* :

$$\theta_{k+1} \;\longleftarrow\; \theta_k + \alpha\,\nabla_\theta J(\theta_k)$$

Algorithme | REINFORCE

Soit le « policy network » $\widehat{\pi}_\theta(\mathbf{a}|\mathbf{s})$ (une politique différentiable), une « batch » de trajectoire $\mathcal{D}$, le « learning rate » $\alpha \in]0,1]$ et le « global step » $k \leftarrow 0$

Répéter *pour chaque « global step »* $k = 1, 2, 3, \ldots$ ***jusqu'à convergence*** *vers* $\approx \pi^\star$ **:**

Répéter *pour chaque* ***échantillon*** $i = 1, 2, 3, \ldots, |\mathcal{D}|$ **:**

- **Générer une trajectoire** en suivant π_{θ_k} jusqu'à terminaison

$$\tau^{(i)} = \mathbf{s}_0, \mathbf{a}_0, r_1, \mathbf{s}_1, \mathbf{a}_1, r_2, \ldots, \mathbf{s}_t, \mathbf{a}_t, r_{t+1}, \ldots, \mathbf{s}_{T-1}, \mathbf{a}_{T-1}, r_T$$

- $\mathcal{D} \longleftarrow \mathcal{D} \cup \{\tau^{(i)}\}$

$$\nabla_{\theta_k} J(\theta_k) \approx \frac{1}{|\mathcal{D}|} \sum_{\tau^{(i)} \in \mathcal{D}} \nabla_{\theta_k} \log \hat{\pi}_{\theta_k}(\tau^{(i)})\, G(\tau^{(i)})$$

$$\theta_{k+1} \longleftarrow \theta_k + \alpha \nabla_{\theta_k} J(\theta_k)$$

REINFORCE [*Monte-Carlo Policy Gradient*]

Pro :

- Fonctionne sur les **espaces d'action discret et continue** ;
- **N'utilise pas la propriété de Markov**
  $\Longrightarrow$ Peu être utilisé sur des MDP partiellement observé sans modification [POMDP] ;
- L'algorithme est agnostique à la dynamique du système
  $\Longrightarrow$ Apprentissage **sans modèle** ;
- Non biaisé ;

Con :

- Algorithme **HORS-ligne + ON-policy** $\Longrightarrow$ l'efficience de l'algorithme **dépend de la capacité de l'environnement à générer rapidement des trajectoires complètes**.
  (Plus la génération des trajectoires est longue, plus l'entrainement devient inéfficace)
- **Beaucoup de variances** $\Longrightarrow$ mauvaise performance ⚠
  ★ Impératif d'utiliser une **technique de réduction de variance** pour mitiger le problème :
    - « reward-to-go »
    - « weighted expected reward (Optimal baseline) »
    - ...

REINFORCE [*Monte-Carlo Policy Gradient*]

Pro :

- Fonctionne sur les **espaces d'action discret et continue** ;
- **N'utilise pas la propriété de Markov**
  $\Longrightarrow$ Peu être utilisé sur des MDP partiellement observé sans modification [POMDP] ;
- L'algorithme est agnostique à la dynamique du système
  $\Longrightarrow$ Apprentissage **sans modèle** ;
- Non biaisé ;

Con :

- Algorithme **HORS-ligne + ON-policy** $\Longrightarrow$ l'efficience de l'algorithme **dépend de la capacité de l'environnement à générer rapidement des trajectoires complètes**.
  (Plus la génération des trajectoires est longue, plus l'entrainement devient inéfficace)
- **Beaucoup de variances** $\Longrightarrow$ mauvaise performance
  - ★ Impératif d'utiliser une **technique de réduction de variance** pour mitiger le problème :
    - « reward-to-go »
    - « weighted expected reward (Optimal baseline) »
    - ...

REINFORCE [*Monte-Carlo Policy Gradient*]

Pro :

- Fonctionne sur les **espaces d'action discret et continue** ;
- **N'utilise pas la propriété de Markov**
  $\Longrightarrow$ Peu être utilisé sur des MDP partiellement observé sans modification [POMDP] ;
- L'algorithme est agnostique à la dynamique du système
  $\Longrightarrow$ Apprentissage **sans modèle** ;
- Non biaisé ;

Con :

- Algorithme **HORS-ligne + ON-policy** $\Longrightarrow$ l'efficience de l'algorithme **dépend de la capacité de l'environnement à générer rapidement des trajectoires complètes**.
  (Plus la génération des trajectoires est longue, plus l'entrainement devient inéfficace)
- **Beaucoup de variances** $\Longrightarrow$ mauvaise performance ⚠

★ Impératif d'utiliser une **technique de réduction de variance** pour mitiger le problème :

- « reward-to-go »
- « weighted expected reward (Optimal baseline) »
- ...

REINFORCE *[Monte-Carlo Policy Gradient]*

Pro :

- Fonctionne sur les **espaces d'action discret et continue** ;
- **N'utilise pas la propriété de Markov**
  $\Longrightarrow$ Peu être utilisé sur des MDP partiellement observé sans modification [POMDP] ;
- L'algorithme est agnostique à la dynamique du système
  $\Longrightarrow$ Apprentissage **sans modèle** ;
- Non biaisé ;

Con :

- Algorithme **HORS-ligne + ON-policy** $\Longrightarrow$ l'efficience de l'algorithme **dépend de la capacité de l'environnement à générer rapidement des trajectoires complètes**.
  (Plus la génération des trajectoires est longue, plus l'entrainement devient inéfficace)
- **Beaucoup de variances** $\Longrightarrow$ mauvaise performance ⚠
  - ★ Impératif d'utiliser une **technique de réduction de variance** pour mitiger le problème :
    - « reward-to-go »
    - « weighted expected reward (Optimal baseline) »
    - ...

REINFORCE [*Monte-Carlo Policy Gradient*]

Pro :

- Fonctionne sur les **espaces d'action discret et continue** ;
- **N'utilise pas la propriété de Markov**
  $\Longrightarrow$ Peu être utilisé sur des MDP partiellement observé sans modification [POMDP] ;
- L'algorithme est agnostique à la dynamique du système
  $\Longrightarrow$ Apprentissage **sans modèle** ;
- Non biaisé ;

Con :

- Algorithme **HORS-ligne + ON-policy** $\Longrightarrow$ l'efficience de l'algorithme **dépend de la capacité de l'environnement à générer rapidement des trajectoires complètes**.
  (Plus la génération des trajectoires est longue, plus l'entrainement devient inéfficace)
- **Beaucoup de variances** $\Longrightarrow$ mauvaise performance ⚠
  ★ Impératif d'utiliser une **technique de réduction de variance** pour mitiger le problème :
  - **« reward-to-go »** $\longleftarrow$ **c'est ce qu'on va voir maintenant**
  - « weighted expected reward (Optimal baseline) »
  - ...

REINFORCE avec *reward-to-go*

REINFORCE avec *reward-to-go*

Problème : REINFORCE soufre d'avoir **trop de variance** ;

Solution : Utiliser le **principe de causalité** $\Longrightarrow$ « le futur n'affecte pas le passé » [1] ;

$$\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N} \nabla_\theta \log \pi_\theta\left(\tau^{(i)}\right) G\left(\tau^{(i)}\right)$$

$$= \dots$$

⟨ Voir l'annexe *Complément théorique* pour les détailes du calcule ⟩

$$= \frac{1}{N}\sum_{i=1}^{N}\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta\left(\mathbf{a}_t^{(i)} | \mathbf{s}_t^{(i)}\right) \widehat{Q}_t\left(\tau^{(i)}\right)$$

Pourquoi ça marche ?

Enlever les récompenses passées **rend la somme des récompenses plus petite**

$\Longrightarrow$ la variance $\mathbb{V} = \mathbb{E}\left[(X - \mathbb{E}[X])^2\right]$ devient également plus petite

Le « reward-to-go »

L'**estimer de Monte-Carlo d'un retour $G_t(\tau)$** est appeler intuitivement le « **reward-to-go** » $\widehat{Q}_t\left(\tau^{(i)}\right)$

$$G_t(\tau) = \sum_{t'=t}^{T} r\left(\mathbf{s}_{t'}, \mathbf{a}_{t'}\right) \approx \sum_{t'=t}^{T} r_{t'}^{(i)} = \widehat{Q}_t\left(\tau^{(i)}\right)$$

1. Voir *CS 294–112 Deep Reinforcement Learning* : *lecture 5 - Policy Gradient* (à partir de 35:09) donné par professeur Sergey Levine à l'université Berkeley en Californie ;

REINFORCE avec *reward-to-go*

⚠ **Problème :** REINFORCE soufre d'avoir **trop de variance** ;

Solution : Utiliser le **principe de causalité** $\Longrightarrow$ « le futur n'affecte pas le passé » [1] ;

$$
\begin{aligned}
\nabla_\theta J(\theta) &\approx \frac{1}{N}\sum_{i=1}^{N} \nabla_\theta \log \pi_\theta\left(\tau^{(i)}\right) G\left(\tau^{(i)}\right) \\
&= \dots \\
&= \frac{1}{N}\sum_{i=1}^{N} \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta\left(\mathbf{a}_t^{(i)} | \mathbf{s}_t^{(i)}\right) \widehat{Q}_t\left(\tau^{(i)}\right)
\end{aligned}
$$

⟨ Voir l'annexe *Complément théorique* pour les détailes du calcule ⟩

Pourquoi ça marche ? $\Longrightarrow$ Enlever les récompenses passées **rend la somme des récompenses plus petite**

la variance $\mathbb{V} = \mathbb{E}\left[(X - \mathbb{E}[X])^2\right]$ devient également plus petite

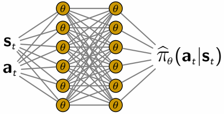
Le « reward-to-go »

L'**estimer de Monte-Carlo d'un retour** $G_t(\tau)$ est appeler intuitivement le « **reward-to-go** » $\widehat{Q}_t\left(\tau^{(i)}\right)$

$$
G_t(\tau) = \sum_{t'=t}^{T} r\left(\mathbf{s}_{t'}, \mathbf{a}_{t'}\right) \approx \sum_{t'=t}^{T} r_{t'}^{(i)} = \widehat{Q}_t\left(\tau^{(i)}\right)
$$

1. Voir *CS 294–112 Deep Reinforcement Learning* : *lecture 5 - Policy Gradient* (à partir de 35:09) donné par professeur Sergey Levine à l'université Berkeley en Californie ;

REINFORCE avec reward-to-go [*Monte-Carlo Policy Gradient*]



Idées clés :

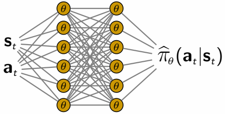
1. Extension de « REINFORCE »
2. **Technique de réduction de la variance**
3. **Remplacer** le retour $G\left(\tau^{(i)}\right)$ par le « **reward-to-go** » $\widehat{Q}_t^\pi\left(\tau^{(i)}\right)$:

$$\widehat{Q}_t^\pi\left(\tau^{(i)}\right) = \sum_{t'=t}^{T} r_{t'}^{(i)}$$

$$\nabla_{\theta_k} J(\theta_k) \approx \frac{1}{|\mathcal{D}|} \sum_{i=1}^{|\mathcal{D}|} \sum_{t=1}^{T} \nabla_{\theta_k} \log \pi_{\theta_k}\left(\mathbf{a}_t^{(i)} \middle| \mathbf{s}_t^{(i)}\right) \widehat{Q}_t\left(\tau^{(i)}\right)$$

Pro : **Simple à implémenter** ;

REINFORCE avec reward-to-go [*Monte-Carlo Policy Gradient*]



Idées clés :

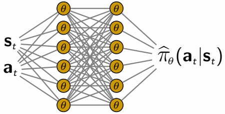
1. Extension de « REINFORCE »
2. **Technique de réduction de la variance**
3. **Remplacer** le retour $G\left(\tau^{(i)}\right)$ par le « **reward-to-go** » $\widehat{Q}_t^\pi\left(\tau^{(i)}\right)$:

$$\widehat{Q}_t^\pi\left(\tau^{(i)}\right) = \sum_{t'=t}^{\mathsf{T}} r_{t'}^{(i)}$$

$$\nabla_{\theta_k} J(\theta_k) \approx \frac{1}{|\mathcal{D}|} \sum_{i=1}^{|\mathcal{D}|} \sum_{t=1}^{\mathsf{T}} \nabla_{\theta_k} \log \pi_{\theta_k}\left(\mathbf{a}_t^{(i)} \middle| \mathbf{s}_t^{(i)}\right) \widehat{Q}_t\left(\tau^{(i)}\right)$$

Pro : Simple à implémenter ;

REINFORCE avec reward-to-go [*Monte-Carlo Policy Gradient*]



Idées clés :

1. Extension de « REINFORCE »
2. **Technique de réduction de la variance**
3. **Remplacer** le retour $G\left(\tau^{(i)}\right)$ par le « **reward-to-go** » $\widehat{Q}_t^{\pi}\left(\tau^{(i)}\right)$:

$$\widehat{Q}_t^{\pi}\left(\tau^{(i)}\right) = \sum_{t'=t}^{T} r_{t'}^{(i)}$$

$$\nabla_{\theta_k} J(\theta_k) \approx \frac{1}{|\mathcal{D}|} \sum_{i=1}^{|\mathcal{D}|} \sum_{t=1}^{T} \nabla_{\theta_k} \log \pi_{\theta_k}\left(\mathbf{a}_t^{(i)} \middle| \mathbf{s}_t^{(i)}\right) \widehat{Q}_t\left(\tau^{(i)}\right)$$

Pro : **Simple à implémenter** ;

Algorithme | REINFORCE avec reward-to-go

Soit le « policy network » $\widehat{\pi}_\theta(\mathbf{a}|\mathbf{s})$ (une politique différentiable), une « batch » de trajectoire $\mathcal{D}$, le « learning rate » $\alpha \in]0,1]$ et le « global step » $k \leftarrow 0$

Répéter *pour chaque « global step »* $k = 1, 2, 3, \ldots$ ***jusqu'à convergence*** *vers* $\approx \pi^\star$ **:**

Répéter *pour chaque* ***échantillon*** $i = 1, 2, 3, \ldots, |\mathcal{D}|$ **:**

- **Générer une trajectoire** en suivant π_{θ_k} jusqu'à terminaison

$$\tau^{(i)} = \mathbf{s}_0^{(i)}, \mathbf{a}_0^{(i)}, r_1^{(i)}, \mathbf{s}_1^{(i)}, \mathbf{a}_1^{(i)}, r_2^{(i)}, \ldots, \mathbf{s}_t^{(i)}, \mathbf{a}_t^{(i)}, r_{t+1}^{(i)}, \ldots, \mathbf{s}_{T-1}^{(i)}, \mathbf{a}_{T-1}^{(i)}, r_T^{(i)}$$

- $\mathcal{D} \longleftarrow \mathcal{D} \cup \{\tau^{(i)}\}$

$$\nabla_{\theta_k} J(\theta_k) \approx \frac{1}{|\mathcal{D}|} \sum_{i=1}^{|\mathcal{D}|} \sum_{t=1}^{T} \nabla_{\theta_k} \log \pi_{\theta_k}\left(\mathbf{a}_t^{(i)} \middle| \mathbf{s}_t^{(i)}\right) \left(\sum_{t'=t}^{T} r_{t'}^{(i)}\right)$$

$$\theta_{k+1} \longleftarrow \theta_k + \alpha \nabla_{\theta_k} J(\theta_k)$$

Advantage Actor-Critic

Élargir la portée de *REINFORCE* avec « *reward-to-go* »

$$\nabla_{\theta_k} J(\theta_k) \approx \frac{1}{|\mathcal{D}|} \sum_{i=1}^{|\mathcal{D}|} \sum_{t=1}^{T} \nabla_{\theta_k} \log \pi_{\theta_k}(\mathbf{a}_t^{(i)} \,|\, \mathbf{s}_t^{(i)}) \; \widehat{Q}_t(\tau^{(i)})$$

Problèmes :

1. $\widehat{Q}_t(\tau^{(i)})$ est **l'estimer d'une espérance basée sur un seul échantillon**
   $\Longrightarrow$ haute variance ;
2. $\widehat{Q}_t(\tau^{(i)})$ **tien compte d'un seul futur possible**, celui qui a été échantillonné dans la trajectoire $\tau^{(i)}$.
   $\Longrightarrow$ ces un estimateur avec une vision très étroite de la réalité

Solution : Remplacer l'estimer à seul échantillon de trajectoire $\widehat{Q}_t(\tau^{(i)})$ par l'***Avantage*** :

$$A^\pi(\mathbf{s}_t, \mathbf{a}_t) = Q^\pi(\mathbf{s}_t, \mathbf{a}_t) - V^\pi(\mathbf{s}_t)$$

$\langle$ **Compromis** : estimer basée sur un seul échantillon de récompense $\rangle$

$$\approx r_t^{(i)} + V^\pi(\mathbf{s}_{t+1}) - V^\pi(\mathbf{s}_t)$$

L'*Avantage* $A^\pi(\mathbf{s}_t, \mathbf{a}_t)$ répond à la question suivante :

*« **à quel point meilleure** est l'action $\mathbf{a}_t$ dans l'état $\mathbf{s}_t$ **par rapport à la moyenne** de tout les actions $\mathbf{a}'_t$ possibles dans cet état state $\mathbf{s}_t$? »*

Élargir la portée de *REINFORCE* avec « *reward-to-go* »

$$\nabla_{\theta_k} J(\theta_k) \approx \frac{1}{|\mathcal{D}|} \sum_{i=1}^{|\mathcal{D}|} \sum_{t=1}^{T} \nabla_{\theta_k} \log \pi_{\theta_k}(\mathbf{a}_t^{(i)} \,|\, \mathbf{s}_t^{(i)}) \; \widehat{Q}_t(\tau^{(i)})$$

Problèmes :

1. $\widehat{Q}_t\,(\tau^{(i)})$ est **l'estimer d'une espérance basée sur un seul échantillon**
$\Longrightarrow$ haute variance ;
2. $\widehat{Q}_t\,(\tau^{(i)})$ **tien compte d'un seul futur possible**, celui qui a été échantillonné dans la trajectoire $\tau^{(i)}$.
$\Longrightarrow$ ces un estimateur avec une vision très étroite de la réalité

Solution : Remplacer l'estimer à seul échantillon de trajectoire $\widehat{Q}_t\,(\tau^{(i)})$ par l'***Avantage*** :

$$A^\pi(\mathbf{s}_t, \mathbf{a}_t) = Q^\pi(\mathbf{s}_t, \mathbf{a}_t) - V^\pi(\mathbf{s}_t)$$

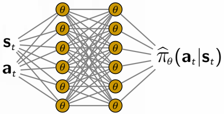
⟨ **Compromis** : estimer basée sur un seul échantillon de récompense ⟩

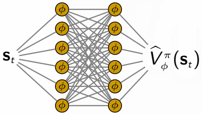
$$\approx r_t^{(i)} + V^\pi(\mathbf{s}_{t+1}) - V^\pi(\mathbf{s}_t)$$

L'*Avantage* $A^\pi(\mathbf{s}_t, \mathbf{a}_t)$ répond à la question suivante :

*« **à quel point meilleure** est l'action $\mathbf{a}_t$ dans l'état $\mathbf{s}_t$ **par rapport à la moyenne** de tout les actions $\mathbf{a}'_t$ possibles dans cet état state $\mathbf{s}_t$? »*

Advantage Actor-Critic





Caractéristiques : Apprentissage **ON-ligne**[1] et **ON-policy** ;

Idées clés :

1. **Apprendre deux réseaux neuronaux** :
   - $\widehat{\pi}_\theta(\mathbf{a}|\mathbf{s}) \longrightarrow$ **l'acteur** : celui responsable de prendre les **décisions** dans l'environnement ;
   - $\widehat{V}_\phi(\mathbf{a}|\mathbf{s}) \longrightarrow$ **le critique** : celui responsable d'**évaluer** si $\widehat{\pi}_\theta$ fait du bon travail ;
2. **Apprendre $\widehat{V}^\pi_\phi(\mathbf{s}_t)$** par MSE avec le « bootstrap target » :

$$\mathbf{y}_k = r^{(i)}_{t+1} + \widehat{V}^\pi_\phi(\mathbf{s}^{(i)}_{t+1})$$

3. **Calculer l'Avantage**

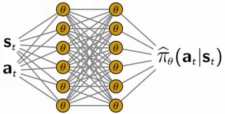
$$\widehat{A}^\pi\left(\mathbf{s}_t, r^{(i)}_t, \mathbf{s}^{(i)}_{t+1}\right) = r^{(i)}_{t+1} + \widehat{V}^\pi_\phi(\mathbf{s}^{(i)}_{t+1}) - \widehat{V}^\pi_\phi(\mathbf{s}_t)$$

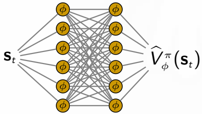
4. **Résoudre l'objectif**

$$\nabla_{\theta_k} J(\theta_k) \approx \sum_{t=1}^{T} \nabla_{\theta_k} \log \pi_{\theta_k}(\mathbf{a}_t|\,\mathbf{s}_t)\, \widehat{A}^\pi\left(\mathbf{s}_t, r^{(i)}_t, \mathbf{s}^{(i)}_{t+1}\right)$$

1. Il existe aussi une version **OFF-ligne**

Advantage Actor-Critic





Caractéristiques : Apprentissage **ON-ligne**[1] et **ON-policy** ;

Idées clés :

1. **Apprendre deux réseaux neuronaux** :

   $\widehat{\pi}_\theta(\mathbf{a}|\mathbf{s}) \longrightarrow$ **l'acteur** : celui responsable de prendre les **décisions** dans l'environnement ;

   $\widehat{V}_\phi(\mathbf{a}|\mathbf{s}) \longrightarrow$ **le critique** : celui responsable d'**évaluer** si $\widehat{\pi}_\theta$ fait du bon travail ;

2. **Apprendre** $\widehat{V}^\pi_\phi(\mathbf{s}_t)$ par MSE avec le « bootstrap target » :

$$\mathbf{y}_k = r^{(i)}_{t+1} + \widehat{V}^\pi_\phi(\mathbf{s}^{(i)}_{t+1})$$

3. **Calculer l'Avantage**

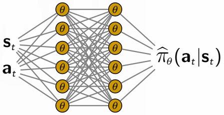
$$\widehat{A}^\pi\left(\mathbf{s}_t, r^{(i)}_t, \mathbf{s}^{(i)}_{t+1}\right) = r^{(i)}_{t+1} + \widehat{V}^\pi_\phi(\mathbf{s}^{(i)}_{t+1}) - \widehat{V}^\pi_\phi(\mathbf{s}_t)$$

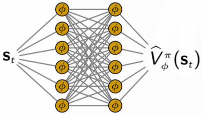
4. **Résoudre l'objectif**

$$\nabla_{\theta_k} J(\theta_k) \approx \sum_{t=1}^{T} \nabla_{\theta_k} \log \pi_{\theta_k}(\mathbf{a}_t|\,\mathbf{s}_t)\, \widehat{A}^\pi\left(\mathbf{s}_t, r^{(i)}_t, \mathbf{s}^{(i)}_{t+1}\right)$$

1. Il existe aussi une version **OFF-ligne**

Advantage Actor-Critic





Caractéristiques : Apprentissage **ON-ligne** [1] et **ON-policy** ;

Idées clés :

1. **Apprendre deux réseaux neuronaux** :

   $\widehat{\pi}_\theta(\mathbf{a}|\mathbf{s}) \longrightarrow$ **l'acteur** : celui responsable de prendre les **décisions** dans l'environnement ;

   $\widehat{V}_\phi(\mathbf{a}|\mathbf{s}) \longrightarrow$ **le critique** : celui responsable d'**évaluer** si $\widehat{\pi}_\theta$ fait du bon travail ;

2. **Apprendre $\widehat{V}^\pi_\phi(\mathbf{s}_t)$** par MSE avec le « bootstrap target » :

$$\mathbf{y}_k = r^{(i)}_{t+1} + \widehat{V}^\pi_\phi(\mathbf{s}^{(i)}_{t+1})$$

3. **Calculer l'Avantage**

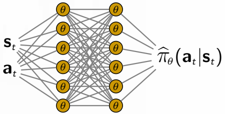
$$\widehat{A}^\pi\left(\mathbf{s}_t, r^{(i)}_t, \mathbf{s}^{(i)}_{t+1}\right) = r^{(i)}_{t+1} + \widehat{V}^\pi_\phi(\mathbf{s}^{(i)}_{t+1}) - \widehat{V}^\pi_\phi(\mathbf{s}_t)$$

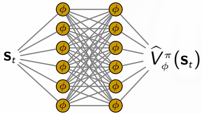
4. **Résoudre l'objectif**

$$\nabla_{\theta_k} J(\theta_k) \approx \sum_{t=1}^{T} \nabla_{\theta_k} \log \pi_{\theta_k}(\mathbf{a}_t|\,\mathbf{s}_t)\, \widehat{A}^\pi\left(\mathbf{s}_t, r^{(i)}_t, \mathbf{s}^{(i)}_{t+1}\right)$$

1. Il existe aussi une version **OFF-ligne**

Advantage Actor-Critic





Caractéristiques : Apprentissage **ON-ligne**[1] et **ON-policy** ;

Idées clés :

1. **Apprendre deux réseaux neuronaux** :

   $\widehat{\pi}_\theta(\mathbf{a}|\mathbf{s}) \longrightarrow$ **l'acteur** : celui responsable de prendre les **décisions** dans l'environnement ;

   $\widehat{V}_\phi(\mathbf{a}|\mathbf{s}) \longrightarrow$ **le critique** : celui responsable d'**évaluer** si $\widehat{\pi}_\theta$ fait du bon travail ;

2. **Apprendre** $\widehat{V}^{\pi}_{\phi}(\mathbf{s}_t)$ par MSE avec le « bootstrap target » :

$$\mathbf{y}_k = r^{(i)}_{t+1} + \widehat{V}^{\pi}_{\phi}(\mathbf{s}^{(i)}_{t+1})$$

3. **Calculer l'Avantage**

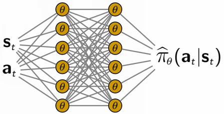
$$\widehat{A}^{\pi}\left(\mathbf{s}_t, r^{(i)}_t, \mathbf{s}^{(i)}_{t+1}\right) = r^{(i)}_{t+1} + \widehat{V}^{\pi}_{\phi}(\mathbf{s}^{(i)}_{t+1}) - \widehat{V}^{\pi}_{\phi}(\mathbf{s}_t)$$

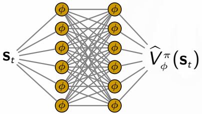
4. **Résoudre l'objectif**

$$\nabla_{\theta_k} J(\theta_k) \approx \sum_{t=1}^{T} \nabla_{\theta_k} \log \pi_{\theta_k}(\mathbf{a}_t|\mathbf{s}_t)\, \widehat{A}^{\pi}\left(\mathbf{s}_t, r^{(i)}_t, \mathbf{s}^{(i)}_{t+1}\right)$$

1. Il existe aussi une version **OFF-ligne**

Advantage Actor-Critic





Caractéristiques : Apprentissage **ON-ligne**[1] et **ON-policy** ;

Idées clés :

1. **Apprendre deux réseaux neuronaux** :

   $\widehat{\pi}_\theta(\mathbf{a}|\mathbf{s}) \longrightarrow$ **l'acteur** : celui responsable de prendre les **décisions** dans l'environnement ;

   $\widehat{V}_\phi(\mathbf{a}|\mathbf{s}) \longrightarrow$ **le critique** : celui responsable d'**évaluer** si $\widehat{\pi}_\theta$ fait du bon travail ;

2. **Apprendre $\widehat{V}^\pi_\phi(\mathbf{s}_t)$** par MSE avec le « bootstrap target » :

$$\mathbf{y}_k \;=\; r^{(i)}_{t+1} + \widehat{V}^\pi_\phi(\mathbf{s}^{(i)}_{t+1})$$

3. **Calculer l'Avantage**

$$\widehat{A}^\pi\left(\mathbf{s}_t, r^{(i)}_t, \mathbf{s}^{(i)}_{t+1}\right) \;=\; r^{(i)}_{t+1} + \widehat{V}^\pi_\phi(\mathbf{s}^{(i)}_{t+1}) - \widehat{V}^\pi_\phi(\mathbf{s}_t)$$

4. **Résoudre l'objectif**

$$\nabla_{\theta_k} J(\theta_k) \;\approx \sum_{t=1}^{\mathrm{T}} \nabla_{\theta_k} \log \pi_{\theta_k}(\mathbf{a}_t|\,\mathbf{s}_t)\, \widehat{A}^\pi\left(\mathbf{s}_t, r^{(i)}_t, \mathbf{s}^{(i)}_{t+1}\right)$$

---

1. Il existe aussi une version **OFF-ligne**

Algorithme | *Advantage Actor-Critic*

Soit « **l'acteur** » $\widehat{\pi}_\theta(\mathbf{a}|\mathbf{s})$ (une politique différentiable), le « **critique** » $\widehat{v}^{\pi}_{\phi}(\mathbf{s})$, une « batch » de trajectoire $\mathcal{D}$, le « learning rate » $\alpha \in]0,1]$ et le « global step » $k \leftarrow 0$

Répéter *pour chaque « global step »* $k = 1, 2, 3, \ldots$ ***jusqu'à convergence*** *vers* $\approx \pi^\star$ **:**

Répéter *pour chaque « timestep »* $t = 0, 1, 2, \ldots, t, t+1, \ldots, T$ *de la trajectoire* $\mathcal{T}$ **:**

$r^{(k)}_{t+1},\ \mathbf{s}^{(k)}_{t+1} \quad \longleftarrow \quad$ exécuter $\mathbf{a}_t \sim \mu(\cdot|\mathbf{s}_t)$ dans l'environnement et observer la réaction

mettre à jour $\widehat{V}^{\pi}_{\phi}(\mathbf{s}_t)$ par MSE avec le « bootstrap target » $\mathbf{y}_k = r^{(i)}_{t+1} + \widehat{V}^{\pi}_{\phi}(\mathbf{s}^{(i)}_{t+1})$

$$\nabla_{\theta_k} J(\theta_k) \approx \nabla_{\theta_k} \log \pi_{\theta_k}(\mathbf{a}_t|\,\mathbf{s}_t) \left(r^{(i)}_{t+1} + \widehat{V}^{\pi}_{\phi}(\mathbf{s}^{(i)}_{t+1}) - \widehat{V}^{\pi}_{\phi}(\mathbf{s}_t) \right)$$

$$\theta_{k+1} \longleftarrow \theta_k + \alpha \nabla_{\theta_k} J(\theta_k)$$

Pro : Moins de variance que REINFORCE avec « reward-to-go » ;

Con : N'est plus non-biaisé par rapport à REINFORCE ;

Ressource théorique

REINFORCE

- *Simple statistical gradient-following algorithms for connectionist reinforcement learning* ▸ Publication

  par Williams, Ronald J. (1992) [10]
- *Reinforcement Learning : An introduction*, chapitre 13 [6] ▸ Publication

  par Sutton & Barto
- *CS 294–112 Deep Reinforcement Learning : lecture 5 - Policy Gradient* ▸ Présentation

  donné par professeur Sergey Levine à l'université Berkeley en Californie

Actor-Critic

- *Reinforcement Learning : An introduction*, chapitre 13, section 5 [6] ▸ Publication

  par Sutton & Barto
- *CS 294–112 Deep Reinforcement Learning : lecture 6 - Actor-Critic Algorithms* ▸ Présentation

  donné par professeur Sergey Levine à l'université Berkeley en Californie

Pour aller plus loin

Pour aller plus loin

Complément théorique

REINFORCE *[Monte-Carlo Policy Gradient]*

Différentiation directe de la politique (dérivation complète)

$$\nabla_\theta J(\theta) = \nabla_\theta \left(\mathbb{E}_{\tau \sim \pi_\theta(\tau)} [\, G(\tau) \,] \right)$$

$\langle$ L'objectif de l'apprentissage par renforcement $\rangle$

$$= \int \nabla_\theta \, \pi_\theta(\tau)\, G(\tau)\, d\tau$$

$\left\langle \text{Identité : } \nabla_\theta \, \pi_\theta(\tau) = \frac{\pi_\theta(\tau)}{\pi_\theta(\tau)} \nabla_\theta \, \pi_\theta(\tau) = \pi_\theta(\tau)\, \nabla_\theta \, \log \pi_\theta(\tau) \right\rangle$

$$= \int \pi_\theta(\tau)\, \nabla_\theta \, \log \pi_\theta(\tau)\, G(\tau)\, d\tau$$

$$= \mathbb{E}_{\tau \sim \pi_\theta(\tau)} [\, \nabla_\theta \log \pi_\theta(\tau)\, G(\tau) \,]$$

$\left\langle \pi_\theta(\tau) = p(\mathbf{s}_1) \prod_{t=1}^{T} \pi_\theta(\mathbf{a}_t|\mathbf{s}_t)\, p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t) \right\rangle$

$$= \mathbb{E}_{\tau \sim \pi_\theta(\tau)} \left[\nabla_\theta \log \left(p(\mathbf{s}_1) \prod_{t=1}^{T} \pi_\theta(\mathbf{a}_t|\mathbf{s}_t)\, p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t) \right) G(\tau) \right]$$

$\left\langle \text{Identité : } \log(A \cdot B) = \log A + \log B \right\rangle$

$$= \mathbb{E}_{\tau \sim \pi_\theta(\tau)} \left[\nabla_\theta \left(\log p(\mathbf{s}_1) + \sum_{t=1}^{T} \log \pi_\theta(\mathbf{a}_t|\mathbf{s}_t) + \log p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t) \right) G(\tau) \right]$$

$\langle$ les termes qui ne dépandent pas de θ s'éliminent $\rangle$

$$= \mathbb{E}_{\tau \sim \pi_\theta(\tau)} \left[\nabla_\theta \left(\sum_{t=1}^{T} \log \pi_\theta(\mathbf{a}_t|\mathbf{s}_t) \right) G(\tau) \right]$$

$\left\langle \text{Propriété de distributivité : } \nabla(A+B) = \nabla A + \nabla B \right\rangle$

$$= \mathbb{E}_{\tau \sim \pi_\theta(\tau)} \left[\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(\mathbf{a}_t|\mathbf{s}_t)\, G(\tau) \right]$$

REINFORCE *[Monte-Carlo Policy Gradient]*

Combinaison du résultat de la **différentiation directe de la politique** et celui de l'**estimer de Monte-Carlo** (**dérivation complète**) :

$$
\begin{aligned}
\nabla_\theta J(\theta) &= \mathbb{E}_{\tau\sim\pi_\theta(\tau)}\left[\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(\mathbf{a}_t|\mathbf{s}_t)\, G(\tau)\right] \\
&= \mathbb{E}_{\tau\sim\pi_\theta(\tau)}\left[\left(\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta(\mathbf{a}_t|\mathbf{s}_t)\right)\left(\sum_{t=1}^{T} r(\mathbf{s}_t,\mathbf{a}_t)\right)\right]
\end{aligned}
$$

$\langle$ **estimer le gradient** par rapport à N échantillons $\rangle$

$$
\begin{aligned}
&\approx \frac{1}{N}\sum_{i=1}^{N}\left(\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta\left(\mathbf{a}_t^{(i)}|\mathbf{s}_t^{(i)}\right)\right)\left(\sum_{t=1}^{T} r_t^{(i)}\right) \\
&= \frac{1}{N}\sum_{i=1}^{N} \nabla_\theta \log \pi_\theta\left(\tau^{(i)}\right) G\left(\tau^{(i)}\right)
\end{aligned}
$$

REINFORCE avec *reward-to-go*

Application du **principe de causalité** à la structure temporelle (**détails du calcul**) :

$$\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N} \nabla_\theta \log \pi_\theta\left(\tau^{(i)}\right) G\left(\tau^{(i)}\right)$$

$$= \frac{1}{N}\sum_{i=1}^{N} \left(\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta\left(\mathbf{a}_t^{(i)}|\mathbf{s}_t^{(i)}\right)\right)\left(\sum_{t=1}^{T} r_t^{(i)}\right)$$

⟨ Distribuer les récompenses à l'intérieur de la somme des politiques ⟩

$$= \frac{1}{N}\sum_{i=1}^{N} \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta\left(\mathbf{a}_t^{(i)}|\mathbf{s}_t^{(i)}\right)\left(\sum_{t'=1}^{T} r_{t'}^{(i)}\right)$$

⟨ Appliquer le principe de causalité en changeant l'index de départ du retour $t'=1 \rightarrow t'=t$ ⟩

$$\geq \frac{1}{N}\sum_{i=1}^{N} \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta\left(\mathbf{a}_t^{(i)}|\mathbf{s}_t^{(i)}\right)\left(\sum_{t'=t}^{T} r_{t'}^{(i)}\right)$$

⟨ « **reward-to-go** » ⟩

$$= \frac{1}{N}\sum_{i=1}^{N} \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta\left(\mathbf{a}_t^{(i)}|\mathbf{s}_t^{(i)}\right) \widehat{Q}_t\left(\tau^{(i)}\right)$$

REINFORCE avec *reward-to-go*

Application du **principe de causalité** à la structure temporelle (**détails du calcul**) :

$$
\begin{aligned}
\nabla_\theta J(\theta) &\approx \frac{1}{N}\sum_{i=1}^{N} \nabla_\theta \log \pi_\theta\left(\tau^{(i)}\right) G\left(\tau^{(i)}\right) \\
&= \frac{1}{N}\sum_{i=1}^{N} \left(\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta\left(\mathbf{a}_t^{(i)} | \mathbf{s}_t^{(i)}\right) \right) \left(\sum_{t=1}^{T} r_t^{(i)} \right)
\end{aligned}
$$

⟨ Distribuer les récompenses à l'intérieur de la somme des politiques ⟩

$$
= \frac{1}{N}\sum_{i=1}^{N} \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta\left(\mathbf{a}_t^{(i)} | \mathbf{s}_t^{(i)}\right) \left(\sum_{t'=1}^{T} r_{t'}^{(i)} \right)
$$

⟨ Appliquer le principe de causalité en changeant l'index de départ du retour $t'=1 \rightarrow t'=t$ ⟩

$$
\geq \frac{1}{N}\sum_{i=1}^{N} \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta\left(\mathbf{a}_t^{(i)} | \mathbf{s}_t^{(i)}\right) \left(\sum_{t'=t}^{T} r_{t'}^{(i)} \right)
$$

⟨ « **reward-to-go** » ⟩

$$
= \frac{1}{N}\sum_{i=1}^{N} \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta\left(\mathbf{a}_t^{(i)} | \mathbf{s}_t^{(i)}\right) \widehat{Q}_t\left(\tau^{(i)}\right)
$$

REINFORCE avec *reward-to-go*

Application du **principe de causalité** à la structure temporelle (**détails du calcul**) :

$$\begin{aligned}
\nabla_\theta J(\theta) &\approx \frac{1}{N}\sum_{i=1}^{N} \nabla_\theta \log \pi_\theta\left(\tau^{(i)}\right) G\left(\tau^{(i)}\right) \\
&= \frac{1}{N}\sum_{i=1}^{N} \left(\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta\left(\mathbf{a}_t^{(i)} | \mathbf{s}_t^{(i)}\right) \right) \left(\sum_{t=1}^{T} r_t^{(i)} \right) \\
&\qquad \langle \text{Distribuer les récompenses à l'intérieur de la somme des politiques} \rangle \\
&= \frac{1}{N}\sum_{i=1}^{N} \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta\left(\mathbf{a}_t^{(i)} | \mathbf{s}_t^{(i)}\right) \left(\sum_{t'=1}^{T} r_{t'}^{(i)} \right) \\
&\qquad \langle \text{Appliquer le principe de causalité en changeant l'index de départ du retour } t'{=}1 \rightarrow t'{=}t \rangle \\
&\geq \frac{1}{N}\sum_{i=1}^{N} \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta\left(\mathbf{a}_t^{(i)} | \mathbf{s}_t^{(i)}\right) \left(\sum_{t'=t}^{T} r_{t'}^{(i)} \right) \\
&\qquad \langle \text{« reward-to-go »} \rangle \\
&= \frac{1}{N}\sum_{i=1}^{N} \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta\left(\mathbf{a}_t^{(i)} | \mathbf{s}_t^{(i)}\right) \widehat{Q}_t\left(\tau^{(i)}\right)
\end{aligned}$$

REINFORCE avec *reward-to-go*

Application du **principe de causalité** à la structure temporelle (**détails du calcul**) :

$$\nabla_\theta J(\theta) \approx \frac{1}{N}\sum_{i=1}^{N} \nabla_\theta \log \pi_\theta\left(\tau^{(i)}\right) G\left(\tau^{(i)}\right)$$

$$= \frac{1}{N}\sum_{i=1}^{N} \left(\sum_{t=1}^{T} \nabla_\theta \log \pi_\theta\left(\mathbf{a}_t^{(i)} | \mathbf{s}_t^{(i)}\right)\right)\left(\sum_{t=1}^{T} r_t^{(i)}\right)$$

⟨ Distribuer les récompenses à l'intérieur de la somme des politiques ⟩

$$= \frac{1}{N}\sum_{i=1}^{N} \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta\left(\mathbf{a}_t^{(i)} | \mathbf{s}_t^{(i)}\right)\left(\sum_{t'=1}^{T} r_{t'}^{(i)}\right)$$

⟨ Appliquer le principe de causalité en changeant l'index de départ du retour $t'=1 \rightarrow t'=t$ ⟩

$$\geq \frac{1}{N}\sum_{i=1}^{N} \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta\left(\mathbf{a}_t^{(i)} | \mathbf{s}_t^{(i)}\right)\left(\sum_{t'=t}^{T} r_{t'}^{(i)}\right)$$

⟨ **« reward-to-go »** ⟩

$$= \frac{1}{N}\sum_{i=1}^{N} \sum_{t=1}^{T} \nabla_\theta \log \pi_\theta\left(\mathbf{a}_t^{(i)} | \mathbf{s}_t^{(i)}\right) \widehat{Q}_t\left(\tau^{(i)}\right)$$

Pour aller plus loin

Références

References I

1. Van Hasselt, H., Guez, A. & Silver, D. Deep Reinforcement Learning with Double Q-learning. *Artificial Intelligence* **230,** 173-191. arXiv : 1509.06461. http://arxiv.org/abs/1509.06461 (sept. 2015).
2. Wang, Z. *et al.* Dueling Network Architectures for Deep Reinforcement Learning. arXiv : 1511.06581. http://arxiv.org/abs/1511.06581 (2015).
3. Munos, R., Stepleton, T., Harutyunyan, A. & Bellemare, M. G. *Safe and efficient off-policy reinforcement learning.* in *Advances in Neural Information Processing Systems* (2016). arXiv : 1606.02647.
4. Schaul, T., Quan, J., Antonoglou, I. & Silver, D. *Prioritized experience replay.* in *4th International Conference on Learning Representations, ICLR 2016 - Conference Track Proceedings* (2016). arXiv : 1511.05952.
5. Lillicrap, T. P. *et al.* Continuous control with deep reinforcement learning. arXiv : 1509.02971. http://arxiv.org/abs/1509.02971 (2015).
6. Sutton, R. S. & Barto, A. G. *Reinforcement learning: An introduction.* 2e éd. (éd. MIT Press) ISBN : 978-0262039246. http://incompleteideas.net/book/RLbook2018.pdf (Cambridge, MA, 2018).

References II

7. Lin, L. J. Programming Robots Using Reinforcement Learning and Teaching. *Aaai-91 the Ninth National Conference on Artificial Intelligence,* 781-786. http://www.aaai.org/Library/AAAI/1991/aaai91-122.php (1991).
8. Mnih, V. *et al.* Playing Atari with Deep Reinforcement Learning. 1-9. arXiv : 1312.5602. http://arxiv.org/abs/1312.5602 (2013).
9. Mnih, V. *et al.* Human-level control through deep reinforcement learning. *Nature* **518,** 529-533. ISSN : 14764687. https://web.stanford.edu/class/psych209/Readings/MnihEtAlHassibis15NatureControlDeepRL.pdf (2015).
10. Williams, R. J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning* **8,** 229-256. ISSN : 0885-6125 (1992).